Permutation Routing

Walter Unger

Lehrstuhl für Informatik 1

11:53 Uhr, den 30. Januar 2017
1 Introduction
   - Networks
   - Permutation Networks
   - Matching on Bipartite Graphs

2 Recall
   - Some Basics
   - Theorem of Hall
   - Theorem of Hall

3 Disjoint Path Lemma
   - The Lemma
   - The Proof

4 Routing on Mesh Networks
   - The Problem
   - Simple Routings
   - Annexstein and Baumslag
Properties of the Networks to be considered

- **Number of nodes.**
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- **Number of edges.**
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths.
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- **Degree.**

- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths.

- Connectivity, i.e. is there a bottle-neck.

- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.

- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.

- **Length of paths in the network:**
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths

- Connectivity, i.e. is there a bottle-neck.

- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.

- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.

Length of paths in the network:
- **Diameter**, i.e. the longest of all shortest paths.
- **Radius**, i.e. length of the shortest of all longest shortest paths.

Connectivity, i.e. is there a bottle-neck.

Regularity,
- May be all nodes look ‘similar’.
- May be all edges look ‘similar’.

Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.

Length of pathes in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths

- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.

- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottleneck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
**Properties of the Networks to be considered**

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths.
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottle-neck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.

Easy routing

- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottleneck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. length of the shortest of all longest shortest paths
- Connectivity, i.e. is there a bottleneck.
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Product of Graphs

Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Product of Graphs

Definition:

Let \( G = (V, E) \) and \( G' = (V', E') \) be graphs. With \( G \times G' \) we denote the product of \( G \) and \( G' \):

- \( G \times G' = (V \times V', E_1 \cup E_2) \).
- \( E_1 = \{(a, a'), (b, b') \} \mid a' = b' \land (a, b) \in E \} \).
- \( E_2 = \{(a, a'), (b, b') \} \mid a = b \land (a', b') \in E' \} \).

Example \( L(10) \times C(4) \):
Product of Graphs

Definition:
Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
**Product of Graphs**

**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Product of Graphs

**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Product of Graphs

**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Grid of dimension $d$

- Grids: $G(n_1, n_2, \cdots, n_d) = L(n_1) \times L(n_2) \times \cdots \times L(N_d)$ with $n_i > 1$

- Grid: $G(14, 4)$:

```
0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0 11,0 12,0 13,0
0,1 1,1 2,1 3,1 4,1 5,1 6,1 7,1 8,1 9,1 10,1 11,1 12,1 13,1
0,2 1,2 2,2 3,2 4,2 5,2 6,2 7,2 8,2 9,2 10,2 11,2 12,2 13,2
0,3 1,3 2,3 3,3 4,3 5,3 6,3 7,3 8,3 9,3 10,3 11,3 12,3 13,3
```
Grid of dimension $d$

- Grids: $G(n_1, n_2, \ldots, n_d) = L(n_1) \times L(n_2) \times \cdots \times L(N_d)$ with $n_i > 1$

  - Nodecount: $\prod_{i=1}^{d} n_i$
  - Degrees: $\{d, \ldots, 2 \cdot d\}$
  - Edgecount: $\sum_{i=1}^{d} (n_i - 1) \prod_{j=1, j \neq i}^{d} n_j$
  - Diameter: $\sum_{i=1}^{d} (n_i - 1)$

- Grid: $G(14, 4)$:

```
0,3 1,3 2,3 3,3 4,3 5,3 6,3 7,3 8,3 9,3 10,3 11,3 12,3 13,3
0,2 1,2 2,2 3,2 4,2 5,2 6,2 7,2 8,2 9,2 10,2 11,2 12,2 13,2
0,1 1,1 2,1 3,1 4,1 5,1 6,1 7,1 8,1 9,1 10,1 11,1 12,1 13,1
0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0 11,0 12,0 13,0
```
Torus of dimension $d$

- Torus: $Tr(n_1, n_2, \cdots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d)$ with $n_i > 1$

- Torus: $Tr(14, 4)$:
Torus of dimension $d$

- Torus: $Tr(n_1, n_2, \cdots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d)$ with $n_i > 1$

  Number of nodes: $\prod_{i=1}^{d} n_i$

  Number of edges: $\prod_{i=1}^{d} n_i$

  Degree: $2 \cdot d$

  Diameter: $\sum_{i=1}^{d} \lfloor n_i / 2 \rfloor$

- Torus: $Tr(14, 4)$:
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$
$$V_{HQ(d)} = \{0, 1\}^d$$
$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension \(d\)

\[
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
\]

\[
V_{HQ(d)} = \{0, 1\}^d
\]

\[
E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\]
Hypercube of dimension $d$

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{ \{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)} \} \]
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$
$$V_{HQ(d)} = \{0, 1\}^d$$
$$E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$

\[ HQ(d) = \left( V_{HQ(d)}, E_{HQ(d)} \right) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{ \{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)} \} \]

Note the Gray-Code.
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}$$

Number of nodes: $2^d$

Number of edges: $d \cdot 2^{d-1}$

Degree: $d$

Diameter: $d$

Note the Gray-Code.
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$
$$V_{HQ(d)} = \{0, 1\}^d$$
$$E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

\[
\begin{align*}
HQ(d) &= (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} &= \{0, 1\}^d \\
E_{HQ(d)} &= \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\end{align*}
\]
Hypercube of dimension $d$ (alternative view)

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$ (alternative view)

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$ (alternative view)

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

$$
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
$$

$$
V_{HQ(d)} = \{0, 1\}^d
$$

$$
E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}
$$
Hypercube of dimension $d$ (alternative view)

$$
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
$$
$$
V_{HQ(d)} = \{0, 1\}^d
$$
$$
E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}
$$
Hypercube of dimension $d$ (alternative view)

\[
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
\]

\[
V_{HQ(d)} = \{0, 1\}^d
\]

\[
E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\]
Hypercube of dimension $d$ (alternative view)

$$\begin{align*}
HQ(d) &= (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} &= \{0, 1\}^d \\
E_{HQ(d)} &= \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\end{align*}$$
Hypercube of dimension $d$ (alternative view)

\[
\begin{align*}
HQ(d) &= (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} &= \{0, 1\}^d \\
E_{HQ(d)} &= \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\end{align*}
\]
Cube-Connected Cycles of dimension \( d \)

\[
\begin{align*}
CCC(d) &= (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)}) \\
V_{CCC(d)} &= \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d \\
E^c_{CCC(d)} &= \{(i, w), ((i + 1) \mod d, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < d \\
E^h_{CCC(d)} &= \{((i, w0w'), (i, w1w')) \mid w' \in \{0, 1\}^{n-i-1}, w \in \{0, 1\}^i\}
\end{align*}
\]
Cube-Connected Cycles of dimension $d$

\[ \text{CCC}(d) = \left( V_{\text{CCC}(d)}, E^c_{\text{CCC}(d)} \cup E^h_{\text{CCC}(d)} \right) \]

\[ V_{\text{CCC}(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d \]

\[ E^c_{\text{CCC}(d)} = \{\{(i, w), ((i + 1) \mod d, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < d\} \]

\[ E^h_{\text{CCC}(d)} = \{\{(i, w0w'), (i, w1w')\} \mid w' \in \{0, 1\}^{n-i-1}, w \in \{0, 1\}^i\} \]
Cube-Connected Cycles of dimension $d$

$$CCC(d) = (V_{CCC(d)}, E_{CCC(d)}^c \cup E_{CCC(d)}^h)$$
$$V_{CCC(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d$$
$$E_{CCC(d)}^c = \{\{(i, w), ((i + 1) \mod d, w)\} | w \in \{0, 1\}^d, 0 \leq i < d\}$$
$$E_{CCC(d)}^h = \{\{(i, w0w'), (i, w1w')\} | w' \in \{0, 1\}^{n-i-1}, w \in \{0, 1\}^i\}$$
Cube-Connected Cycles of dimension $d$

$$CCC(d) = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)})$$

$$V_{CCC(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{CCC(d)} = \{(i, w), ((i + 1) \mod d, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < d$$

$$E^h_{CCC(d)} = \{(i, w0w'), (i, w1w')\} \mid w' \in \{0, 1\}^{n - i - 1}, w \in \{0, 1\}^i$$
Cube-Connected Cycles of dimension $d$

\[
\begin{align*}
CCC(d) & = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)}) \\
V_{CCC(d)} & = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d \\
E^c_{CCC(d)} & = \{(i, w), ((i + 1) \mod d, w) \mid w \in \{0, 1\}^d, 0 \leq i < d\} \\
E^h_{CCC(d)} & = \{(i, w0w'), (i, w1w') \mid w' \in \{0, 1\}^{n-i-1}, w \in \{0, 1\}^i\}
\end{align*}
\]
Cube-Connected Cycles of dimension $d$

$$CCC(d) = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)})$$

$$V_{CCC(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{CCC(d)} = \{(i, w), ((i + 1) \mod d, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < d$$

$$E^h_{CCC(d)} = \{((i, w0w'), (i, w1w'))\} \mid w' \in \{0, 1\}^{n-i-1}, w \in \{0, 1\}^i$$

Number of nodes: $d \cdot 2^d$

Degree: 3

Number of edges: $3 \cdot d \cdot 2^{d-1}$

Diameter: $2 \cdot d - 2 + \lceil d/2 \rceil$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E_{BF(d)}^c \cup E_{BF(d)}^h) = \{((i, w0w'), (i, w1w')) | w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

$$V_{BF(d)} = \{0, 1, \cdots, d-1\} \times \{0, 1\}^d$$

$$E_{BF(d)}^c = \{((i, w), ((i+1) \mod d, w)) | w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E_{BF(d)}^h = \{((i, w0w'), ((i+1) \mod d, w1w')) | w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E_{BF(d)}^c \cup E_{BF(d)}^h) = \{(i, w0w'), (i, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$$
$$V_{BF(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d$$
$$E_{BF(d)}^c = \{((i, w), ((i + 1) \mod d, w')) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$
$$E_{BF(d)}^h = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{nCCC(d)}) = \{(i, w0w'), (i, w1w') \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

$$V_{BF(d)} = \{0, 1, \cdots, d-1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \text{ mod } d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \text{ mod } d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)}) = \{(i, w0w'), (i, w1w') \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

$$V_{BF(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{nCC(d)}) = \{(i, w0w'), (i, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\)$$

$$V_{BF(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{CCC(d)}) = \{((i, w0w'), (i, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

$$V_{BF(d)} = \{0, 1, \cdots, d-1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{nCC(d)}) = \{(i, w0w'), (i, w1w') \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

$$V_{BF(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{(i, w), ((i + 1) \mod d, w) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E^h_{BF(d)} = \{(i, w0w'), ((i + 1) \mod d, w1w') \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$
Butterfly of dimension $d$

\[ BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{nccc(d)}) = \{(i, w0w'), (i, w1w') \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\} \]

\[ V_{BF(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d \]

\[ E^c_{BF(d)} = \{((i, w), ((i + 1) \mod d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\} \]

\[ E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\} \]
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E_{BF(d)}^{c} \cup E_{BF(d)}^{h})$$

$$V_{BF(d)} = \{0, 1, \ldots, d-1\} \times \{0, 1\}^d$$

$$E_{BF(d)}^{c} = \{(i, w), ((i+1) \mod d, w) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E_{BF(d)}^{h} = \{(i, w0w'), ((i+1) \mod d, w1w') \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$

