

33. Algorithmus der Woche Mehrheitsbestimmung Wer wird Klassensprecher?

Autor

Thomas Erlebach, University of Leicester

Heiko sitzt vor einem Stapel von Zetteln. In seiner Klasse wurde gerade eine Wahl zur Bestimmung des Klassensprechers (oder der Klassensprecherin) durchgeführt. Alle Schüler haben den Namen ihres Wunschkandidaten auf einen Zettel geschrieben, und Heiko hat sich freiwillig gemeldet, das Wahlergebnis zu bestimmen. Zuvor haben sich alle darauf geeinigt, dass ein Kandidat nur dann Klassensprecher werden soll, wenn mehr als die Hälfte der Schüler für ihn gestimmt haben. Falls niemand die absolute Mehrheit der Stimmen bekommen hat, soll die Wahl wiederholt werden. Heikos Aufgabe ist es jetzt also herauszufinden, ob irjemand mehr als die Hälfte aller Stimmen bekommen hat.



Wie soll Heiko diese Aufgabe erledigen? Er macht sich keine grossen Gedanken und fängt einfach mit der naheliegendsten Methode an. Auf ein Blatt Papier will er alle Namen schreiben, für die gestimmt wurde, und zu jedem Namen eine Strichliste machen, die angibt, wie oft für diese Person gestimmt wurde. Er nimmt einen Wahlzettel nach dem anderen und schaut nach, welcher Name darauf steht. Wenn er den Namen noch nicht auf seinem Blatt stehen hat, so schreibt er ihn auf das Blatt und macht einen Strich daneben. Wenn der Name dagegen bereits auf dem Blatt steht, so macht er bei dem Namen nur einen zusätzlichen Strich. Als Heiko mit allen Wahlzetteln fertig ist, ergibt sich folgendes Bild:

Tobias	+++
Anton	
Heinz	
Corinna	
Kevin	
Monika	+++ +++ IIII
Laura	

Nun sucht Heiko denjenigen Namen, der am meisten Striche hat, und zählt diese Striche. Wenn die Anzahl der Striche mehr als die Hälfte der Anzahl Schüler in seiner Klasse ist, so hat dieser Kandidat die Wahl

gewonnen. Andernfalls hat niemand eine absolute Mehrheit und es muss eine neue Wahl stattfinden. Heiko sieht, dass Monika die meisten Stimmen bekommen hat, nämlich 14. In Heikos Klasse sind insgesamt 27 Schüler. Da 14 mehr als 13.5 (die Hälfte von 27) ist, hat Monika die absolute Mehrheit bekommen und ist somit zur Klassensprecherin gewählt. Monika war bereits letztes Jahr Klassensprecherin und wird die Aufgabe sicher auch weiterhin gut erledigen. Alle gratulieren Monika zum Gewinn der Wahl.

Später lässt sich Heiko das Auszählen der Wahlzettel noch einmal durch den Kopf gehen und überlegt, wie gut das von ihm verwendete Verfahren denn war. Er musste für jeden Wahlzettel die Liste aller Namen auf seinem Blatt durchgehen und dann entweder einen Strich machen oder den Namen neu in seine Liste aufnehmen. Bei 27 Wahlzetteln war das kein Problem, aber bei einer grösseren Wahl würde das wohl ganz schön viel Arbeit machen. Man stelle sich nur eine Wahl mit Hunderten oder Tausenden von Wahlzetteln vor. Da könnte die Liste der Namen sehr, sehr lang werden. Wenn man einen Wahlzettel bearbeitet, würde es dementsprechend lange dauern herauszufinden, ob der Name bereits in der Liste ist oder nicht. Außerdem hat Heiko mit seiner Methode viel mehr Informationen produziert als verlangt: Er hat nicht nur die Wahlsiegerin bestimmt, sondern auch noch für alle Kandidaten gezählt, wie viele Stimmen sie erhalten haben; letztere Information wäre eigentlich gar nicht nötig gewesen, um seine Aufgabe zu lösen. Vielleicht hätte man die Berechnung unnötiger Informationen vermeiden und die Aufgabe dann mit weniger Aufwand lösen können. (An dieser Stelle wäre noch anzumerken, dass es allgemein auch aus Datenschutzgründen oft sehr wichtig ist, das unnötige Erheben und Auswerten von Daten zu vermeiden und nur die Informationen zu generieren, die zur Erfüllung der gestellten Aufgabe unbedingt notwendig sind; auf diese Weise wird die Gefahr des Missbrauchs persönlicher Daten deutlich eingeschränkt. Wir können hier aber nicht weiter auf diese Datenschutzproblematik eingehen.)

