Union-Find*

Hans Jürgen Ohlbach

23. März 2018

Keywords: Verwaltung disjunkter Mengen

Empfohlene Vorkenntnisse: $\mathcal{O}$-Notation

Inhaltsverzeichnis

1 Einleitung **2**

2 Die Idee an einem Beispiel **2**

3 Union-Find **5**

4 Anwendungen **6**

*Dieser Text ist Teil einer Sammlung von Miniskripten zur Einführung in die Informatik. Er ist in erster Linien für Nichtinformatiker gedacht, kann aber natürlich auch als erste Einführung für Informatiker nützlich sein.

1 Einleitung

Disjunkte Mengen sind Mengen, die keinen gemeinsamen Schnitt haben, z.B. gerade und ungerade Zahlen, Männer und Frauen (zumindest genetisch), Äpfel und Birnen, usw. Im Computer kann man natürlich nur *endliche* disjunkte Mengen konkret abspeichern. Die einfachste Methode wäre natürlich, die verschiedenen Mengen in verschiedenen Arrays oder Listen zu speichern. Ein Test, ob zwei Element in der gleichen Menge sind, oder nicht, würde dann aber eine Suche durch diese Listen oder Arrays erfordern. Mit der Union-Find Datenstruktur kann man diese Suche ganz erheblich beschleunigen, so dass man auch mit sehr große Mengen effizient umgehen kann.

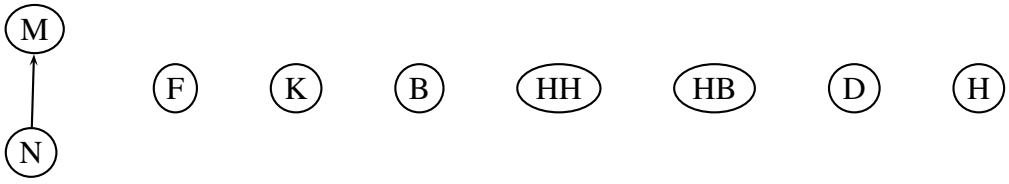
2 Die Idee an einem Beispiel

Um die Idee zu erläutern betrachten wir deutsche Städte mit Flughäfen. Zunächst gibt es keine Flugverbindungen zwischen den Städten. Nach und nach werden Flugverbindungen eingerichtet. Jede der disjunkten Mengen beinhaltet die Städte, die man dann über diese Verbindungen, evtl. mit Umsteigen, erreichen kann. Sind zwei Städte in verschiedenen Mengen, dann gibt es keine Möglichkeit, diese per Flugzeug zu erreichen.

Die Städte in unserem Beispiel sind München (M), Nürnberg (N), Frankfurt(F), Köln (K), Berlin (B), Hamburg (HH), Bremen (HB), Dortmund (D), Hannover (H).

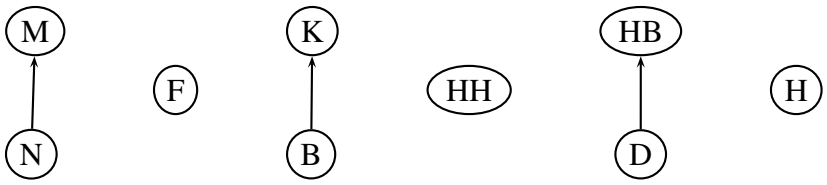
Zunächst gibt es überhaupt keine Flugverbindungen. Jede der Städte ist dann allein in der Menge der verbundenen Städte.

Jetzt wird eine Verbindung zwischen München und Nürnberg eingerichtet. Bei der normalen Vorgehensweise würde man jetzt eine Liste [M,N] erzeugen. In dem Union-Find Ansatz bauen wir stattdessen einen Baum bzw. Wald auf. Der würde so aussehen:



M und N könnten auch umgekehrt sein, das spielt in diesem Stadium keine Rolle. Wichtig ist aber der Rückwärtsverweis vom Unterknoten N zum Oberknoten M.

Jetzt richten wir weitere Verbindungen ein, zwischen Köln und Bonn, zwischen Bremen und Dortmund. Die Struktur sieht dann so aus.

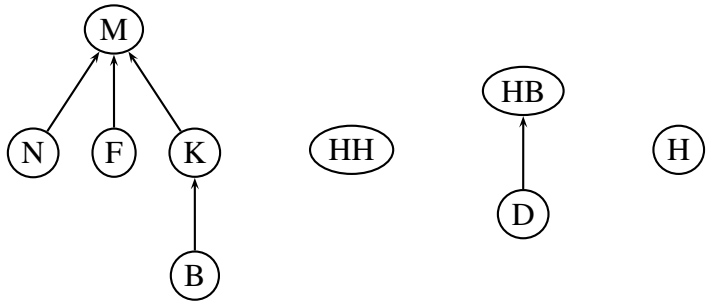


Nun kommt eine Verbindung von Nürnberg nach Frankfurt hinzu.