Number of nodes: $d \cdot 2^d$

Degree: 4

Number of edges: $d \cdot 2^{d+1}$

Diameter: $d + \lfloor d/2 \rfloor$
DeBruijn network of dimension $d$

- **DeBruijn network:**
  \[
  DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se})
  \]
  \[
  V_{DB(d)} = \{0, 1\}^d
  \]
  \[
  E_{DB(d)}^s = \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}
  \]
  \[
  E_{DB(d)}^{se} = \{(aw, wb) | a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}
  \]
DeBruijn network of dimension $d$

- **DeBruijn network:**
  
  \[ DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se}) \]
  
  \[ V_{DB(d)} = \{0, 1\}^d \]
  
  \[ E_{DB(d)}^s = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\} \]
  
  \[ E_{DB(d)}^{se} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\} \]
DeBruijn network of dimension $d$

- DeBruijn network:
  \[
  DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se})
  \]
  \[
  V_{DB(d)} = \{0, 1\}^d
  \]
  \[
  E_{DB(d)}^s = \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}
  \]
  \[
  E_{DB(d)}^{se} = \{(aw, wb) | a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}
  \]
DeBruijn network of dimension $d$

- DeBruijn network:
  \[
  DB(d) = \left( V_{DB(d)}, E^s_{DB(d)} \cup E^{se}_{DB(d)} \right)
  \]
  \[
  V_{DB(d)} = \{0, 1\}^d
  \]
  \[
  E^s_{DB(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}
  \]
  \[
  E^{se}_{DB(d)} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}
  \]
DeBruijn network of dimension $d$

- DeBruijn network:
  \[ DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se}) \]
  \[ V_{DB(d)} = \{0, 1\}^d \]
  \[ E_{DB(d)}^s = \{ (aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)} \} \]
  \[ E_{DB(d)}^{se} = \{ (aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)} \} \]

- Number of nodes: $2^d$
- Degree: $2 + 2$
- Number of edges: $2^{d+1}$
- Diameter: $d$
Shuffle-Exchange network of dimension \( d \)

- Shuffle-Exchange network:
  
  \[
  \begin{align*}
  SE(d) &= (V_{SE(d)}, E^s_{SE(d)} \cup E^e_{SE(d)}) \\
  V_{SE(d)} &= \{0, 1\}^d \\
  E^s_{SE(d)} &= \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{SE(d)}\} \\
  E^e_{SE(d)} &= \{(wa, wb) | a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}
  \end{align*}
  \]
Shuffle-Exchange network of dimension $d$

Shuffle-Exchange network:

$$SE(d) = (V_{SE(d)}, E_{SE(d)}^s \cup E_{SE(d)}^e)$$

$$V_{SE(d)} = \{0, 1\}^d$$

$$E_{SE(d)}^s = \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{SE(d)}\}$$

$$E_{SE(d)}^e = \{(wa, wb) | a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}$$
Shuffle-Exchange network of dimension $d$

- **Shuffle-Exchange network:**
  
  \[
  \begin{align*}
  SE(d) &= (V_{SE(d)}, E_{SE(d)}^s \cup E_{SE(d)}^e) \\
  V_{SE(d)} &= \{0, 1\}^d \\
  E_{SE(d)}^s &= \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{SE(d)}\} \\
  E_{SE(d)}^e &= \{(wa, wb) \mid a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}
  \end{align*}
  \]
Shuffle-Exchange network of dimension $d$

- Shuffle-Exchange network:
  \[ SE(d) = (V_{SE(d)}, E^s_{SE(d)} \cup E^e_{SE(d)}) \]
  \[ V_{SE(d)} = \{0, 1\}^d \]
  \[ E^s_{SE(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{SE(d)}\} \]
  \[ E^e_{SE(d)} = \{(wa, wb) \mid a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\} \]

  Number of nodes: $2^d$
  Degree: $2 + 2$
  Number of edges: $2^{d+1}$
  Diameter: $2 \cdot d - 1$
Recall Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E_{BF(d)}^c \cup E_{BF(d)}^h)$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E_{BF(d)}^c = \{\{(i, w), ((i + 1) \mod d, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$

$$E_{BF(d)}^h = \{\{(i, w0w'), ((i + 1) \mod d, w1w')\} \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$

Number of nodes: $d \cdot 2^d$
Degree: 4
Number of edges: $d \cdot 2^{d+1}$
Diameter: $d + \lceil d/2 \rceil$
Recall Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$
$$V_{BF(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d$$
$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod d, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$
$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod d, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\}$$

Number of nodes: $d \cdot 2^d$
Degree: 4
Number of edges: $d \cdot 2^{d+1}$
Diameter: $d + \lfloor d/2 \rfloor$
Unwrapped Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$
$$V_{BF(d)} = \{0, \ldots, d\} \times \{0, 1\}^d$$
$$E^c_{BF(d)} = \{((i, w), (i + 1, w)) \mid w \in \{0, 1\}^d, 0 \leq i < d\}$$
$$E^h_{BF(d)} = \{((i, w0w'), (i + 1, w1w')) \mid w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i, 0 \leq i < d\}$$

Number of nodes: $(d + 1) \cdot 2^d$
Number of edges: $d \cdot 2^{d+1}$
Permutation network

\[ PN(d) = (V_{PN(d)}, E^c_{PN(d)} \cup E^h_{PN(d)}) \]
\[ V_{PN(d)} = \{1, 2, \ldots, d, -1, -2, \ldots, -d\} \times \{0, 1\}^d \]
\[ E^c_{PN(d)} = \{\{(i, w), (i + 1, w)\} | w \in \{0, 1\}^d, 1 \leq i < d\} \]
\[ \quad \cup \{\{(1, w), (-1, w)\} | w \in \{0, 1\}^d\} \]
\[ \quad \cup \{\{(-i, w), (-i - 1, w)\} | w \in \{0, 1\}^d, 1 \leq i < d\} \]
\[ E^h_{PN(d)} = \{\{(i, w0w'), (i + 1, w1w')\} | w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\} \]
\[ \quad \cup \{\{(1, w0w'), (-1, w1w')\} | w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\} \]
\[ \quad \cup \{\{(-i, w0w'), (-i - 1, w1w')\} | w \in \{0, 1\}^{n-i-1}, w' \in \{0, 1\}^i\} \]
Large Example Permutation network
Extended Permutation network

\[ PN(n, d) = (V_{PN(n,d)}, E^c_{PN(n,d)} \cup E^h_{PN(n,d)}) \]
\[ V_{PN(d)} = \{1, 2, \ldots, d, -1, -2, \ldots, -d\} \times \{0, \ldots, n-1\}^d \]
\[ E^c_{PN(n,d)} = \{(i, w), (i + 1, w)\} | w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \]
\[ \cup \{(1, w), (-1, w)\} | w \in \{0, \ldots, n-1\}^d \]
\[ \cup \{(-i, w), (-i - 1, w)\} | w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \]
\[ E^h_{PN(n,d)} = \{(i, w0w'), (i + 1, w1w')\} | w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^{i} \]
\[ \cup \{(1, w0w'), (-1, w1w')\} | w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^{i} \]
\[ \cup \{(-i, w0w'), (-i - 1, w1w')\} | w \in \{0, \ldots, n-1\}^{*}, w' \in \{0, \ldots, n-1\}^{i} \]

- The 2d \cdot n^d nodes of (n, d)-PN are partitioned into 2d levels and n^d columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length d over \(\{0, 1, \ldots, n-1\}\).
- The parameter d is called the dimension of the network.
- The nodes on level \(-d\) [resp. d] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Extended Permutation network

\begin{align*}
PN(n, d) & = (V_{PN(n,d)}, E^c_{PN(d)} \cup E^h_{PN(n,d)}) \\
V_{PN(d)} & = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n-1\}^d \\
E^c_{PN(n,d)} & = \{(i, w), (i + 1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \\
E^h_{PN(n,d)} & = \{(1, w), (-1, w)\} \mid w \in \{0, \ldots, n-1\}^d \\
& \cup \{(i, w), (i - 1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \\
& \cup \{(i, w0w'), (i + 1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \\
& \cup \{(1, w0w'), (-1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \\
& \cup \{(-i, w0w'), (-i - 1, w1w')\} \mid w \in \{0, \ldots, n-1\}^*, w' \in \{0, \ldots, n-1\}^i \\
\end{align*}

- The $2d \cdot n^d$ nodes of $(n, d)$-PN are partitioned into $2d$ levels and $n^d$ columns.
- Levels are numbered $-d, \ldots, -1, 1, \ldots, d$.
- Columns are labeled with strings of length $d$ over $\{0, 1, \ldots, n-1\}$.
- The parameter $d$ is called the dimension of the network.
- The nodes on level $-d$ [resp. $d$] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network $(n, 1)$ is complete (all possible connections).
Extended Permutation network

\[ PN(n, d) = (V_{PN(n,d)}, E^c_{PN(d)} \cup E^h_{PN(n,d)}) \]

\[ V_{PN(d)} = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n-1\}^d \]

\[ E^c_{PN(n,d)} = \{(i, w), (i + 1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \]

\[ \cup \{(1, w), (-1, w)\} \mid w \in \{0, \ldots, n-1\}^d \]

\[ \cup \{(-i, w), (-i - 1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \]

\[ E^h_{PN(n,d)} = \{(i, w0w'), (i + 1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \]

\[ \cup \{(1, w0w'), (-1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \]

\[ \cup \{(-i, w0w'), (-i - 1, w1w')\} \mid w \in \{0, \ldots, n-1\}^*, w' \in \{0, \ldots, n-1\}^i \]