Heiko hat sich in letzter Zeit etwas mit effizienten Algorithmen beschäftigt und weiß inzwischen, dass es in vielen Fällen geschickte Verfahren gibt, die deutlich schneller als die naheliegendste Methode sind. Er beschließt daher, der Sache auf den Grund zu gehen und nachzuforschen, ob es eine effizientere Methode zur Bestimmung der Mehrheit gibt. Zusammen mit Laura, die sich ebenfalls für Algorithmen interessiert, schlägt Heiko in mehreren Büchern über Algorithmen nach, ob etwas zu diesem Problem zu finden ist.

Majority-Algorithmus

Laura und Heiko finden heraus, dass das Problem, unter N Elementen das Mehrheitselement (d.h. ein Element, das mehr als $N/2$ mal vorkommt) zu bestimmen, unter dem Namen Majority (Majority ist das englische Wort für Mehrheit) behandelt wird. In einem Buch wird der folgende Algorithmus beschrieben.

MAJORITY-ALGORITHMUS

- Verwende einen Stapel von Elementen, der anfangs leer ist.
- Phase 1: Bearbeite nacheinander jedes der N gegebenen Elemente und führe dabei für jedes Element X das folgende aus:
 - Falls der Stapel leer ist, so lege X oben auf den Stapel.
 - Andernfalls vergleiche X mit dem obersten Element des Stapels. Falls X und dieses Element gleich sind, so lege X oben auf den Stapel; falls die beiden Elemente nicht gleich sind, so entferne das oberste Element des Stapels.
- Falls der Stapel leer ist, so melde, dass es kein Mehrheitselement gibt.
- Phase 2: Andernfalls nimm das oberste Element Y des Stapels und zähle, wie oft das Element unter allen N gegebenen Elementen vorkommt. Wenn Y mehr als $N/2$ mal vorkommt, so gib Y als Antwort aus. Wenn Y nicht mehr als $N/2$ mal vorkommt, so melde, dass es kein Mehrheitselement gibt.

Laura und Heiko sind recht verbüfft, dass das funktionieren soll. Der Algorithmus würde das Auszählen einer grossen Wahl in der Tat einfacher machen: Bei jedem Wahlzettel müsste man den Namen nur mit einem einzigen Namen vergleichen, nämlich dem Namen auf dem Wahlzettel, der oben auf dem Stapel

liegt. Am Ende müsste man zwar noch ein zweites Mal alle Wahlzettel durchgehen und zählen, wie oft der in Phase 1 bestimmte Name insgesamt vorkommt, aber das wäre auch keine große Sache. Der Algorithmus klingt also interessant, aber Heiko und Laura bezweifeln noch, dass das wirklich funktioniert. Es sieht so aus, als ob am Ende von Phase 1 irgendein Element oben auf dem Stapel liegen könnte, das unter den letzten Elementen zufällig ein paar mal vorkommt, obwohl vielleicht ein ganz anderes Element das Mehrheitselement ist. Sie sind also skeptisch und probieren daher erst einmal aus, wie der Algorithmus mit ein paar Beispielingaben umgeht. Nehmen wir zum Beispiel als Eingabe die folgenden $N=7$ Elemente (der Einfachheit halber nehmen wir Buchstaben als Elemente):

B,B,A,A,C,A,A

In Phase 1 läuft der Algorithmus wie folgt ab (jede Zeile der folgenden Tabelle entspricht einem Schritt des Algorithmus):

Stapel (von unten nach oben)	Betrachtetes Element	Aktion
leer	B	B auf den Stapel legen
B	B	B auf den Stapel legen
B,B	A	B vom Stapel entfernen
B	A	B vom Stapel entfernen
leer	C	C auf den Stapel legen
C	A	C vom Stapel entfernen
leer	A	A auf den Stapel legen
A		

Tatsächlich, am Ende liegt ein A auf dem Stapel. Der Algorithmus wird nun in Phase 2 zählen, wie oft A unter den gegebenen Elementen vorkommt. Da A unter den $N=7$ Elementen B,B,A,A,C,A,A viermal vorkommt, wird A als Mehrheitselement erkannt und ausgegeben. Auf diesem Beispiel hat der Algorithmus also richtig funktioniert. Man sieht auch, dass zu jeder Zeit alle Elemente auf dem Stapel gleich sind; das ist ja eigentlich klar, da der Algorithmus nur dann ein Element auf den Stapel legt, wenn der Stapel leer ist oder wenn das Element gleich dem obersten Element auf dem Stapel ist.

