

Bin Packing oder „Wie bekomme ich die Klamotten in die Kisten?“

Ich habe diesen Sommer mein Abi gemacht und möchte zum Herbst mit dem Studium beginnen – Informatik natürlich! Da es in meinem kleinen Ort keine Uni gibt, werde ich deshalb in Kürze umziehen müssen. Dann heißt es, all die tausend Sachen in meinen Schränken und Regalen in Umzugskartons zu verpacken. Um den Umzug möglichst kostengünstig zu gestalten, werde ich mich dabei bemühen, die Gegenstände in möglichst wenige Kartons zu verpacken.



Wenn ich nun alle Gegenstände einfach so der Reihe nach aus den Regalen nehmen und in einen Umzugskarton nach dem anderen verpacken würde, würde ich dabei eine ganze Menge Platz in den Kartons verschwenden, da die Gegenstände verschieden groß und unterschiedlich geformt sind, und sich dadurch eine Menge Lücken in dem Umzugskartons ergeben würden.

Die optimale Lösung für dieses Problem, d.h. die kleinstmögliche Anzahl benötigter Umzugskartons, würde ich sicherlich finden, wenn ich alle Möglichkeiten, die Gegenstände in Umzugskartons zu verpacken, der Reihe nach ausprobiere. Doch das würde bei so vielen Gegenständen ein halbe Ewigkeit dauern und obendrein ein riesiges Chaos in der Wohnung verursachen.

Deshalb würde ich die Gegenstände schon am liebsten in der Reihenfolge in Umzugskartons verpacken, in der sie mir beim Ausräumen der Schränke und Regale zufällig gerade in die Hand kommen. Die entscheidende Frage ist daher, wie viele Umzugskartons ich bei dieser Vorgehensweise mehr benötige als bei der optimalen Lösung. Um dies herauszufinden, werde ich das Problem nun analysieren.

Das Online-Problem „billig umziehen“

Da ich die Gegenstände der Reihe nach aus den Schränken bzw. Regalen nehmen und verpacken möchte, bedeutet das, dass ich es mit einem *Online*-Problem (vgl. [Einführung in Online Algorithmen](#)) zu tun habe, denn:

- Die relevanten Daten (hier: die Größe der einzelnen Gegenstände) treffen erst nach und nach im Laufe der Zeit ein. Die Liste dieser Größen, in der Reihenfolge ihres Auftretens, bezeichnen wir im Folgenden mit σ .
- Es liegen keine Informationen über die zukünftigen Daten (hier: die Größen der noch nicht betrachteten Gegenstände) vor.

- Die Anzahl der zu bearbeitenden Daten (hier: der zu verpackenden Gegenstände) ist nicht im Voraus bekannt.
- Die aktuelle Anfrage muss sofort bearbeitet werden (hier: Gegenstände werden nicht vorübergehend zur Seite gelegt).

In der Realität liegen für einige Aspekte zwar Schätzwerte vor, da ich in der Wohnung schließlich jahrelang gelebt habe und somit eine gewisse Vorstellung davon habe, was sich alles in den Regalen und Schränken befindet; dies werde ich hier jedoch außer Acht lassen und das Problem somit idealisiert betrachten.

Meine Verpack-Strategie sieht nun als Online-Algorithmus folgendermaßen aus: Die Eingabe besteht aus einer Daten-Sequenz σ mit den Größen G_i der zu verpackenden Gegenstände. Die Umzugskartons werden mit K_i bezeichnet, und die Ausgabe besteht aus der Anzahl n der benötigten Umzugskartons.

Algorithmus NextFit:

1. Setze $n := 1$.
2. Für jedes G_i aus σ tue folgendes:
 3. Falls G_i in Karton K_n keinen Platz mehr hat,
 4. schließe Karton K_n und
 5. setze $n := n + 1$.
6. Packe G_i in Karton K_n .