Obwohl Nürnberg und Frankfurt eine neue Verbindung haben, kommt Frankfurt direkt unter den Wurzelknoten München. Der Baum spiegelt eben *nicht* die Flugverbindungen wider, sondern hilft, auf effiziente Weise herauszubekommen, ob es zwischen den verschiedenen Städten überhaupt eine Verbindung gibt.

Wie geht das?
Um z.B. herauszubekommen, ob es zwischen Frankfurt und Nürnberg eine Verbindung gibt, verfolgt man die Rückwärtspfeile bis zum Wurzelknoten. In diesem Fall sind die Wurzelknoten identisch. Daher liegen Frankfurt und Nürnberg in der gleichen Menge. Macht man das z.B. mit Frankfurt und Bonn, dann sind die beiden Wurzelknoten verschieden. Sie liegen dann in verschiedenen Mengen.

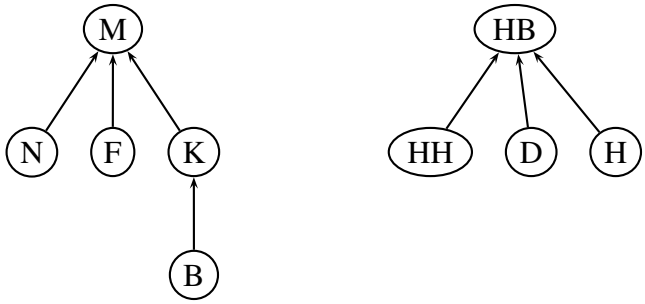
Jetzt richten wir eine Verbindung zwischen Nürnberg und Bonn ein. Für Nürnberg ist der Wurzelknoten München, und für Bonn ist der Wurzelknoten Köln. Der Wurzelknoten des kleinere Baums (Köln) wird jetzt direkt unter den Wurzelknoten des größeren Baums (München) gehängt. *Das ist die* **Union**-*Operation.*



Jetzt kann man herausfinden, ob es zwischen Frankfurt und Bonn eine Flugverbindungsmöglichkeit gibt, indem man für beide über die Rückwärtspfeile die Wurzelknoten ermittelt und vergleicht. In beiden Fällen ist es München. Man kann also per Flugzeug von Frankfurt nach Bonn kommen.

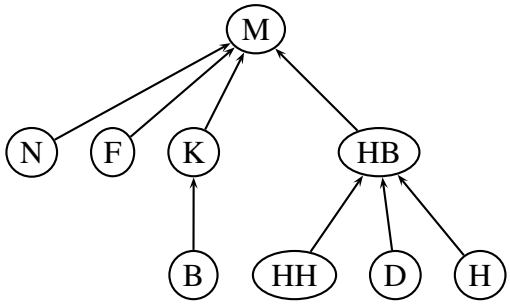
Testet man das mit Frankfurt und Dortmund, dann sind es verschiedene Wurzelknoten. Daher gibt es keine Verbindungsmöglichkeit.

Um das Beispiel weiterzuführen, richten wir noch Flugverbindungen zwischen Hamburg und Bremen, sowie zwischen Dortmund und Hannover ein.



Um jetzt zu testen, ob es Verbindungsmöglichkeiten zwischen z.B. Bonn und Hannover gibt, muss man von Bonn aus 2 Rückwärtspfeilen folgen, und von Hannover aus 1 Rückwärtspfeil. Mit 3 solchen Operationen und einem Vergleich München $\neq$ Bremen, erhält man das Ergebnis: keine Verbindung.

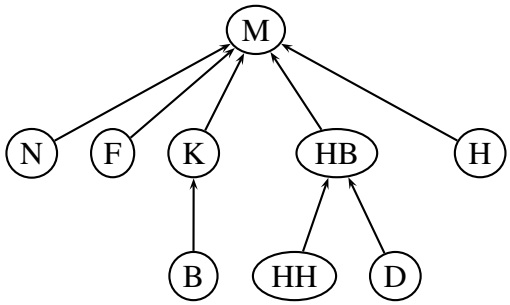
Zum Schluss richten wir noch eine Verbindung zwischen Bonn und Dortmund ein. Jetzt wird die Wurzel des flacheren Baums (Bremen) unter die Wurzel des tieferen Baums (München) gehängt.



Damit sind alle Städte untereinander per Flugverbindung erreichbar. Über jeweils maximal 2 Rückwärtspfeile kommt man von allen Städten nach München, was deutlich weniger Aufwand ist, als die Suche in Listen.