Was macht der Algorithmus, wenn die Eingabe kein Mehrheitselement enthält, also z.B. bei der Eingabe A,B,C,C? Hier kommt C zwar unter $N=4$ Elementen zweimal vor, aber ein Mehrheitselement müsste ja echt mehr als $N/2$ mal vorkommen. Der Algorithmus würde in Phase 1 wie folgt ablaufen:

Stapel (von unten nach oben)	Betrachtetes Element	Aktion
leer	A	A auf den Stapel legen
A	B	ein A vom Stapel entfernen
leer	C	C auf den Stapel legen
C	C	C auf den Stapel legen
C,C		

Hier ist am Ende von Phase 1 also ein C oben auf dem Stapel. Man sieht somit, dass Phase 2 des Algorithmus wirklich nötig ist. In Phase 2 zählt der Algorithmus nämlich, wie oft das C in der Eingabe vorkommt. Da es nur zweimal vorkommt, erkennt der Algorithmus richtig, dass die Eingabe kein Mehrheitselement enthält.

Wie wäre es mit einer Eingabe der Form A,A,A,B,B, bei der am Ende ein anderes Element als das Mehrheitselement oft vorkommt? Auch hier läßt sich der Algorithmus nicht täuschen:

Stapel (von unten nach oben)	Betrachtetes Element	Aktion
leer	A	A auf den Stapel legen
A	A	A auf den Stapel legen
A,A	A	A auf den Stapel legen
A,A,A	B	A vom Stapel entfernen
A,A	B	A vom Stapel entfernen
A		

Am Ende liegt ein A oben auf dem Stapel, also findet der Algorithmus auch hier wieder das korrekte Mehrheitselement. Irgendwie scheint der Algorithmus also vielleicht doch richtig zu sein, obwohl Laura und Heiko immer noch nicht richtig verstehen, wieso er eigentlich funktioniert. Die beiden schauen noch einmal in das Algorithmenbuch. Dort finden sie einen Beweis für die Korrektheit des Algorithmus. Zuerst scheint der Beweis recht kompliziert und schwer nachzuvollziehen, doch Laura und Heiko arbeiten ihn ein paar mal durch und helfen einander, indem sie sich gegenseitig die Teile des Beweises erklären, die sie jeweils schon verstanden haben. Am Ende verstehen sie, warum der Algorithmus tatsächlich immer funktioniert.

Korrektheit des Majority-Algorithmus

Der Korrektheitsbeweis für den Majority-Algorithmus lässt sich wie folgt zusammenfassen. Wenn der Algorithmus ein Mehrheitselement X ausgibt, so muss dieses Element tatsächlich das richtige Mehrheitselement sein, da der Algorithmus in Phase 2 ja bestätigt hat, dass X mehr als $N/2$ mal vorkommt. Somit könnte der Algorithmus also nur dann ein falsches Ergebnis liefern, wenn die Eingabe ein Mehrheitselement X enthält, am Ende von Phase 1 aber entweder der Stapel leer ist oder ein anderes Element als X oben auf dem Stapel liegt (und der Algorithmus dann ausgeben würde, dass es kein Mehrheitselement gibt). Die folgenden Überlegungen zeigen, dass dieser Fall nie eintreten kann.

