Tiefensuche und Breitensuche

Die Tiefensuche ist ein Fall von brute force

Eines der Verfahren, die mit *brute force*[1] eine Lösung zu finden versuchen, ist die Tiefensuche. Bei der Tiefensuche wird immer der erste noch nicht untersuchte Ast des Suchbaumes[2] weiter verfolgt. Wird er verworfen, dann geht man eine Stufe zurück und schaut sich dort nach Alternativen um. Findet man eine Alternative, dann versucht man diese und verfolgt den Baum auch hier nach demselben Prinzip.

Hier ist zu erkennen: Eine typisch rekursive Beschreibung des Problems drängt sich auf!

Tiefensuche mit backtracking

Typisch für die funktionale Programmierung ist, dass man im Auswertungsteil keinen Ablauf beschreibt, sondern wie der Rückgabewert zu bestimmen ist. Bei rekursivem Aufruf solcher Funktionen trifft man während der Bearbeitung bei jedem Schritt also immer wieder auf denselben Auswertungsteil. Das bedeutet natürlich nicht, dass jedesmal dasselbe gemacht wird. Dann wäre unser Programm in einer Endlosschleife gelandet. Bei der Bearbeitung muss also mindestens ein Parameter sich verändern. Daher muss der Auswertungsteil alle verschiedenen Möglichkeiten beschreiben, auf die man dabei treffen kann.

Es gibt typische Abbruchbedingungen, die dabei vorkommen. Bei Listen wird man in der Regel abbrechen müssen, wenn man auf eine leere Liste trifft. Bei Zahlen ist häufig der Wert 0 Abbruchwert.

Alternativen bei der Tiefensuche

Beim Eintreten in die Rekursionsstufe sind alle Alternativen zu bestimmen, die es hier gibt. Man spricht von einer Expansion des Suchraums (Graphen: ein Knoten wird expandiert). Dafür muss uns eine Funktion zur Verfügung stehen, die uns die möglichen Alternativen bestimmt. Typisch für die Tiefensuche ist, dass von diesen mit Hilfe einer Nachfolger-funktion bestimmten Alternativen zunächst nur eine weiter verfolgt wird, ehe man sich mit den anderen beschäftigt.

Bei einer Suche haben wir wie bei jeder Rekursion zunächst die Fälle zu untersuchen, die zu einem Rekursionsabbruch führen:

- Zielwert erreicht
  Damit hat unsere Suche zu einem Erfolg geführt. Es muss in diesem Fall ein Rekursionsabbruch erfolgen, allerdings mit einem Rückgabewert, der den Erfolgsfall kennzeichnet. In vielen Fällen wird man außerdem als Nebeneffekt eine Ausgabe auf dem Bildschirm haben wollen, bei welcher der erfolgreiche Suchpfad ausgegeben wird.
- Keine Alternativen (mehr)
  Dies bedeutet für die Suche den Misserfolgsfall. Es ist das backtracking auszulösen und ein Wert zurückzugeben, der auf der aufrufenden Rekursionsstufe als Misserfolgsfall interpretiert werden kann. Das kann beispielsweise der boolean – Wert `#f` sein, eine leere Liste o.ä.
- Die erste Alternative kommt schon im Suchpfad vor.

1 Brutale Gewalt, hier also einfache alle Möglichkeiten durchprobieren.
2 In unserem Fall ist es ein Baum. Das ist leider jedoch nicht zwingend, wie wir noch sehen werden.

Wir haben bei unserer Suche einen Zyklus gefunden. Bei Suchbäumen tritt dieser Fall natürlich nicht auf, in vielen Suchräumen muss man aber mit Zyklen rechnen. Die Tiefensuche muss mit dem Problem fertig werden, sie versackt sonst in einer Endlosschleife.
Das Mittel dazu ist das Führen einer Besuchtliste. Auch dies bedeutet für die Suche den Misserfolgsfall. Es ist das backtracking auszulösen und ein Wert zurückzugeben, der auf der aufrufenden Rekursionsstufe als Misserfolgsfall interpretiert werden kann.