Mit etwas mehr Aufwand kann man auch folgendermaßen vorgehen: ich könnte die Umzugskartons erst zum Schluss endgültig schließen und beim Verpacken eines jeden Gegenstands alle angefangenen Umzugskartons darauf überprüfen, ob vielleicht noch genügend Platz übrig ist. Dies hätte dann einen Vorteil, wenn ich für einen besonders großen Gegenstand einen neuen Umzugskarton anfangen muss, und danach wieder eine Reihe kleinerer Gegenstände zu verpacken sind, denn diese kleineren Gegenstände können eventuell noch in einem früheren Umzugskarton Platz finden.

Als Online-Algorithmus sieht diese zweite Strategie folgendermaßen aus:

Algorithmus FirstFit:

1. Setze $n := 1$.
2. Für jedes G_i aus σ tue folgendes:
 3. Für $j := 1, \dots, n$ tue folgendes:
 4. Falls G_i in Karton K_j noch Platz hat,
 5. packe G_i in Karton K_j und
 6. fahre mit dem nächsten Gegenstand fort (gehe zu Schritt 2).
 7. Setze $n := n + 1$ und
 8. Packe G_i in Karton K_n .

Analyse der Algorithmen

Um die Analyse, wie gut oder schlecht meine Strategien funktionieren, etwas zu vereinfachen, treffe ich folgende Annahme: Ein Gegenstand passt genau dann in einen Umzugskarton, wenn das noch unbenutzte Volumen des Umzugskartons größer oder gleich dem Volumen des zu verpackenden Gegenstands ist (ich vernachlässige also den aufgrund von „Verschnitt“ nicht nutzbaren Platz in den Umzugskartons). Weiterhin wähle ich die Volumeneinheit derart, dass die Kapazität der Umzugskartons genau 1 ist (und die Größen der zu verpackenden Gegenstände somit kleiner oder

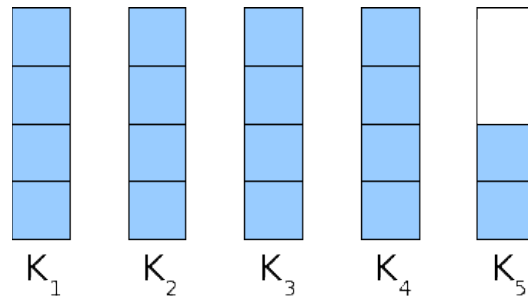
gleich 1 sind).

Um ein Gefühl für die Güte des Ergebnisses meiner Online-Algorithmen zu bekommen, schaue ich mir ein paar Beispiele an. Sind alle Gegenstände gleich groß – wie in Beispiel 1 – so liefern NextFit und FirstFit beide das optimale Ergebnis:

Beispiel 1:

$$\sigma = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \right)$$

NextFit = FirstFit: n = 5

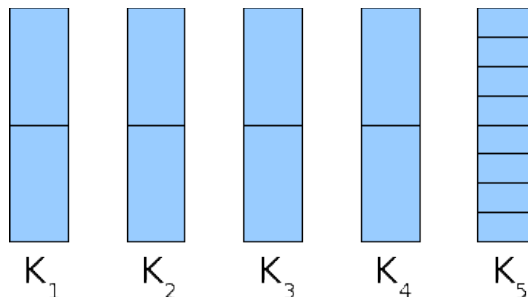


In Beispiel 2 liefern NextFit und FirstFit ebenfalls noch beide das optimale Ergebnis:

Beispiel 2:

$$\sigma = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8} \right)$$

NextFit = FirstFit: n = 5

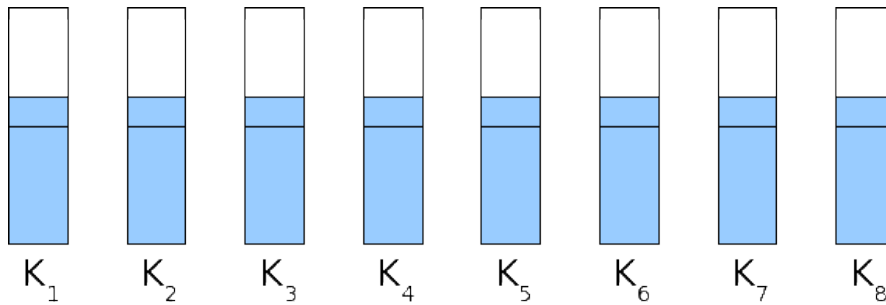


Beispiel 3 jedoch zeigt, dass die Reihenfolge der Gegenstände der Ergebnis beeinflussen kann – hier schneidet NextFit deutlich schlechter ab:

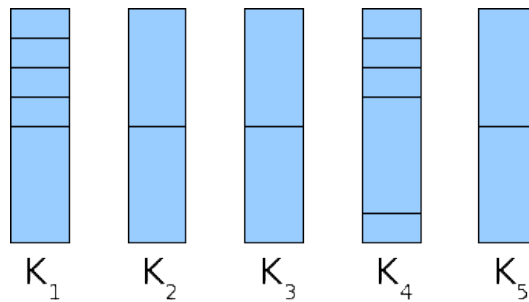
Beispiel 3:

$$\sigma = \left(\frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8} \right)$$

NextFit: $n = 8$

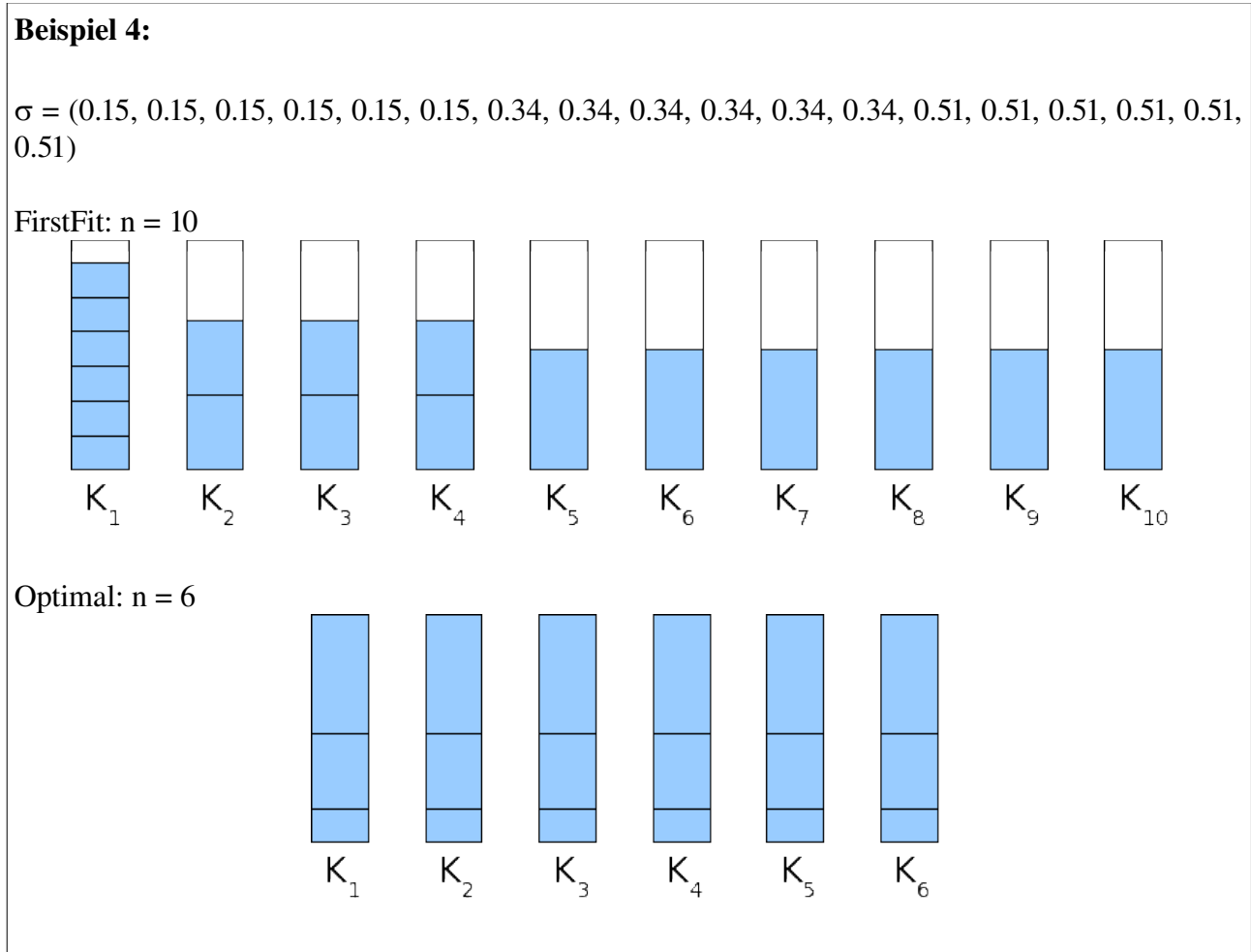


FirstFit: $n = 5$



Wählt man in Beispiel 3 anstatt der Größe $\frac{1}{8}$ eine viel kleinere Zahl, so steigt der ungenutzte Platz pro Umzugskarton in NextFit sogar auf fast die Hälfte. Die Strategie NextFit verbraucht im Extremfall also fast doppelt so viele Umzugskartons wie bei optimaler Packung nötig wären.

Die Strategie FirstFit hat in Beispiel 3 zwar noch optimal abgeschnitten, doch auch für FirstFit gibt es ungünstige Eingaben, wie Beispiel 4 zeigt:



Hier ergibt sich also ein Verhältnis von 10:6 für FirstFit gegenüber der optimalen Packung, d.h. FirstFit benötigt 1,67-mal so viele Umzugskartons. Wir nennen einen solchen Algorithmus 1,67-kompetitiv. Allgemein heißt eine Online-Strategie k -kompetitiv, wenn man nie mehr als das k -fache dessen benötigt, was man benötigt hätte, wenn man die Zukunft gekannt hätte.