Betrachten wir eine beliebige Eingabe mit N Elementen, unter denen mehr als $N/2$ Elemente gleich X sind. Nehmen wir an, dass am Ende von Phase 1 des Algorithmus kein X oben auf dem Stapel liegt. Da der Stapel immer lauter gleiche Elemente enthält, liegt also am Ende von Phase 1 überhaupt kein X am Stapel. Dann muss jedes X eines der folgenden beiden Schicksale erlitten haben:

1. Als X bearbeitet wurde, enthielt der Stapel ein oder mehrere Elemente ungleich X. Daher wurde X nicht auf den Stapel gelegt, sondern ein Element Y vom Stapel entfernt.
2. Als X bearbeitet wurde, war der Stapel leer oder enthielt Elemente gleich X. Daher wurde X auf den Stapel gelegt. Da am Ende von Phase 1 kein X mehr am Stapel liegt, muss nach diesem X später einmal ein Element Z gekommen sein, das dazu geführt hat, dass dieses X wieder vom Stapel entfernt wurde.

Gemäß diesen Überlegungen kann man jedes Element gleich X also einem Element ungleich X zuordnen: In Fall 1 wird X dem betreffenden Element Y zugeordnet, im Fall 2 dem betreffenden Element Z. Weiter sieht man, dass keine zwei Elemente X auf diese Weise demselben Element ungleich X zugeordnet werden. Es folgt also, dass es mindestens so viele Elemente ungleich X gibt wie es Elemente gleich X gibt. Da es insgesamt nur N Elemente gibt, ist dies ein Widerspruch zu der Voraussetzung, dass es mehr als $N/2$ viele Elemente gleich X gibt. Die Annahme, dass am Ende von Phase 1 kein X oben auf dem Stapel liegt, führt also auf einen Widerspruch und muss somit falsch sein. Es folgt daher, dass am Ende von Phase 1 wirklich ein X oben auf dem Stapel liegt und der Algorithmus somit korrekt X als Mehrheitselement erkennt.

Wie viele Vergleiche sind nötig?

Es ist auch interessant zu betrachten, wie viele Vergleiche zwischen Elementen ein Algorithmus zur Mehrheitsberechnung machen muss, bis das Ergebnis bestimmt ist. Als Vergleich bezeichnen wir hier die Operation, die prüft, ob zwei Elemente gleich sind oder nicht. Der oben beschriebene Majority-Algorithmus

macht in Phase 1 höchstens $N-1$ Vergleiche: Das erste Element wird ja einfach ohne einen Vergleich auf den Stapel gelegt, und jedes andere Element wird höchstens mit dem Element oben auf dem Stapel verglichen. Phase 2 kann ebenfalls leicht mit $N-1$ Vergleichen realisiert werden. Insgesamt macht der Algorithmus also auf einer Eingabe mit N Elementen höchstens $2N-2$ Vergleiche.

Da stellt sich natürlich die Frage: Geht es besser, d.h. mit weniger als $2N-2$ Vergleichen? Die Antwort ist ja, denn die folgende Abwandlung des Majority-Algorithmus kommt immer mit höchstens $\lceil 3N/2 \rceil - 2$ Vergleichen aus. (Die Notation $\lceil \cdot \rceil$ bezeichnet hier den zur nächsten ganzen Zahl aufgerundeten Wert; für $N=5$ gilt zum Beispiel $\lceil 3N/2 \rceil = \lceil 15/2 \rceil = \lceil 7,5 \rceil = 8$, und für $N=6$ hätten wir $\lceil 3N/2 \rceil = \lceil 18/2 \rceil = \lceil 9 \rceil = 9$.)

VERFEINERTER MAJORITY-ALGORITHMUS

1. Verwende zwei Stapel von Elementen, die anfangs beide leer sind.
2. Phase 1: Bearbeite nacheinander jedes der N gegebenen Elemente und führe dabei für jedes Element X das folgende aus:
 - Falls Stapel 2 nicht leer ist und X gleich dem Element oben auf Stapel 2 ist, so lege X auf Stapel 1.
 - Andernfalls lege X auf Stapel 2 und, falls Stapel 1 nicht leer ist, entferne auch das oberste Element von Stapel 1 und lege es auf Stapel 2.
3. Nimm an, dass am Ende von Phase 1 ein Y das oberste Element auf Stapel 2 ist.
4. Phase 2: Solange Stapel 2 nicht leer ist, wiederhole folgende Operationen:
 - Vergleiche das oberste Element auf Stapel 2 mit Y .
 - Falls die beiden Elemente gleich sind, entferne die beiden obersten Elemente von Stapel 2. (Falls Stapel 2 nur ein Element enthält, entferne es von Stapel 2 und lege es auf Stapel 1.)
 - Andernfalls (d.h. falls die beiden Elemente nicht gleich sind) entferne das oberste Element von Stapel 1 und das oberste Element von Stapel 2. (Falls Stapel 1 leer war und daher kein Element von Stapel 1 entfernt werden kann, so terminiere und gib aus, dass es kein Mehrheitselement gibt.)
5. Falls Stapel 1 nicht leer ist, gib Y als Mehrheitselement aus. Andernfalls gib aus, dass es kein Mehrheitselement gibt.