- The 2\(d \cdot n^d\) nodes of \((n, d)\)-PN are partitioned into 2\(d\) levels and \(n^d\) columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length \(d\) over \(\{0, 1, \ldots, n-1\}\).
- The parameter \(d\) is called the dimension of the network.
- The nodes on level \(-d\) [resp. \(d\)] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Extended Permutation network

\[ PN(n, d) = (V_{PN(n,d)}, E^c_{PN(n,d)} \cup E^h_{PN(n,d)}) \]

\[ V_{PN(d)} = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n - 1\}^d \]

\[ E^c_{PN(n,d)} = \{(i, w), (i + 1, w)\} \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d \]

\[ E^h_{PN(n,d)} = \{(1, w), (-1, w)\} \mid w \in \{0, \ldots, n - 1\}^d \]

\[ \cup \{(i, w), (-i, w)\} \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d \]

\[ E^h_{PN(n,d)} = \{(i, w0w'), (i + 1, w1w')\} \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i \]

\[ \cup \{(1, w0w'), (-1, w1w')\} \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i \]

\[ \cup \{(-i, w0w'), (-i - 1, w1w')\} \mid w \in \{0, \ldots, n - 1\}^*, w' \in \{0, \ldots, n - 1\}^i \]

- The \(2d \cdot n^d\) nodes of \((n, d)\)-PN are partitioned into \(2d\) levels and \(n^d\) columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length \(d\) over \(\{0, 1, \ldots, n - 1\}\).
- The parameter \(d\) is called the dimension of the network.
- The nodes on level \(-d\) [resp. \(d\)] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Extended Permutation network

\[ PN(n, d) = (V_{PN(n,d)}, E_{PN(n,d)}^c \cup E_{PN(n,d)}^h) \]
\[ V_{PN(d)} = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n - 1\}^d \]
\[ E_{PN(n,d)}^c = \{((i, w), (i + 1, w)) \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d\} \]
\[ \cup \{((1, w), (-1, w)) \mid w \in \{0, \ldots, n - 1\}^d\} \]
\[ \cup \{((-i, w), (-i - 1, w)) \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d\} \]
\[ E_{PN(n,d)}^h = \{((i, w0w'), (i + 1, w1w')) \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i\} \]
\[ \cup \{(1, w0w'), (-1, w1w') \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i\} \]
\[ \cup \{((-i, w0w'), (-i - 1, w1w')) \mid w \in \{0, \ldots, n - 1\}^*, w' \in \{0, \ldots, n - 1\}^i\} \]

- The \(2d \cdot n^d\) nodes of \((n, d)\)-PN are partitioned into \(2d\) levels and \(n^d\) columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length \(d\) over \(\{0, 1, \ldots, n - 1\}\).
- The parameter \(d\) is called the dimension of the network.
- The nodes on level \(-d\) [resp. \(d\)] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Extended Permutation network

$PN(n, d) = (V_{PN(n,d)}, E^c_{PN(d)} \cup E^h_{PN(n,d)})$

$V_{PN(d)} = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n-1\}^d$

$E^c_{PN(n,d)} = \{((i, w), (i+1, w)) | w \in \{0, \ldots, n-1\}^d, 1 \leq i < d\}$

$\cup \{((1, w), (-1, w)) | w \in \{0, \ldots, n-1\}^d\}$

$\cup \{((-i, w), (-i-1, w)) | w \in \{0, \ldots, n-1\}^d, 1 \leq i < d\}$

$E^h_{PN(n,d)} = \{((i, w0w'), (i+1, w1w')) | w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i\}$

$\cup \{(1, w0w'), (-1, w1w') | w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i\}$

$\cup \{((-i, w0w'), (-i-1, w1w')) | w \in \{0, \ldots, n-1\}^*, w' \in \{0, \ldots, n-1\}^i\}$

- The $2d \cdot n^d$ nodes of $(n, d)$-PN are partitioned into $2d$ levels and $n^d$ columns.
- Levels are numbered $-d, \ldots, -1, 1, \ldots, d$.
- Columns are labeled with strings of length $d$ over $\{0, 1, \ldots, n-1\}$.
- The parameter $d$ is called the dimension of the network.
- The nodes on level $-d$ [resp. $d$] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network $(n, 1)$ is complete (all possible connections).
Extended Permutation network

\[
\begin{align*}
PN(n, d) & = (V_{PN(n,d)}, E_{PN(n,d)}^c \cup E_{PN(n,d)}^h) \\
V_{PN(d)} & = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n - 1\}^d \\
E_{PN(n,d)}^c & = \{((i, w), (i + 1, w)) \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d\} \\
& \cup \{((1, w), (-1, w)) \mid w \in \{0, \ldots, n - 1\}^d\} \\
& \cup \{((-i, w), (-i - 1, w)) \mid w \in \{0, \ldots, n - 1\}^d, 1 \leq i < d\} \\
E_{PN(n,d)}^h & = \{((i, w0w'), (i + 1, w1w')) \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i\} \\
& \cup \{((1, w0w'), (-1, w1w')) \mid w \in \{0, \ldots, n - 1\}^{n-i-1}, w' \in \{0, \ldots, n - 1\}^i\} \\
& \cup \{((-i, w0w'), (-i - 1, w1w')) \mid w \in \{0, \ldots, n - 1\}^*, w' \in \{0, \ldots, n - 1\}^i\}
\end{align*}
\]

- The \(2d \cdot n^d\) nodes of \((n, d)\)-PN are partitioned into \(2d\) levels and \(n^d\) columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length \(d\) over \(\{0, 1, \ldots, n - 1\}\).
- The parameter \(d\) is called the dimension of the network.
- The nodes on level \(-d\) [resp. \(d\)] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Extended Permutation network

\[ PN(n, d) = (V_{PN(n,d)}, E^c_{PN(n,d)} \cup E^h_{PN(n,d)}) \]

\[ V_{PN(d)} = \{1, 2, \cdots, d, -1, -2, \cdots, -d\} \times \{0, \ldots, n-1\}^d \]

\[ E^c_{PN(n,d)} = \{(i, w), (i+1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \}

\[ \cup \{(1, w), (-1, w)\} \mid w \in \{0, \ldots, n-1\}^d \]

\[ \cup \{(-i, w), (-i-1, w)\} \mid w \in \{0, \ldots, n-1\}^d, 1 \leq i < d \}

\[ E^h_{PN(n,d)} = \{(i, w0w'), (i+1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \}

\[ \cup \{(1, w0w'), (-1, w1w')\} \mid w \in \{0, \ldots, n-1\}^{n-i-1}, w' \in \{0, \ldots, n-1\}^i \}

\[ \cup \{(-i, w0w'), (-i-1, w1w')\} \mid w \in \{0, \ldots, n-1\}^*, w' \in \{0, \ldots, n-1\}^i \}

- The \(2d \cdot n^d\) nodes of \((n, d)\)-PN are partitioned into \(2d\) levels and \(n^d\) columns.
- Levels are numbered \(-d, \ldots, -1, 1, \ldots, d\).
- Columns are labeled with strings of length \(d\) over \(\{0, 1, \ldots, n-1\}\).
- The parameter \(d\) is called the dimension of the network.
- The nodes on level \(-d\) [resp. \(d\)] are called inputs [resp. outputs].
- Each input (output) is identified with its column label.
- Permutation networks have a recursive structure.
- The Permutation network \((n, 1)\) is complete (all possible connections).
Recall

Definition (Bipartite graph)

A graph $G = (V, E)$ is called bipartite if there exist $U, W \subseteq V$ with $U \cup W = V$ and $\forall e \in E : \exists u \in U, w \in W : e = \{u, w\}$.

Definition (Matching)

For a given Graph $G = (V, E)$, a matching $M \subseteq E$ is a set of non-incident edges, i.e., $\forall e, f \in M : e \cap f = \emptyset$.

Definition (Perfect matching)

A matching $M$ is called perfect, if it contains all nodes from $G$: $\forall v \in V \exists e \in M : v \in e$. 
Theorem of Hall

**Definition**

Let $G = (V_1, V_2, E)$ be a bipartite graph, and $A \subseteq V_1$. We denote:

$$\Gamma(A) = \{v \in V_2 \mid (v, w) \in E, w \in A\}.$$ 

**Theorem (Hall)**

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

**Corollary**

Every regular bipartite Graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$ contains a complete matching.
Theorem of Hall

**Definition**

Let $G = (V_1, V_2, E)$ be a bipartite graph, and $A \subseteq V_1$. We denote:

$$\Gamma(A) = \{ v \in V_2 \mid (v, w) \in E, w \in A \}.$$

**Theorem (Hall)**

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$  

**Corollary**

Every regular bipartite Graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$ contains a complete matching.
Theorem of Hall

Definition
Let $G = (V_1, V_2, E)$ be a bipartite graph, and $A \subseteq V_1$. We denote:

$$\Gamma(A) = \{v \in V_2 \mid (v, w) \in E, w \in A\}.$$ 

Theorem (Hall)
Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

Corollary
Every regular bipartite Graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$ contains a complete matching.
**Proof (Hall)**

**Theorem (Hall)**

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$

⇒ simple:

- Let $M$ be a matching with $|M| = |V_1|$ and let $A \subseteq V_1$ arbitrary.
- $|\Gamma(A)| = \{|v \in V_2 \mid (v, w) \in E, w \in A\|$. 
- $|\Gamma(A)| \geq \{|v \in V_2 \mid (v, w) \in M, w \in A\|$. 
- $|\Gamma(A)| \geq |A|$. 