Die Verfolgung der Rückwärtspfeile bis zum Wurzelknoten ist die **Find**-Operation. Sie ist die Basis, um die Zugehörigkeit zu den Mengen effizient entscheiden zu können.

Die Find-Operation wird um so effizienter, je weniger Rückwärtspfeile man verfolgen muss. Mit folgender Idee kann man das noch verbessern. Angenommen *Find(Hannover)* wird aufgerufen, und liefert München. Jetzt wissen wir, dass es von Hannover einen Rückwärtsweg nach München gibt. Um zukünftige *Find(Hannover)*-Operationen noch effizienter zu machen, kann man Hannover direkt unter München hängen.



Jede Find-Operation mit dieser Optimierung (*Pfadkompression*) verbessert also die Effizienz zukünftiger Find-Operationen.

3 Union-Find

Etwas formaler kann man diese Struktur zur Repräsentation disjunkter Mengen folgendermaßen beschreiben:

Datenstruktur: Disjunkte Mengen werden als Bäume repräsentiert.
Jeder Baum hat ein ausgezeichnetes Element als Wurzelknoten.
Von jedem Unterknoten zum Oberknoten gibt es einen Rückwärtspfeil.

Union-Operation: *Union(Baum1,Baum2)* hängt den Wurzelknoten des flacheren der beiden Bäume unter den Wurzelknoten des tieferen der beiden Bäume. Falls beide Wurzelknoten identisch sind, muss natürlich nichts getan werden.

Find-Operation: *Find(X)* verfolgt von Knoten X die Rückwärtspfeile und liefert den Wurzelknoten. Falls *Find(X) = Find(Y)*, dann sind X und Y in den gleichen Mengen, ansonsten sind sie in verschiedenen Mengen.

Pfadkompression: Damit bezeichnet man die Technik, einen Knoten *X* direkten unter den Wurzelknoten *Find(X)* zu hängen, um zukünftige Find-Operation zu beschleunigen.

Zeitkomplexität:
Für die Union-Operation nehmen wir an, dass an jedem Knoten noch zusätzlich die Tiefe seines Unterbaumes abgespeichert wird. Damit besteht die Union-Operation nur aus der Einführung eines neuen Rückwärtspfeiles. Das geht mit konstantem Aufwand. Daher ist $\mathit{Union} \in \mathcal{O}(1)$.

Für die Komplexität der Find-Operation ist die Heuristik, dass man den flacheren Baum unter den Wurzelknoten des tieferen Baum hängt, wichtig. Damit kann man zeigen, dass die maximale Baumtiefe bei n Knoten $\mathcal{O}(\log(n))$ ist. Die Find-Operation muss also maximal $\log(n)$ Rückwärtspfeile verfolgen. Das geht dann mit $\mathcal{O}(\log(n))$ Aufwand.

Um diese Verbesserung gegenüber dem einfachen Verfahren, suchen in Listen, einzuschätzen, betrachten wir ein paar Zahlen:

n	$\log_2(n)$	Verbesserung
10	$\sim 3,3$	~ 3
100	$\sim 6,64$	~ 15
1024	10	102.4
1.000.000	~ 20	~ 50171

4 Anwendungen

Die Notwendigkeit, disjunkte Mengen zu verwalten, gibt es sowohl konkret für Anwendungsdaten, als auch implizit in anderen Algorithmen. Wenn es z.B. um die Verwaltung von Graphen geht, dann kann es wichtig sein, Zusammenhangskomponenten zu verwalten. Eine Zusammenhangskomponente in einem Graphen ist ein Teilgraph, wo es von jedem Knoten zu jedem anderen Knoten einen Pfad gibt. Zwei verschiedene Zusammenhangskomponenten sind nicht über Pfade erreichbar.

Etwas abstrakter, die Äquivalenzklassen von Äquivalenzrelationen sind disjunkte Mengen. Wenn sie endlich sind können sie mit Union-Find verwaltet werden.

Wenn jedoch von vorne herein die Struktur der Mengen feststeht, z.B. Männer und Frauen, dann lohnt sich Union-Find nicht. Dann kann man jedes Element mit der Zusatzinformation, zu welcher Menge es gehört, versehen, und das wars.

Nur wenn sich die Mengenstruktur dynamisch ändert, indem Mengen verschmolzen werden, ist Union-Find angebracht. Werden allerdings Mengen wieder getrennt, in unserem Flugverbindungsbeispiel wenn Verbindungen gestrichen werden, dann muss die Struktur neu aufgebaut werden. Das ist etwas aufwendiger.