Um den verfeinerten Majority-Algorithmus besser zu verstehen, betrachten wir einmal, wie bei ihm auf der Eingabe

B,B,A,A,C,D,A,A,A

die Phase 1 abläuft:

Stapel 1 (von unten nach oben)	Stapel 2 (von unten nach oben)	Betrachtetes Element	Aktion
leer	leer	B	B auf Stapel 2 legen
leer	B	B	B auf Stapel 1 legen
B	B	A	A auf Stapel 2 legen, ein B von Stapel 1 auf Stapel 2 legen
leer	B,A,B	A	A auf Stapel 2 legen
leer	B,A,B,A	C	C auf Stapel 2 legen
leer	B,A,B,A,C	D	D auf Stapel 2 legen
leer	B,A,B,A,C,D	A	A auf Stapel 2 legen
leer	B,A,B,A,C,D,A	A	A auf Stapel 1 legen
A	B,A,B,A,C,D,A	A	A auf Stapel 1 legen
A,A	B,A,B,A,C,D,A		

Wir sehen, dass das Mehrheitselement A am Ende von Phase 1 tatsächlich oben auf Stapel 2 liegt. Phase 2 läuft nun wie folgt ab:

Stapel 1 (von unten nach oben)	Stapel 2 (von unten nach oben)	Aktion
A,A	B,A,B,A,C,D,A	Element oben auf Stapel 2 ist gleich A (das muss automatisch so sein, da wir A ja als das Element gewählt haben, das am Ende von Phase 1 oben auf Stapel 2 lag), also entferne zwei Elemente (D,A) von Stapel 2
A,A	B,A,B,A,C	Element C oben auf Stapel 2 ist ungleich A, also entferne ein Element von Stapel 2 und ein Element von Stapel 1
A	B,A,B,A	Element oben auf Stapel 2 ist gleich A, also entferne zwei Elemente (B,A) von Stapel 2
A	B,A	Element oben auf Stapel 2 ist gleich A, also entferne zwei Elemente (B,A) von Stapel 2
A	leer	

Nun ist der Stapel 2 leer, und Phase 2 endet somit. Da Stapel 1 nicht leer ist und ein A enthält, gibt der Algorithmus korrekt das Mehrheitselement A aus. Wir sehen auch, dass der Algorithmus in Phase 2 nur 4 Vergleiche gemacht hat. Beim ersten der vier Vergleiche war das Ergebnis des Vergleichs außerdem sowieso klar, es mussten also in Phase 2 nur drei echte Vergleiche gemacht werden.

Wir wollen kurz skizzieren, wieso der verfeinerte Majority-Algorithmus richtig funktioniert. Die Rolle des Stapels aus dem ersten Majority-Algorithmus wird hier im Wesentlichen von Stapel 1 zusammen mit dem obersten Element von Stapel 2 übernommen. Ferner gilt, dass Stapel 1 immer identische Elemente enthält (die auch gleich dem obersten Element auf Stapel 2 sind) und dass auf Stapel 2 niemals zwei gleiche Elemente direkt aufeinander zu liegen kommen. Daraus folgt, dass am Ende von Phase 1 das Element Y oben auf Stapel 2 der einzige Kandidat für das Mehrheitselement ist. In jeder Iteration von Phase 2 werden dann zwei Elemente entfernt, von denen eines (aber niemals beide) gleich Y ist. Y ist also genau dann das Mehrheitselement, wenn am Ende von Phase 2 der Stapel 1 nicht leer ist (d.h. noch mindestens ein Y enthält). Somit folgt die Korrektheit des verfeinerten Majority-Algorithmus. Ferner macht der Algorithmus in Phase 1 höchstens $N-1$ Vergleiche und in Phase 2 höchstens $\lfloor N/2 \rfloor - 1$ Vergleiche (davon kann man sich überzeugen, indem man in Betracht zieht, dass in jeder Iteration von Phase 2 mit einem Vergleich zwei Elemente entfernt werden; das läßt sich auch an dem obigen Beispiel gut beobachten). Somit sieht man, dass der Algorithmus insgesamt höchstens $\lfloor 3N/2 \rfloor - 2$ Vergleiche braucht, um das Mehrheitselement zu bestimmen (oder herauszufinden, dass es keines gibt).