---

**Introduction**

Recall

**Disjoint Path Lemma**

Routing on Mesh Networks


RWTH
Theorem (Hall)

Let \( G = (V_1, V_2, E) \) be a bipartite graph. There exists a complete matching from \( V_1 \) to \( V_2 \), iff for each \( A \subseteq V_1 \) we have

\[
|\Gamma(A)| \geq |A|.
\]

\[\implies\] simple:

- Let \( M \) be a matching with \( |M| = |V_1| \) and let \( A \subset V_1 \) arbitrary.
- \( |\Gamma(A)| = |\{v \in V_2 | (v, w) \in E, w \in A\}|.\)
- \( |\Gamma(A)| \geq |\{v \in V_2 | (v, w) \in M, w \in A\}|.\)
- \( |\Gamma(A)| \geq |A|\).
**Proof (Hall)**

---

**Theorem (Hall)**

*Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have*

$$|\Gamma(A)| \geq |A|.$$  

---

**simple:**

- Let $M$ be a matching with $|M| = |V_1|$ and let $A \subseteq V_1$ arbitrary.
- $|\Gamma(A)| = |\{v \in V_2 \mid (v, w) \in E, w \in A\}|.$
- $|\Gamma(A)| \geq |\{v \in V_2 \mid (v, w) \in M, w \in A\}|.$
- $|\Gamma(A)| \geq |A|.$
Proof (Hall)

**Theorem (Hall)**

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

--- simple:

- Let $M$ be a matching with $|M| = |V_1|$ and let $A \subset V_1$ arbitrary.
- $|\Gamma(A)| = |\{v \in V_2 \mid (v, w) \in E, w \in A\}|.$
- $|\Gamma(A)| \geq |\{v \in V_2 \mid (v, w) \in M, w \in A\}|.$
- $|\Gamma(A)| \geq |A|.$
**Proof (Hall)**

**Theorem (Hall)**

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$  

\[ \implies \text{simple:} \]

- Let $M$ be a matching with $|M| = |V_1|$ and let $A \subseteq V_1$ arbitrary.
- $|\Gamma(A)| = |\{ v \in V_2 \mid (v, w) \in E, w \in A \}|$.
- $|\Gamma(A)| \geq |\{ v \in V_2 \mid (v, w) \in M, w \in A \}|$.
- $|\Gamma(A)| \geq |A|$.
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$

$\Leftarrow$ by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 | \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 | \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 

$\Rightarrow$ by contradiction:

- Let $M$ be the largest matching with $|M| < \frac{1}{2}|V_1|$.
- Let $A_1 = \{v \in V_1 | \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 | \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 

Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

$\iff$ by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 | \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 | \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subseteq A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A'_1 \cup A'_2$ with $A'_i \subseteq A_i$ and $|A'_1| = |A'_2|$.
- Thus we have $\Gamma(A'_1 \cup \{a\}) \subseteq A'_2$.
- $|A'_1 \cup \{a\}| > |A'_2|$.
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

$\Leftarrow$ by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 

$\Rightarrow$ by induction: 

- Let $M$ be a complete matching.
- Let $A \subseteq V_1$.
- Let $a \in A_1 \setminus A$.
- Then $\Gamma(a) \subset A_2$
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 

Recall Disjoint Path Lemma
Concluding Routing on Mesh Networks
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

$\Leftarrow$ by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subset A_i$ and $|A_i'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subset A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 
Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exists a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

\[ \Leftarrow \text{ by contradiction:} \]

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subseteq A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subseteq A_i$ and $|A_1'| = |A_2'|$.
- Thus we have $\Gamma(A_1' \cup \{a\}) \subseteq A_2'$.
- $|A_1' \cup \{a\}| > |A_2'|$. 
Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$ 

\[\leftarrow\text{by contradiction:}\]

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subseteq A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A'_1 \cup A'_2$ with $A'_i \subseteq A_i$ and $|A'_1| = |A'_2|$.
- Thus we have $\Gamma(A'_1 \cup \{a\}) \subseteq A'_2$.
- $|A'_1 \cup \{a\}| > |A'_2|$.
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$

$\leftarrow$ by contradiction:

1. Let $M$ be the largest matching with $|M| < |V_1|$.
2. Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
3. Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
4. Let $a \in V_1 \setminus A_1$.
5. $\Gamma(a) \subseteq A_2$, because $M$ is the largest matching.
6. Any alternating path starting from $a$ reaches only nodes in $A_1' \cup A_2'$ with $A_i' \subseteq A_i$ and $|A_1'| = |A_2'|$.
7. Thus we have $\Gamma(A_1' \cup \{a\}) \subseteq A_2'$.
8. $|A_1' \cup \{a\}| > |A_2'|$. 

\[\]
Proof (Hall)

Theorem (Hall)

Let $G = (V_1, V_2, E)$ be a bipartite graph. There exits a complete matching from $V_1$ to $V_2$, iff for each $A \subseteq V_1$ we have

$$|\Gamma(A)| \geq |A|.$$

$\Longleftarrow$ by contradiction:

- Let $M$ be the largest matching with $|M| < |V_1|$.
- Let $A_1 = \{v \in V_1 \mid \exists b \in V_2 : \{v, b\} \in M\}$.
- Let $A_2 = \{v \in V_2 \mid \exists b \in V_1 : \{v, b\} \in M\}$.
- Let $a \in V_1 \setminus A_1$.
- $\Gamma(a) \subset A_2$, because $M$ is the largest matching.
- Any alternating path starting from $a$ reaches only nodes in $A'_1 \cup A'_2$ with $A'_i \subset A_i$ and $|A'_1| = |A'_2|$.
- Thus we have $\Gamma(A'_1 \cup \{a\}) \subset A'_2$.
- $|A'_1 \cup \{a\}| > |A'_2|$. 
Definition (Edge coloring)

Let $G = (V, E)$ be a graph. 
\[ \psi : E \rightarrow \{1, \ldots, k\} \]
is an edge coloring if every pair of incident edges $e_1, e_2$ is colored in different colors, i.e., $\psi(e_1) \neq \psi(e_2)$.

Definition

The Edge-Colouring-Problem for a graph $G$ corresponds to the node-colouring of $L(G)$:
\[ \chi'(G) = \chi(L(G)) \].

Theorem (Vizing 1965)

\[ \chi'(K_{2n}) = 2n - 1 \quad \text{and} \quad \chi'(K_{2n+1}) = 2n + 1 \].

Theorem

\[ \chi'(G) \geq \omega(L(G)) \geq \Delta(G). \]
Definition (Edge coloring)

Let $G = (V, E)$ be a graph.
$\psi : E \rightarrow \{1, \ldots, k\}$ is an edge coloring if every pair of incident edges $e_1, e_2$ is colored in different colors, i.e., $\psi(e_1) \neq \psi(e_2)$.

Definition

The Edge-Colouring-Problem for a graph $G$ corresponds to the node-colouring of $L(G)$:
$\chi'(G) = \chi(L(G))$.

Theorem (Vizing 1965)

$\chi'(K_{2n}) = 2n - 1$ and $\chi'(K_{2n+1}) = 2n + 1$.

Theorem

$\chi'(G) \geq \omega(L(G)) \geq \Delta(G)$.
**Definition (Edge coloring)**

Let $G = (V, E)$ be a graph. $\psi : E \rightarrow \{1, ..., k\}$ is an edge coloring if every pair of incident edges $e_1, e_2$ is colored in different colors, i.e., $\psi(e_1) \neq \psi(e_2)$.

**Definition**

The Edge-Colouring-Problem for a graph $G$ corresponds to the node-colouring of $L(G)$:

$\chi'(G) = \chi(L(G))$.

**Theorem (Vizing 1965)**

$\chi'(K_{2n}) = 2n - 1$ and $\chi'(K_{2n+1}) = 2n + 1$.

**Theorem**

$\chi'(G) \geq \omega(L(G)) \geq \Delta(G)$.
**Definition (Edge coloring)**

Let $G = (V, E)$ be a graph. 

$\psi : E \rightarrow \{1, ..., k\}$ is an **edge coloring** if every pair of incident edges $e_1, e_2$ is colored in different colors, i.e., $\psi(e_1) \neq \psi(e_2)$.

**Definition**

The Edge-Colouring-Problem for a graph $G$ corresponds to the node-colouring of $L(G)$:

$\chi'(G) = \chi(L(G))$.

**Theorem (Vizing 1965)**

$\chi'(K_{2n}) = 2n - 1$ and $\chi'(K_{2n+1}) = 2n + 1$.

**Theorem**

$\chi'(G) \geq \omega(L(G)) \geq \Delta(G)$. 
Definition (Regular graphs)
A graph is called \( n \)-regular, \( n \in \mathbb{N} \), if all nodes have the same degree \( n \).

Theorem
A bipartite \( n \)-regular graph \( G = (V_1 \cup V_2, E) \) has an edge coloring with \( n \) colors.

Theorem (Holyer)
The \( d \)-Edge-Colouring-Problem is NP-complete for \( d \geq 3 \).

Theorem (König 1916)
Any bipartite graph with degree \( \Delta \) is \( \Delta \) edge-colourable (Running-Time \( O(nm) \)).

Theorem (Vizing 1964)
Any graph with degree \( \Delta \) is \( \Delta + 1 \) edge-colourable (Running-Time \( O(nm) \)).
Definition (Regular graphs)

A graph is called $n$-regular, $n \in \mathbb{N}$, if all nodes have the same degree $n$.

Theorem

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Theorem (Holyer)

The $d$-Edge-Colouring-Problem is NP-complete for $d \geq 3$.

Theorem (König 1916)

Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

Theorem (Vizing 1964)

Any graph with degree $\Delta$ is $\Delta + 1$ edge-colourable (Running-Time $O(nm)$).
Edge-Colouring II

Definition (Regular graphs)
A graph is called $n$-regular, $n \in \mathbb{N}$, if all nodes have the same degree $n$.

Theorem
A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Theorem (Holyer)
The $d$-Edge-Colouring-Problem is NP-complete for $d \geq 3$.

Theorem (König 1916)
Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

Theorem (Vizing 1964)
Any graph with degree $\Delta$ is $\Delta + 1$ edge-colourable (Running-Time $O(nm)$).
Definition (Regular graphs)
A graph is called $n$-regular, $n \in \mathbb{N}$, if all nodes have the same degree $n$.

Theorem
A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Theorem (Holyer)
The $d$-Edge-Colouring-Problem is NP-complete for $d \geq 3$.

Theorem (König 1916)
Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

Theorem (Vizing 1964)
Any graph with degree $\Delta$ is $\Delta + 1$ edge-colourable (Running-Time $O(nm)$).
Simple Proof

Theorem

A bipartite $n$-regular graph $G = (V_1 \dot{\cup} V_2, E)$ has an edge coloring with $n$ colors.

Proof:

- **We use an induction on the node degree $n$.**
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$  
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.  
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.  
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.  
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

Theorem

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Proof:

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

**Theorem**

A bipartite \( n \)-regular graph \( G = (V_1 \cup V_2, E) \) has an edge coloring with \( n \) colors.

**Proof:**

- We use an induction on the node degree \( n \).
- **Base Case:** For \( n = 1 \) the statement is trivially true.
- **Induction step:** Let \( n > 1 \).