- Rekursionsaufruf für die erste Alternative
  Dieser Aufruf ist ein normal rekursiver Aufruf. Zu beachten ist, dass er zu einem Wert führt, der in irgendeiner angemessenen Weise interpretiert werden muss. Bekommt man von der aufgerufenen Stufe Erfolg gemeldet, ist ggf. als Nebeneffekt eine Ausgabe zu machen. Abhängig von der Frage, ob man eine vollständige Suche durchführt oder nicht, sind danach noch die Alternativen zu behandeln. In der Regel ist auch auf dieser Stufe der Wert zurückzugeben, der den Erfolgsfall kennzeichnet.
- Aufruf der restlichen Alternativen
  Dieser Rekursionsaufruf erfolgt nun allerdings endrekursiv, wir dürfen also in der Rekursionsebene nicht weiter absteigen.

Alternativen wirklich erzeugen?

Typisch für Tiefensuche mit backtracking ist, dass vor dem Schritt in die Tiefe die Alternativen nicht wirklich bestimmt werden, sondern ihr Erzeugen nur angelegt wird. Scheme legt den noch möglichen Aufruf [möglich deswegen, weil er nur erfolgt, wenn der Schritt in die Tiefe nicht erfolgreich war] auf dem Systemstack ab und führt ihn ggf. erst nach dem Rücksprung aus.

Systemstack

Im Systemstack verwaltet Scheme also ohne unser Zutun den Programmzustand auf jeder Rekursionsstufe, also welchen Wert die Parameter aktuell haben und wo wir uns im Funktionsablauf befinden. Wir brauchen uns daher um die Verwaltung der Datenstruktur nur bei den Parametern der Aufrufe zu kümmern.

Breitensuche, ein weiterer Fall von bruteforce

Den Unterschied zwischen Tiefensuche und Breitensuche zu beschreiben und zu verstehen ist sehr einfach. Die Stichworte sind ***Tiefe zuerst*** oder ***Breite zuerst***[1].

Bei der Breitensuche müssen erst alle Alternativen der aktuellen Stufe bearbeitet werden, bevor eine Stufe tiefer gegangen wird.

Kaum eine Änderung nötig?

Bei diesem kleinen Unterschied ist die erste Vermutung, dass man mit demselben Programm bei geringfügigen Änderungen zur Lösung kommen müsste. Das ist jedoch leider nicht der Fall. Während uns bei der Tiefensuche der Systemstack die Sicherung der möglichen Alternativen bis zum Wiedereintreten in die aktuelle Ebene abnimmt – Daten und Programmposition werden wieder in den vorigen Zustand versetzt – müssen wir uns bei der Breitensuche selbst darum kümmern.

Die Breitensuche benötigt eine andere Datenstruktur

Oben heißt es: Bei der Breitensuche muss erreicht werden, dass erst alle Alternativen einer Stufe bearbeitet werden, bevor eine tiefere Stufe angegangen wird.

Die dazu verwendete Datenstruktur heißt Warteschlange (queue). Der Unterschied zwischen ***stack*** und ***queue*** wird schlagwortartig formuliert durch LIFO und FIFO:

Bei der **Tiefensuche** heißt es:

*Zum Speichern der Möglichkeiten muss eine **LIFO** - Struktur (= **l**ast **i**n **f**irst **o**ut), also eine **stack** - Struktur (Stapel) benutzt werden.*

Bei der **Breitensuche** muss es heißen:

*Zum Speichern der Möglichkeiten muss eine **FIFO** - Struktur (= **f**irst **i**n **f**irst **o**ut), also eine **queue** - Struktur (Warteschlange) benutzt werden.*

Geht man davon aus, dass immer vorn geleert wird[2], dann unterscheiden sich beide in der Position, an der gefüllt wird: Beim stack wird immer vorn eingebaut *und* abgebaut, bei der Warteschlange wird zwar auch *vorn* abgebaut, aber immer *hinten* eingebaut!

Abgesehen davon ist das Programm tatsächlich vergleichbar. Zu der zu bearbeitenden Alternative werden jeweils ihre Nachfolger bestimmt und bearbeitet oder gespeichert.

Aufgabe zum Vergleich von Tiefensuche und Breitensuche

- Welche Vorteile und Nachteile bieten jeweils die Tiefensuche und die Breitensuche?
- Formulieren und Probieren!

1 Es geht also nicht um die Frage, ***ob*** in die Tiefe und Breite, sondern ***wann*** während der Bearbeitung.
2 Diese Argumentation kann man natürlich auch umdrehen.