Es stellt sich nun die Frage, ob die Negativbeispiele 3 und 4 bereits die schlechtestmöglichen Eingabefolgen für NextFit bzw. FirstFit darstellen, oder ob es noch schlimmer kommen kann. Um die Approximationsgüte von NextFit zu berechnen, bezeichnen wir mit k die Anzahl der Gegenstände und mit n die Anzahl der benötigten Umzugskartons. Weiterhin beschreiben wir mit $v(G_i)$ das Volumen des Gegenstands G_i und mit $v(K_i)$ das belegte Volumen im Umzugskarton K_i . Betrachtet man nun zwei nacheinander befüllte Umzugskartons K_i und K_{i+1} , $0 < i < n$, so gilt:

$$v(K_i) + v(K_{i+1}) > 1$$

Wäre diese Bedingung nicht erfüllt, hätten sämtliche Gegenstände in K_{i+1} noch in K_i Platz gehabt und wären somit nach Definition von NextFit nicht in K_{i+1} gelegt worden. Addiert man nun das belegte Volumen von K_1 und K_2 , von K_3 und K_4 , von K_5 und K_6 , usw., so ist dies jeweils echt größer als 1 und man erhält somit für die Summe über alle Umzugskartons:

$$\sum_{i=1}^n v(K_i) > \lfloor \frac{n}{2} \rfloor$$

Für den Fall, dass n ungerade ist, runden wir den Bruch auf der rechten Seite der Ungleichung ab. Nun gilt, dass das Gesamtvolumen aller Gegenstände gleich dem insgesamt in allen Umzugskartons belegtem Volumen ist, d.h.