  - **Claim:** \( \forall S \subseteq V_1 : |\Gamma(S)| \geq |S| \)
    - **Proof:** The number of edges from \( S \) into \( \Gamma_G(S) \) is \( k := n \cdot |S| \).
    - Hence, \( \Gamma_G(S) \) has at least \( k \) incident edges.
    - Each node in \( \Gamma(S) \) is incident to at most \( n \) of these \( k \) edges.
    - Hence, \( |\Gamma_G(S)| \geq k/n = |S| \).

  - Now Hall’s theorem implies that \( G \) has a perfect matching \( M \).
  - The edges of \( M \) get assigned color \( n - 1 \).
  - The remaining graph is \( n - 1 \)-regular and, by our induction hypothesis, can be colored with the remaining colors \( \{0, 1, \ldots, n - 2\} \).
Simple Proof

Theorem

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Proof:

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.

- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

Theorem

A bipartite $n$-regular graph $G = (V_1 \hat{\cup} V_2, E)$ has an edge coloring with $n$ colors.

Proof:

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n-1$.
- The remaining graph is $n-1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n-2\}$.
**Simple Proof**

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$.
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - **Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.**
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.

- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

Theorem

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

Proof:

- We use an induction on the node degree $n$.
- **Base Case**: For $n = 1$ the statement is trivially true.
- **Induction step**: Let $n > 1$.
- **Claim**: $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof**: The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.

- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$.
Simple Proof

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.

Now Hall’s theorem implies that $G$ has a perfect matching $M$.

- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- **Now Hall’s theorem implies that $G$ has a perfect matching $M$.**
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 
Simple Proof

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$.
Simple Proof

**Theorem**

A bipartite $n$-regular graph $G = (V_1 \cup V_2, E)$ has an edge coloring with $n$ colors.

**Proof:**

- We use an induction on the node degree $n$.
- **Base Case:** For $n = 1$ the statement is trivially true.
- **Induction step:** Let $n > 1$.
- **Claim:** $\forall S \subseteq V_1 : |\Gamma(S)| \geq |S|$
  - **Proof:** The number of edges from $S$ into $\Gamma_G(S)$ is $k := n \cdot |S|$.
  - Hence, $\Gamma_G(S)$ has at least $k$ incident edges.
  - Each node in $\Gamma(S)$ is incident to at most $n$ of these $k$ edges.
  - Hence, $|\Gamma_G(S)| \geq k/n = |S|$.
- Now Hall’s theorem implies that $G$ has a perfect matching $M$.
- The edges of $M$ get assigned color $n - 1$.
- The remaining graph is $n - 1$-regular and, by our induction hypothesis, can be colored with the remaining colors $\{0, 1, \ldots, n - 2\}$. 


Proof (König)

**Theorem (König)**

*Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).*

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

**Theorem (König)**

*Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).*

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

Theorem (König)

*Any bipartite graph with degree \( \Delta \) is \( \Delta \) edge-colourable (Running-Time \( O(nm) \)).*

- Show how to colour an edge \((a, b)\) in \( O(n) \) time.
- Let \( c_a, c_b \) be the unused colours at the nodes \( a, b \).
- If \( c_a = c_b \), we may colour \((a, b)\) with \( c_a \).
- Observe now the graph \( H_{a, b} \), who consists only of edges coloured with \( c_a, c_b \).
- \( H_{a, b} \) consists of a disjoined set of paths and cycles.
- \( a \) and \( b \) are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

Theorem (König)

Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
**Proof (König)**

Theorem (König)

*Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).*

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

Theorem (König)

Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

**Theorem (König)**

*Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).*

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a, b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a, b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Proof (König)

**Theorem (König)**

Any bipartite graph with degree $\Delta$ is $\Delta$ edge-colourable (Running-Time $O(nm)$).

- Show how to colour an edge $(a, b)$ in $O(n)$ time.
- Let $c_a, c_b$ be the unused colours at the nodes $a, b$.
- If $c_a = c_b$, we may colour $(a, b)$ with $c_a$.
- Observe now the graph $H_{a,b}$, who consists only of edges coloured with $c_a, c_b$.
- $H_{a,b}$ consists of a disjoined set of paths and cycles.
- $a$ and $b$ are the endpoints of two different paths.
- Thus we may exchange the colours of one path.
- Running-Time: store for each node and colour the corresponding edge.
Disjoint Path Lemma

Lemma (Disjoint Path Lemma)

For every permutation \( \pi : \{0, 1, \ldots, n-1\}^d \to \{0, 1, \ldots, n-1\}^d \), there is a collection of \( n^d \) node disjoint paths in \((n, d)\)-PN that, for every \( a \in \{0, 1, \ldots, n-1\}^d \), contains a path \( W_a \) connecting input \( a \) with output \( \pi(a) \).
Proof

Lemma (Disjoint Path Lemma)

For every permutation \( \pi : \{0, 1, ..., n - 1\}^d \rightarrow \{0, 1, ..., n - 1\}^d \), there is a collection of \( n^d \) node disjoint paths in \((n, d)\)-PN that, for every \( a \in \{0, 1, ..., n - 1\}^d \), contains a path \( W_a \) connecting input \( a \) with output \( \pi(a) \).

Proof:

- **Induction over** \( d \).
- **Base Case:** \( d = 1 \): This case is trivially true since the inputs and the outputs are completely connected in \( P(n, 1) \).
- **Induction step:** \((d - 1) \rightarrow d\).
- Idea is: Recall the recursive description of \((n, d)\)-PN.
Proof

Lemma (Disjoint Path Lemma)

For every permutation \( \pi : \{0, 1, ..., n - 1\}^d \to \{0, 1, ..., n - 1\}^d \), there is a collection of \( n^d \) node disjoint paths in \((n, d)\)-PN that, for every \( a \in \{0, 1, ..., n - 1\}^d \), contains a path \( W_a \) connecting input \( a \) with output \( \pi(a) \).

Proof:

- Induction over \( d \).

- **Base Case:** \( d = 1 \): This case is trivially true since the inputs and the outputs are completely connected in \( P(n, 1) \).

- **Induction step:** \( (d - 1) \to d \).

- Idea is: Recall the recursive description of \((n, d)\)-PN.
Lemma (Disjoint Path Lemma)

For every permutation $\pi : \{0, 1, \ldots, n - 1\}^d \rightarrow \{0, 1, \ldots, n - 1\}^d$, there is a collection of $n^d$ node disjoint paths in $(n, d)$-PN that, for every $a \in \{0, 1, \ldots, n - 1\}^d$, contains a path $W_a$ connecting input $a$ with output $\pi(a)$.

Proof:

- Induction over $d$.

- **Base Case: $d = 1$**: This case is trivially true since the inputs and the outputs are completely connected in $P(n, 1)$.

- **Induction step: $(d - 1) \rightarrow d$**.

- Idea is: Recall the recursive description of $(n, d)$-PN.
Proof (Disjoint Path Lemma)

For every permutation $\pi : \{0, 1, ..., n - 1\}^d \rightarrow \{0, 1, ..., n - 1\}^d$, there is a collection of $n^d$ node disjoint paths in $(n, d)$-PN that, for every $a \in \{0, 1, ..., n - 1\}^d$, contains a path $W_a$ connecting input $a$ with output $\pi(a)$.

Proof:

- **Induction over $d$.**
- **Base Case:** $d = 1$: This case is trivially true since the inputs and the outputs are completely connected in $P(n, 1)$.
- **Induction step:** $(d - 1) \rightarrow d$.
- Idea is: Recall the recursive description of $(n, d)$-PN.
Proof

Lemma (Disjoint Path Lemma)

For every permutation $\pi : \{0, 1, ..., n - 1\}^d \to \{0, 1, ..., n - 1\}^d$, there is a collection of $n^d$ node disjoint paths in $(n, d)$-PN that, for every $a \in \{0, 1, ..., n - 1\}^d$, contains a path $W_a$ connecting input $a$ with output $\pi(a)$.

Proof:

- Induction over $d$.
- **Base Case:** $d = 1$: This case is trivially true since the inputs and the outputs are completely connected in $P(n, 1)$.
- **Induction step:** $(d - 1) \to d$.
- Idea is: Recall the recursive description of $(n, d)$-PN.
An input/output-pair \((a, \pi(a))\) that should be connected by a path is called a request.

For each request, we choose a subnetwork \(B^{(i)}\), \(i \in \{0, 1, ..., n - 1\}\), through which the request is routed.
An input/output-pair \((a, \pi(a))\) that should be connected by a path is called a request.

For each request, we choose a subnetwork \(B^{(i)}, i \in \{0, 1, \ldots, n - 1\}\), through which the request is routed.
The choices of the subnetworks satisfy the following properties:

1. Each input of each subnetwork is used by exactly one of the requests.
2. Each output of each subnetwork is used by exactly one request.
The choices of the subnetworks satisfy the following properties:

1. Each input of each subnetwork is used by exactly one of the requests.
2. Each output of each subnetwork is used by exactly one request.
The choices of the subnetworks satisfy the following properties:

1. Each input of each subnetwork is used by exactly one of the requests.
2. Each output of each subnetwork is used by exactly one request.
Proof (Recursive Step)

This way, for every $i \in \{0, 1, \ldots, n - 1\}$, the requests mapped to $B^{(i)}$ define a permutation.

Thus, these requests can be routed along disjoint paths in $B^{(i)}$ by our induction hypothesis, so that the Disjoint Path Lemma follows.

We have to show how to choose the subnetworks for the requests.
Proof (Recursive Step)

This way, for every \( i \in \{0, 1, \ldots, n - 1\} \), the requests mapped to \( B^{(i)} \) define a permutation.

Thus, these requests can be routed along disjoint paths in \( B^{(i)} \) by our induction hypothesis, so that the Disjoint Path Lemma follows.

We have to show how to choose the subnetworks for the requests.
This way, for every $i \in \{0, 1, \ldots, n - 1\}$, the requests mapped to $B^{(i)}$ define a permutation.

Thus, these requests can be routed along disjoint paths in $B^{(i)}$ by our induction hypothesis, so that the Disjoint Path Lemma follows.

We have to show how to choose the subnetworks for the requests.
Proof (by Conflict Graph)

- Towards this end, we define the following bipartite conflict graph:
  \[ G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi) \]

- The set \( E_\pi \) contains an edge \( e_a = \{u_{\hat{a}}, u_{\hat{b}}\} \)
  - for every request \((a, b)\) with \(b = \pi(a)\), where
  - \(\hat{x}\) drops the leading letter of a string
    \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d, \]
  - i.e., \(\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1}\).