Geht es noch besser? Dieses Mal ist die Antwort nein! Man kann zeigen, dass jeder Algorithmus auf manchen Eingaben mit N Elementen mindestens $\lfloor 3N/2 \rfloor - 2$ Vergleiche machen muss, um das Mehrheitselement zu bestimmen. Besser als der verfeinerte Majority-Algorithmus geht es also im allgemeinen bezüglich der Anzahl der Vergleiche nicht.

Sowohl der verfeinerte Majority-Algorithmus als auch der Beweis, dass jeder Algorithmus zur Mehrheitsbestimmung auf manchen Eingaben mindestens $\lfloor 3N/2 \rfloor - 2$ Vergleiche machen muss, wurden bereits im Jahr 1982 von M.J. Fischer und S.L. Salzberg in *Journal of Algorithms*, einer internationalen Fachzeitschrift für Algorithmen, vorgestellt.

Anwendungen und Erweiterungen

Das Problem der Bestimmung des Mehrheitselements tritt nicht nur bei der Auswertung von Wahlen auf, sondern auch in ganz anderen Anwendungen. Zum Beispiel kann man sicherheitskritische Berechnungen, bei denen ein korrektes Ergebnis extrem wichtig ist, von N verschiedenen Prozessoren unabhängig voneinander ausführen lassen und am Ende unter allen N Ergebnissen das Mehrheitselement bestimmen und als

Ergebnis verwenden. Auf diese Weise bekommt man immer das richtige Ergebnis, sofern weniger als die Hälfte der N Prozessoren Fehler haben und ein falsches Ergebnis liefern.

Das Mehrheitselement unter N Elementen ist das Element, das mehr als $N/2$ mal vorkommt. Etwas allgemeiner kann man **häufige** Elemente betrachten, die mehr als N/K mal vorkommen, wobei K eine feste Zahl ist. Für $K=10$ etwa sind das die Elemente, die mehr als $N/10$ mal vorkommen. Interessant sind häufige Elemente zum Beispiel bei der Beobachtung des Datenverkehrs im Internet, wo man gerne wissen möchte, welche Anwendungen oder welche Benutzer den meisten Verkehr produzieren. Da die Datenpakete extrem schnell bearbeitet werden müssen, ist hier ein Algorithmus nötig, der jedes neue Datenpaket in kürzestmöglicher Zeit erledigt. Der Majority-Algorithmus lässt sich auf solche Problemstellungen verallgemeinern; dies wird in dem Artikel „Frequency Estimation of Internet Packet Streams with Limited Space“ von Erik D. Demaine, Alejandro Lopez-Ortiz und Ian Munro beschrieben, der im Jahr 2002 auf der Konferenz ESA (*European Symposium on Algorithms*) vorgestellt wurde.

Was können wir aus diesem Beispiel lernen?

- Die naheliegendste Methode ist nicht automatisch *effizient*,
- Oft gibt es geschickte Algorithmen, die dieselbe Aufgabe mit viel weniger Aufwand lösen können.
- Es ist nicht immer leicht zu sehen, dass ein Algorithmus tatsächlich immer das *richtige Ergebnis* liefert.
- Manchmal kann man zeigen, dass es unmöglich ist, einen noch besseren Algorithmus für ein Problem zu finden.

Autoren:

- Dr. Thomas Erlebach
<http://www.cs.le.ac.uk/people/te17/>

Weiterführende Materialien:

- Foliensatz des Beitrags (ppt, mit Animationen)
<http://www-i1.informatik.rwth-aachen.de/~algorithmus/Algorithmen/algo33/Mehrheitsbestimmung.ppt>
- Foliensatz des Beitrags (pdf, ohne Animationen)
<http://www-i1.informatik.rwth-aachen.de/~algorithmus/Algorithmen/algo33/Mehrheitsbestimmung.pdf>