$$\sum_{i=1}^k v(G_i) = \sum_{i=1}^n v(K_i)$$

Also benötigt aufgrund des Gesamtvolumens der Gegenstände selbst die bestmögliche Packung mindestens

$$\lceil \sum_{i=1}^k v(G_i) \rceil \geq \lceil \frac{n}{2} \rceil$$

Umzugskartons. Das Gesamtvolumen der Gegenstände muss hierbei aufgerundet werden, da die Anzahl der Umzugskartons ganzzahlig sein muss. Für das Verhältnis der von NextFit benötigten Umzugskartons zur Anzahl der bei der optimalen Packung benötigten Umzugskartons folgt somit:

$$n \div \lceil \frac{n}{2} \rceil \leq 2$$

Der Online-Algorithmus NextFit ist also 2-kompetitiv (vgl. [Einführung in Online Algorithmen](#)). Da dieser Beweis auf FirstFit übertragbar ist, ist somit auch FirstFit 2-kompetitiv. Mit einem deutlich komplizierteren Beweis (den wir hier nicht vorführen, vgl. weiterführende Materialien) kann man sogar zeigen, dass FirstFit 1,7-kompetitiv ist.

Wie gut kann ein Online-Algorithmus für Bin Packing sein?

Wir kennen nun die Approximationsgüte von NextFit bzw. FirstFit und wissen, dass es Eingabefolgen gibt, für die das Ergebnis fast einen Faktor 2 bzw. 1,7 vom theoretischen Optimum entfernt ist. Auf der einen Seite ist dies ein gutes Ergebnis, da wir wissen, dass der verschwendete Platz in den Umzugskartons niemals einen bestimmten Faktor überschreitet, andererseits ist es das Ergebnis aber auch etwas unbefriedigend, denn es macht schon einen schmerzlichen Unterschied, ob der Umzug nun 1000 EUR oder 2000 EUR (bzw. 1700 EUR) kostet.

Um nun abschließend zu beurteilen, wie gut oder schlecht die Verpackungsstrategien tatsächlich sind, gilt es noch zu untersuchen, wie gut ein Online-Algorithmus für dieses Problem überhaupt sein kann. Denn da die Eingabefolge nicht im Voraus bekannt ist, dürfte es wohl unmöglich sein, einen Online-Algorithmus zu entwerfen, der immer ein optimales Ergebnis liefert. Dies werden wir nun beweisen.

Angenommen, wir haben eine Eingabefolge, die $2 \cdot x$ Gegenstände der Größe $\frac{1}{2} - \varepsilon$ enthält, wobei x eine positive Ganzzahl ist und ε eine beliebig kleine, positive Dezimalzahl ist. Die optimale Packung für diese Eingabefolge sind offensichtlich x Umzugskartons, die jeweils mit 2 Gegenständen gefüllt sind. Wir betrachten nun einen beliebigen Online-Algorithmus und bezeichnen diesen mit BinPac. BinPac wird die $2 \cdot x$ Gegenstände der Eingabefolge – je nach Strategie – so auf die Umzugskartons verteilen, dass in jedem Umzugskarton entweder ein oder zwei Gegenstände landen. Mit b_1 bezeichnen wir die Anzahl an Umzugskartons, in denen ein Gegenstand liegt, und mit b_2 die Anzahl der Umzugskartons, in denen zwei Gegenstände liegen. Mit $b = b_1 + b_2$ bezeichnen wir die Anzahl der insgesamt von BinPac benötigten Umzugskartons. Man stelle nun fest, dass folgender Zusammenhang gilt:

$$\begin{aligned} b_1 + 2 \cdot b_2 &= 2 \cdot x \\ \Rightarrow b_1 &= 2 \cdot x - 2 \cdot b_2 \end{aligned}$$