- Edges incident to the same node \(u_x\) represent an input conflict, that is,
  - the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \(i_x\), for some \(i \in \{0, 1, \ldots, n-1\}\).

- Analogously, edges incident to the same node \(v_x\) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

- \(G_\pi\) is \(n\)-regular and bipartite.
**Proof (by Conflict Graph)**

- Towards this end, we define the following bipartite **conflict graph**:
  
  \[ G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi) \]

  - The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)
    - for every request \((a, b)\) with \(b = \pi(a)\), where
    - \(\hat{x}\) drops the leading letter of a string
      \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d, \]
    - i.e., \(\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}\).

  - Edges incident to the same node \(u_x\) represent an **input conflict**, that is,
    - the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \(i x\), for some \(i \in \{0, 1, \ldots, n - 1\}\).

  - Analogously, edges incident to the same node \(v_x\) represent an **output conflict** and the corresponding requests should be routed through different subnetworks as well.

- \(G_\pi\) is \(n\)-regular and bipartite.
Towards this end, we define the following bipartite conflict graph:

$G_\pi = (\{u_x | x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x | x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi)$

The set $E_\pi$ contains an edge $e_a = \{u_\hat{x}, u_\hat{b}\}$

- for every request $(a, b)$ with $b = \pi(a)$, where
- $\hat{x}$ drops the leading letter of a string $x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d$,
- i.e., $\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}$.

Edges incident to the same node $u_x$ represent an input conflict, that is,

the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column $i x$, for some $i \in \{0, 1, \ldots, n - 1\}$.

Analogously, edges incident to the same node $v_x$ represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

$G_\pi$ is $n$-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

$$G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi)$$

The set $E_\pi$ contains an edge $e_a = \{u_\hat{a}, u_\hat{b}\}$

- for every request $(a, b)$ with $b = \pi(a)$, where
  - $\hat{a}$ drops the leading letter of a string
    - $x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d$,
  - i.e., $\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1}$.

Edges incident to the same node $u_x$ represent an input conflict, that is,

- the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column $i\hat{x}$, for some $i \in \{0, 1, \ldots, n-1\}$.

Analogously, edges incident to the same node $v_x$ represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

$G_\pi$ is $n$-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

\[ G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi) \]

The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)

- for every request \((a, b)\) with \( b = \pi(a) \), where
- \( \hat{x} \) drops the leading letter of a string
  \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d, \]
  i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1} \).

Edges incident to the same node \( u_x \) represent an input conflict, that is,

- the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \( ix \), for some \( i \in \{0, 1, \ldots, n-1\} \).

Analogously, edges incident to the same node \( v_x \) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

\( G_\pi \) is \( n \)-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

\[ G_\pi = (\{ u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \} \cup \{ v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \}, E_\pi) \]

The set \( E_\pi \) contains an edge \( e = \{ u_\hat{x}, u_b \} \)

- for every request \((a, b)\) with \( b = \pi(a) \), where
- \( \hat{x} \) drops the leading letter of a string
  \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d, \]
  i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}. \]

Edges incident to the same node \( u_x \) represent an input conflict, that is,

the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \( i x \), for some \( i \in \{0, 1, \ldots, n - 1\} \).

Analogously, edges incident to the same node \( v_x \) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

\( G_\pi \) is \( n \)-regular and bipartite.
Towards this end, we define the following bipartite conflict graph:

\[ G_\pi = (\{ u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \} \cup \{ v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \}, E_\pi) \]

The set \( E_\pi \) contains an edge \( e_a = \{ u_\hat{a}, u_\hat{b} \} \)

- for every request \((a, b)\) with \( b = \pi(a) \), where
- \( \hat{x} \) drops the leading letter of a string
  \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d, \]
  i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1} \).

Edges incident to the same node \( u_x \) represent an input conflict, that is,

- the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \( i x \), for some \( i \in \{0, 1, \ldots, n - 1\} \).

Analogously, edges incident to the same node \( v_x \) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

\( G_\pi \) is \( n \)-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

\[ G_\pi = (\{ u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \} \cup \{ v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1} \}, E_\pi) \]

The set \( E_\pi \) contains an edge \( e_a = \{ u_{\hat{a}}, u_{\hat{b}} \} \)

- for every request \((a, b)\) with \(b = \pi(a)\), where
- \( \hat{x} \) drops the leading letter of a string
  \[ x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d, \]
  i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1} \).

Edges incident to the same node \( u_x \) represent an input conflict, that is,

- the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \( i x \), for some \( i \in \{0, 1, \ldots, n - 1\} \).

Analogously, edges incident to the same node \( v_x \) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

\( G_\pi \) is \( n \)-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

$$G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi)$$

The set $E_\pi$ contains an edge $e_a = \{u_\hat{a}, u_\hat{b}\}$

- for every request $(a, b)$ with $b = \pi(a)$, where
- $\hat{x}$ drops the leading letter of a string $x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d$,
- i.e., $\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}$.

Edges incident to the same node $u_x$ represent an input conflict, that is, the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column $i\times$, for some $i \in \{0, 1, \ldots, n - 1\}$.

Analogously, edges incident to the same node $v_x$ represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

$G_\pi$ is $n$-regular and bipartite.
Proof (by Conflict Graph)

Towards this end, we define the following bipartite conflict graph:

\[ G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi) \]

The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)

- for every request \((a, b)\) with \(b = \pi(a)\), where \(\hat{x}\) drops the leading letter of a string \(x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d\),
- i.e., \(\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}\).

Edges incident to the same node \(u_x\) represent an input conflict, that is,

- the corresponding requests must be routed through different subnetworks as, otherwise, they would share the same subnetwork input in column \(i x\), for some \(i \in \{0, 1, \ldots, n - 1\}\).

Analogously, edges incident to the same node \(v_x\) represent an output conflict and the corresponding requests should be routed through different subnetworks as well.

\(G_\pi\) is \(n\)-regular and bipartite.
Proof (by Conflict Graph)

- \( G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi) \)

- The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{x}, u_\hat{b}\} \)
  - for every request \((a, b)\) with \( b = \pi(a) \), where
  - \( \hat{x} \) drops the leading letter of a string
    - \( x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d \),
  - i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1} \).

- \( G_\pi \) is \( n \)-regular and bipartite.

- By the Coloring Lemma, \( E_\pi \) can be colored with colors \( 0, 1, \ldots, n-1 \).

- For any \( i \in \{0, \ldots, n-1\} \), the edges of color \( i \) build a matching in \( G_\pi \) and, hence, the corresponding requests do not have input or output conflict.

- If all requests of color \( i \in \{0, \ldots, n-1\} \) are routed through subnetwork \( B^{(i)} \) then the Properties 1 and 2 are satisfied.

- This completes the proof of the Disjoint Path Lemma.
Proof (by Conflict Graph)

- \( G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi) \)

- The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)
  - for every request \((a, b)\) with \( b = \pi(a) \), where
  - \( \hat{x} \) drops the leading letter of a string
    \( x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d \),
  - i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1} \).

- \( G_\pi \) is \( n \)-regular and bipartite.
- By the Coloring Lemma, \( E_\pi \) can be colored with colors \( 0, 1, \ldots, n - 1 \).
- For any \( i \in \{0, \ldots, n - 1\} \), the edges of color \( i \) build a matching in \( G_\pi \) and, hence, the corresponding requests do not have input or output conflict.
- If all requests of color \( i \in \{0, \ldots, n - 1\} \) are routed through subnetwork \( B^{(i)} \) then the Properties 1 and 2 are satisfied.
- This completes the proof of the Disjoint Path Lemma.
Proof (by Conflict Graph)

- \( G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi) \)

- The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)
  - for every request \((a, b)\) with \( b = \pi(a) \), where
  - \( \hat{x} \) drops the leading letter of a string
    - \( x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d \),
  - i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1} \).

- \( G_\pi \) is \( n \)-regular and bipartite.

- By the Coloring Lemma, \( E_\pi \) can be colored with colors \( 0, 1, \ldots, n - 1 \).

- For any \( i \in \{0, \ldots, n - 1\} \), the edges of color \( i \) build a matching in \( G_\pi \) and, hence, the corresponding requests do not have input or output conflict.

- If all requests of color \( i \in \{0, \ldots, n - 1\} \) are routed through subnetwork \( B^{(i)} \) then the Properties 1 and 2 are satisfied.

- This completes the proof of the Disjoint Path Lemma.
Proof (by Conflict Graph)

- \(G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n - 1\}^{d-1}\}, E_\pi)\)
- The set \(E_\pi\) contains an edge \(e_a = \{u_\hat{a}, u_\hat{b}\}\)
  - for every request \((a, b)\) with \(b = \pi(a)\), where
  - \(\hat{x}\) drops the leading letter of a string
    \(x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^d\),
  - i.e., \(\hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n - 1\}^{d-1}\).

- \(G_\pi\) is \(n\)-regular and bipartite.
- By the Coloring Lemma, \(E_\pi\) can be colored with colors \(0, 1, \ldots, n - 1\).
- For any \(i \in \{0, \ldots, n - 1\}\), the edges of color \(i\) build a matching in \(G_\pi\)
  and, hence, the corresponding requests do not have input or output conflict.
- If all requests of color \(i \in \{0, \ldots, n - 1\}\) are routed through subnetwork \(B^{(i)}\) then the Properties 1 and 2 are satisfied.
- This completes the proof of the Disjoint Path Lemma.
Proof (by Conflict Graph)

- \( G_\pi = (\{u_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\} \cup \{v_x \mid x \in \{0, 1, \ldots, n-1\}^{d-1}\}, E_\pi) \)

- The set \( E_\pi \) contains an edge \( e_a = \{u_\hat{a}, u_\hat{b}\} \)
  - for every request \((a, b)\) with \( b = \pi(a) \), where
  - \( \hat{x} \) drops the leading letter of a string
  - \( x = x_0, x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^d \)
  - i.e., \( \hat{x} = x_1, \ldots, x_{d-1} \in \{0, 1, \ldots, n-1\}^{d-1} \)

- \( G_\pi \) is \( n \)-regular and bipartite.

- By the Coloring Lemma, \( E_\pi \) can be colored with colors \( 0, 1, \ldots, n-1 \).

- For any \( i \in \{0, \ldots, n-1\} \), the edges of color \( i \) build a matching in \( G_\pi \) and, hence, the corresponding requests do not have input or output conflict.

- If all requests of color \( i \in \{0, \ldots, n-1\} \) are routed through subnetwork \( B^{(i)} \) then the Properties 1 and 2 are satisfied.

- This completes the proof of the Disjoint Path Lemma.
The Routing Problem

Definition (Permutation routing problem)

Let \( G = (V, E) \) be a network. A permutation routing problem is defined by a permutation \( \pi : V \to V \). Each node \( v \in V \) has a message (packet) that shall be routed to node \( \pi(v) \).

Note: We use the synchronous congestion model from Peleg’s book: In each step, each edge can forward one packet in each direction.
The Routing Problem

Definition (Permutation routing problem)

Let $G = (V, E)$ be a network. A permutation routing problem is defined by a permutation $\pi : V \to V$. Each node $v \in V$ has a message (packet) that shall be routed to node $\pi(v)$.

Note: We use the synchronous congestion model from Peleg’s book: In each step, each edge can forward one packet in each direction.
Example

On $M(n, d)$, sending a packet from a source to a destination can be done by using dimension-by-dimension routing, that is, first the packet is routed to the target position with respect to dimension 0, then with respect to dimension 1, and so on.

On the two-dimensional array $M(n, 2)$, this approach is also called row-column routing as a packet is first routed to the target position in the row and then to the target position in the column.
Examples

On the hypercube $M(2, d)$, the paths chosen by dimension-by-dimension routing are called **bit-fixing paths**.
More Routing on Meshes

In the following, let $D$ denote the diameter of the network.

**Observation**

*Every permutation $\pi$ can be routed along dimension-by-dimension paths in at most $D$ steps on $M(n, 1)$ and $M(n, 2)$.***

**Lemma**

*Consider $M(n, 3)$. There is a permutation $\pi$ such that every packet routing algorithm using dimension-by-dimension paths needs at least $\Omega(D^2)$ steps for routing $\pi$.***
More Routing on Meshes

In the following, let $D$ denote the diameter of the network.

**Observation**

*Every permutation $\pi$ can be routed along dimension-by-dimension paths in at most $D$ steps on $M(n, 1)$ and $M(n, 2)$.*

**Lemma**

*Consider $M(n, 3)$. There is a permutation $\pi$ such that every packet routing algorithm using dimension-by-dimension paths needs at least $\Omega(D^2)$ steps for routing $\pi$.***
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
A Grid

Exchange between $a_i$'s

Red edge is used.

Exchange between $b_i$'s

Exchange between $c_i$'s

Exchange between $d_i$'s

Exchange between $e_i$'s

Exchange between $f_i$'s

Exchange between $g_i$'s

Exchange between $h_i$'s

Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- **Red edge is used.**
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$’s
- Red edge is used.
- Exchange between $b_i$’s
- Exchange between $c_i$’s
- Exchange between $d_i$’s
- Exchange between $e_i$’s
- Exchange between $f_i$’s
- Exchange between $g_i$’s
- Exchange between $h_i$’s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$’s
- Red edge is used.
- Exchange between $b_i$’s
- Exchange between $c_i$’s
- **Exchange between $d_i$’s**
- Exchange between $e_i$’s
- Exchange between $f_i$’s
- Exchange between $g_i$’s
- Exchange between $h_i$’s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between \( a_i \)'s
- Red edge is used.
- Exchange between \( b_i \)'s
- Exchange between \( c_i \)'s
- Exchange between \( d_i \)'s
- Exchange between \( e_i \)'s
- Exchange between \( f_i \)'s
- Exchange between \( g_i \)'s
- Exchange between \( h_i \)'s
- Red edge is always used in both directions.
Idea for Proof of Lemma

- A Grid
- Exchange between $a_i$'s
- Red edge is used.
- Exchange between $b_i$'s
- Exchange between $c_i$'s
- Exchange between $d_i$'s
- Exchange between $e_i$'s
- Exchange between $f_i$'s
- Exchange between $g_i$'s
- Exchange between $h_i$'s
- Red edge is always used in both directions.
More Routing on Meshes

In the following, let $D$ denote the diameter of the network.

**Observation**

*Every permutation $\pi$ can be routed along dimension-by-dimension paths in at most $D$ steps on $M(n,1)$ and $M(n,2)$.***

**Lemma**

*Consider $M(n,3)$. There is a permutation $\pi$ such that every packet routing algorithm using dimension-by-dimension paths needs at least $\Omega(D^2)$ steps for routing $\pi$.***

**Question:**

Can one achieve time complexity $O(D)$ on meshes of dimension $d > 2$?

**Idea:** Translate the routing algorithm for permutation networks into an efficient algorithm for mesh networks.
More Routing on Meshes

In the following, let $D$ denote the diameter of the network.

**Observation**

*Every permutation $\pi$ can be routed along dimension-by-dimension paths in at most $D$ steps on $M(n,1)$ and $M(n,2)$.*

**Lemma**

*Consider $M(n,3)$. There is a permutation $\pi$ such that every packet routing algorithm using dimension-by-dimension paths needs at least $\Omega(D^2)$ steps for routing $\pi$.*

**Question:**

Can one achieve time complexity $O(D)$ on meshes of dimension $d > 2$?

**Idea:** Translate the routing algorithm for permutation networks into an efficient algorithm for mesh networks.
Notation

**Notation (d-dimensional mesh of side length n)**

Let $n \geq 1$ and $d \geq 0$ be integers. The $d$-dimensional mesh of side length $n$, denoted $M(n, d)$, is the graph $G(\{0, 1, \ldots, n - 1\}^d, E)$ with

$$E = \{ \{a, b\} \mid \exists i \in \{0, 1, \ldots, d - 1\} : |a_i - b_i| = 1 \text{ and } a_j = b_j, \text{ for } j \neq i \}.$$

- $M(n, d)$ has $n^d$ nodes and $d \cdot n^d - d \cdot n^{d-1}$ edges.
- The diameter of a $M(n, d)$-mesh is $d \cdot (n - 1)$.
- For fixed numbers $i \in \{0, 1, \ldots, d - 1\}, \ell \in \{0, 1, \ldots, n - 1\}$, the subgraph $M(n, d)\vert_{a \in \{0, 1, \ldots, n - 1\}^d \mid a_i = \ell}$ is isomorphic to $M(n, d - 1)$.
- For a fixed vector $b \in \{0, 1, \ldots, n - 1\}^{d-1}$, the subgraph $M(n, d)\vert_{a \in \{0, 1, \ldots, n - 1\}^d \mid a = ib}$ is isomorphic to $M(n, 1)$. 
Notation (d-dimensional mesh of side length n)

Let $n \geq 1$ and $d \geq 0$ be integers. The $d$-dimensional mesh of side length $n$, denoted $M(n, d)$, is the graph $G(\{0, 1, \ldots, n-1\}^d, E)$ with

$$E = \{ \{a, b\} \mid \exists i \in \{0, 1, \ldots, d-1\} : |a_i - b_i| = 1 \text{ and } a_j = b_j, \text{ for } j \neq i \}.$$ 

- $M(n, d)$ has $n^d$ nodes and $d \cdot n^d - d \cdot n^{d-1}$ edges.
- The diameter of a $M(n, d)$-mesh is $d \cdot (n - 1)$.
- For fixed numbers $i \in \{0, 1, \ldots, d-1\}$, $\ell \in \{0, 1, \ldots, n-1\}$, the subgraph $M(n, d)_{\{a \in \{0, 1, \ldots, n-1\}^d \mid a_i = \ell\}}$ is isomorphic to $M(n, d-1)$.
- For a fixed vector $b \in \{0, 1, \ldots, n-1\}^{d-1}$, the subgraph $M(n, d)_{\{a \in \{0, 1, \ldots, n-1\}^d \mid a = ib\}}$ is isomorphic to $M(n, 1)$. 
Notation

Notation \((d\text{-dimensional mesh of side length } n)\)

Let \(n \geq 1\) and \(d \geq 0\) be integers. The \(d\text{-dimensional mesh of side length } n\), denoted \(M(n, d)\), is the graph \(G(\{0, 1, \ldots, n - 1\}^d, E)\) with

\[
E = \{ \{a, b\} \mid \exists i \in \{0, 1, \ldots, d - 1\} : |a_i - b_i| = 1 \text{ and } a_j = b_j, \text{ for } j \neq i \}\.
\]

- \(M(n, d)\) has \(n^d\) nodes and \(d \cdot n^d - d \cdot n^{d-1}\) edges.
- The diameter of a \(M(n, d)\)-mesh is \(d \cdot (n - 1)\).
- For fixed numbers \(i \in \{0, 1, \ldots, d - 1\}, \ell \in \{0, 1, \ldots, n - 1\}\), the subgraph \(M(n, d)_{\{a \in \{0,1,\ldots,n-1\}^d \mid a_i = \ell\}}\) is isomorphic to \(M(n, d - 1)\).
- For a fixed vector \(b \in \{0, 1, \ldots, n - 1\}^{d-1}\), the subgraph \(M(n, d)_{\{a \in \{0,1,\ldots,n-1\}^d \mid a = ib\}}\) is isomorphic to \(M(n, 1)\).
Notation (\(d\)-dimensional mesh of side length \(n\))

Let \(n \geq 1\) and \(d \geq 0\) be integers. The \(d\)-dimensional mesh of side length \(n\), denoted \(M(n, d)\), is the graph \(G(\{0, 1, \ldots, n - 1\}^d, E)\) with

\[
E = \{ \{a, b\} \mid \exists i \in \{0, 1, \ldots, d - 1\} : |a_i - b_i| = 1 \text{ and } a_j = b_j, \text{ for } j \neq i \}.
\]

- \(M(n, d)\) has \(n^d\) nodes and \(d \cdot n^d - d \cdot n^{d-1}\) edges.
- The diameter of a \(M(n, d)\)-mesh is \(d \cdot (n - 1)\).
- For fixed numbers \(i \in \{0, 1, \ldots, d - 1\}, \ell \in \{0, 1, \ldots, n - 1\}\), the subgraph \(M(n, d)|_{\{a \in \{0, 1, \ldots, n - 1\}^d | a_i = \ell\}}\) is isomorphic to \(M(n, d - 1)\).
- For a fixed vector \(b \in \{0, 1, \ldots, n - 1\}^{d-1}\), the subgraph \(M(n, d)|_{\{a \in \{0, 1, \ldots, n - 1\}^d | a = ib\}}\) is isomorphic to \(M(n, 1)\).
Example of the Decomposition

Illustration of the decomposition of $M(n, d)$ into:

$n$ submeshes $M_0, ... , M_{n-1}$ and
Example of the Decomposition

Illustration of the decomposition of $M(n, d)$ into:

$n$ submeshes $M_0, ..., M_{n-1}$ and one of the columns $A_b$: 

![Diagram of a mesh network divided into submeshes](image-url)
Example of the Decomposition

Illustration of the decomposition of $M(n, d)$ into:

$n$ submeshes $M_0, \ldots, M_{n-1}$ and one of the columns $A_b$: 
Theorem (Annexstein and Baumslag 1990)

$M(n, d)$ can route a permutation in time $O(n \cdot d) = O(D)$.