Setzt man dies in $b = b_1 + b_2$ ein, so erhält man:

$$b = (2 \cdot x - 2 \cdot b_2) + b_2 = 2 \cdot x - b_2 \quad (1)$$

Auf dieses Zwischenergebnis werden wir später wieder zurückkommen. Nun überlegen wir uns, was passiert, wenn unsere Eingabefolge $4 \cdot x$ Gegenstände enthält, wobei die ersten $2 \cdot x$ Gegenstände wieder die Größe $\frac{1}{2} - \varepsilon$ haben und die restlichen $2 \cdot x$ Gegenstände die Größe $\frac{1}{2} + \varepsilon$. Da Online-Algorithmen nicht in die Zukunft schauen können, wird sich BinPac für die ersten $2 \cdot x$ kleineren Gegenstände genauso verhalten wie vorhin, als keine weiteren Gegenstände folgten. Die Gegenstände der Größe $\frac{1}{2} + \varepsilon$ können nun zunächst auf die b_1 Umzugskartons, in denen noch Platz ist, verteilt werden, und für die restlichen $2 \cdot x - b_1$ Gegenstände muss jeweils ein neuer Umzugskartons angefangen werden. Insgesamt benötigt BinPac bei dieser Eingabefolge also

$$b + (2 \cdot x - b_1) = (b_1 + b_2) + (2 \cdot x - b_1) = 2 \cdot x + b_2 \quad (2)$$

Umzugskartons. Die optimale Lösung wäre aber gewesen, in jeden Umzugskarton jeweils einen der kleineren und einen der größeren Gegenstände zu legen, so dass man insgesamt nur $2 \cdot x$ Umzugskartons benötigt hätte.

Mit diesen Vorüberlegungen können wir nun beweisen, dass kein Online-Algorithmus besser als $4/3$ -kompetitiv sein kann. Wir führen diesen Beweis, indem wir zunächst annehmen, dass es sehr wohl einen besseren Online-Algorithmus gäbe, und führen diese Annahme dann zum Widerspruch. Angenommen, BinPac wäre besser als $4/3$ -kompetitiv, dann müsste die Anzahl der benötigten Umzugskartons bei der ersten betrachteten Eingabefolge echt kleiner sein $4/3$ der bei einer optimalen Lösung benötigten Umzugskartons – formal:

$$b < \frac{4}{3} \cdot x$$

Wendet man diese Bedingung auf Gleichung (1) an, so erhält man:

$$\begin{aligned} 2 \cdot x - b_2 &< \frac{4}{3} \cdot x \\ \Rightarrow b_2 &> \frac{2}{3} \cdot x \quad (3) \end{aligned}$$

Analog müsste für die zweite Eingabefolge gelten, dass die Anzahl der benötigten Umzugskartons (vgl. Gleichung (2)) echt weniger ist als $4/3$ der optimalen Lösung ($2 \cdot x$) – formal:

$$\begin{aligned} 2 \cdot x + b_2 &< \frac{4}{3} (2 \cdot x) \\ \Rightarrow b_2 &< \frac{2}{3} \cdot x \quad (4) \end{aligned}$$

Damit haben wir einen Widerspruch, denn laut Gleichungen (3) und (4) müsste b_2 sowohl echt kleiner als auch echt größer als $2/3 \cdot x$ sein, was unmöglich ist. Folglich muss die Annahme falsch gewesen sein, und wir haben bewiesen:

Satz:

Es gibt keinen α -kompetitiven Online-Algorithmus für das Bin Packing Problem mit $\alpha < 4/3$.

Unter dem Aspekt, dass selbst die bestmögliche Online-Strategie für das Umzugsproblem nicht besser als $4/3$ -kompetitiv sein kann, erscheint nun die $1,7$ -kompetitive Strategie FirstFit in einem ganz anderen Licht. Na dann kanns mit dem Packen ja losgehen!

Eine weitere Einsatzmöglichkeiten des Bin Packing ist beispielsweise, Dateien auf CDs zu verteilen (etwa im Rahmen einer Datensicherung). In diesem Fall lassen sich die oben beschriebenen Strategien sogar direkt anwenden, d.h. man muss keine vereinfachende Annahmen bzgl. des „Verschnitts“ treffen, da dieses Problem bei Datenströmen nicht auftritt.

Autoren:

- [Prof. Dr. Friedhelm Meyer auf der Heide](#)
- [Joachim Gehweiler](#)

Externe Links:

- [FirstFit Algorithmus als Animation \(Java-Applet\)](#)