Proof We 'simulate' the $(n, d)$-PN on $M(n, d)$.

- Decompose $M(n, d)$ into $n$ submeshes $M_0, M_1, ..., M_{n-1}$ of dimension $d - 1$ by fixing the last digit of the label, that is, for $i \in \{0, 1, ..., n - 1\}$,

  $$M_i := M(n, d)_{\{a \in \{0, 1, ..., n-1\}^d | a_{d-1} = i\}}.$$

- Each of these submeshes $M_i$ "plays the role" of a sub-PN $B^{(i)}$.

- These submeshes are connected by one-dimensional meshes (columns), one for each $d - 1$-dimensional vector $b \in \{0, 1, ..., n - 1\}^{d-1}$, namely

  $$A_b := M(n, d)_{\{a | a = ib\}}.$$
Theorem (Annexstein and Baumslag 1990)

\( M(n, d) \) can route a permutation in time \( O(n \cdot d) = O(D) \).

**Proof** We 'simulate' the \((n, d)\)-PN on \( M(n, d) \).

- Decompose \( M(n, d) \) into \( n \) submeshes \( M_0, M_1, \ldots, M_{n-1} \) of dimension \( d - 1 \) by fixing the last digit of the label, that is, for \( i \in \{0, 1, \ldots, n-1\} \),

\[
M_i := M(n, d)|_{\{a \in \{0,1,\ldots,n-1\}^d | a_{d-1} = i\}}.
\]

- Each of these submeshes \( M_i \) "plays the role" of a sub-PN \( B^{(i)} \).
- These submeshes are connected by one-dimensional meshes (columns), one for each \( d - 1 \)-dimensional vector \( b \in \{0, 1, \ldots, n - 1\}^{d-1} \), namely

\[
A_b := M(n, d)|_{\{a | a = ib\}}.
\]
Annexstein and Baumslag

Theorem (Annexstein and Baumslag 1990)

\( M(n, d) \) can route a permutation in time \( O(n \cdot d) = O(D) \).

Proof We 'simulate' the \((n, d)\)-PN on \( M(n, d) \).

- Decompose \( M(n, d) \) into \( n \) submeshes \( M_0, M_1, ..., M_{n-1} \) of dimension \( d - 1 \) by fixing the last digit of the label, that is, for \( i \in \{0, 1, ..., n - 1\} \),

\[
M_i := M(n, d)_{\{a \in \{0, 1, ..., n-1\}^d | a_{d-1} = i\}}.
\]

- Each of these submeshes \( M_i \) "plays the role" of a sub-PN \( B^{(i)} \).

- These submeshes are connected by one-dimensional meshes (columns), one for each \( d - 1 \)-dimensional vector \( b \in \{0, 1, ..., n - 1\}^{d-1} \), namely

\[
A_b := M(n, d)_{\{a | a = ib\}}.
\]
Theorem (Annexstein and Baumslag 1990)

\[ M(n, d) \text{ can route a permutation in time } O(n \cdot d) = O(D). \]

Proof We 'simulate' the \((n, d)\)-PN on \(M(n, d)\).

- Decompose \(M(n, d)\) into \(n\) submeshes \(M_0, M_1, \ldots, M_{n-1}\) of dimension \(d - 1\) by fixing the last digit of the label, that is, for \(i \in \{0, 1, \ldots, n - 1\}\),

\[ M_i := M(n, d)_{\{a \in \{0, 1, \ldots, n-1\}^d | a_{d-1} = i\}}. \]

- Each of these submeshes \(M_i\) “plays the role” of a sub-PN \(B^{(i)}\).

- These submeshes are connected by one-dimensional meshes (columns), one for each \(d - 1\)-dimensional vector \(b \in \{0, 1, \ldots, n - 1\}^{d-1}\), namely

\[ A_b := M(n, d)_{\{a | a = ib\}}. \]
Theorem (Annexstein and Baumslag 1990)

\( M(n, d) \) can route a permutation in time \( O(n \cdot d) = O(D) \).

**Proof** We 'simulate' the \((n, d)\)-PN on \( M(n, d) \).

- Decompose \( M(n, d) \) into \( n \) submeshes \( M_0, M_1, \ldots, M_{n-1} \) of dimension \( d - 1 \) by fixing the last digit of the label, that is, for \( i \in \{0, 1, \ldots, n - 1\} \),

  \[
  M_i := M(n, d) | \{a \in \{0, 1, \ldots, n - 1\}^d | a_{d - 1} = i\}.
  \]

- Each of these submeshes \( M_i \) “plays the role” of a sub-PN \( B^{(i)} \).

- These submeshes are connected by one-dimensional meshes (columns), one for each \( d - 1 \)-dimensional vector \( b \in \{0, 1, \ldots, n - 1\}^{d-1} \), namely

  \[
  A_b := M(n, d) | \{a | a = ib\}.
  \]
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with $n$ colors. The following algorithm then performs the routing:

- Packets with color $i$ route from their sources to submesh $M_i$ (inside the corresponding column $A_b$)
- In each submesh $M_i$: Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in $M_i$ is solved recursively.)
- Packets route from submesh $M_i$ to their destinations (inside the corresponding column $A_b$)

Analysis of the time complexity:

- Let $T(n, d)$ be the routing time for $M(n,d)$.

  - $d = 1$ : $T(n, 1) = n - 1$
  - $d > 1$ : $T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)$
- Solving the recurrence gives $T(n, d) = (2d - 1)(n - 1) \leq 2D$. 
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with $n$ colors. The following algorithm then performs the routing:

- Packets with color $i$ route from their sources to submesh $M_i$ (inside the corresponding column $A_b$)

- In each submesh $M_i$: Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in $M_i$ is solved recursively.)

- Packets route from submesh $M_i$ to their destinations (inside the corresponding column $A_b$)

Analysis of the time complexity:

- Let $T(n, d)$ be the routing time for $M(n,d)$.

  $d = 1: \quad T(n, 1) = n - 1$

  $d > 1: \quad T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)$

- Solving the recurrence gives $T(n, d) = (2d - 1)(n - 1) \leq 2D$. 
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with $n$ colors. The following algorithm then performs the routing:

- Packets with color $i$ route from their sources to submesh $M_i$ (inside the corresponding column $A_b$)
- In each submesh $M_i$: Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in $M_i$ is solved recursively.)
- Packets route from submesh $M_i$ to their destinations (inside the corresponding column $A_b$)

Analysis of the time complexity:

- Let $T(n, d)$ be the routing time for $M(n,d)$.
  - $d = 1: \quad T(n, 1) = n - 1$
  - $d > 1: \quad T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)$
- Solving the recurrence gives $T(n, d) = (2d - 1)(n - 1) \leq 2D$. 
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with \( n \) colors. The following algorithm then performs the routing:

- Packets with color \( i \) route from their sources to submesh \( M_i \) (inside the corresponding column \( A_b \))

- In each submesh \( M_i \): Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in \( M_i \) is solved recursively.)

- Packets route from submesh \( M_i \) to their destinations (inside the corresponding column \( A_b \))

Analysis of the time complexity:

- Let \( T(n, d) \) be the routing time for \( M(n,d) \).
  
  \[ d = 1 : \quad T(n, 1) = n - 1 \]
  
  \[ d > 1 : \quad T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1) \]

  Solving the recurrence gives \( T(n, d) = (2d - 1)(n - 1) \leq 2D \).
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with $n$ colors. The following algorithm then performs the routing:

- Packets with color $i$ route from their sources to submesh $M_i$ (inside the corresponding column $A_b$)
- In each submesh $M_i$: Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in $M_i$ is solved recursively.)
- Packets route from submesh $M_i$ to their destinations (inside the corresponding column $A_b$)

Analysis of the time complexity:

- Let $T(n, d)$ be the routing time for $M(n,d)$.
  
- $d = 1: \ T(n, 1) = n - 1$
  
- $d > 1: \ T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)$

Solving the recurrence gives $T(n, d) = (2d - 1)(n - 1) \leq 2D$. 
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with \( n \) colors. The following algorithm then performs the routing:

- Packets with color \( i \) route from their sources to submesh \( M_i \) (inside the corresponding column \( A_b \))
- In each submesh \( M_i \): Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in \( M_i \) is solved recursively.)
- Packets route from submesh \( M_i \) to their destinations (inside the corresponding column \( A_b \))

Analysis of the time complexity:

- Let \( T(n, d) \) be the routing time for \( M(n,d) \).

\[
\begin{align*}
d = 1: & \quad T(n, 1) = n - 1 \\
d > 1: & \quad T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)
\end{align*}
\]

- Solving the recurrence gives \( T(n, d) = (2d - 1)(n - 1) \leq 2D. \)
Proof

Analogous to the algorithm for permutation networks, we color the requests (packets) with $n$ colors. The following algorithm then performs the routing:

- Packets with color $i$ route from their sources to submesh $M_i$ (inside the corresponding column $A_b$)
- In each submesh $M_i$: Each packet is routed from the position isomorphic to its source to the position isomorphic to its destination. (The permutation routing problem in $M_i$ is solved recursively.)
- Packets route from submesh $M_i$ to their destinations (inside the corresponding column $A_b$)

Analysis of the time complexity:

- Let $T(n, d)$ be the routing time for $M(n,d)$.
  - $d = 1$: $T(n, 1) = n - 1$
  - $d > 1$: $T(n, d) = T(n, 1) + T(n, d - 1) + T(n, 1)$
- Solving the recurrence gives $T(n, d) = (2d - 1)(n - 1) \leq 2D$. 
Literature

Legende

- : Nicht relevant
- : Grundlagen, die implizit genutzt werden
- : Idee des Beweises oder des Vorgehens
- : Struktur des Beweises oder des Vorgehens
- : Vollständiges Wissen