Backtracking

Theorie

Wir wollen systematisch *alle* Lösungswege für ein Problem ausprobieren. Dazu versuchen wir schrittweise ein Teillösung weiter auszubauen, bis wir eine Lösung für das gesamte Problem gefunden haben. Gelangen wir bei unserer Suche in eine Sackgasse, finden also keine weitere Ausbaumöglichkeit mehr, dann machen wir unsere letzten Schritte so lange rückgängig, bis sich neue, noch nicht betrachtete Lösungsvarianten ergeben.

So finden wir garantiert eine Lösung, sofern auch eine existiert, oder wissen am Ende mit Sicherheit, dass es keine Lösung gibt. Man bezeichnet dieses Vorgehen auch als Versuch-und-Irrtum-Prinzip (trial-and-error).

Implementiert wird ein Backtracking-Verfahren meist mittels Rekursion. Wir schreiben also eine Funktion, die um die Lösung eines Problems zu finden, sich selbst auf kleinere Teilprobleme aufruft.

Rechenaufwand

Meist bedeutet ein Backtracking-Ansatz exponentielle Laufzeit. Häufig müssen für eine Eingabe von N Elementen alle Teilmengen (Zweierpotenz von N) oder alle Reihenfolgen (Fakultät von N) untersucht werden. In der Praxis bedeutet dies, dass meist nur kleine Eingaben mit bis rund 15 Elementen berechnet werden können.

Problemtypen

Wege suchen (z.B. Labyrinth durchsuchen)

Rucksackproblem (allenfalls auch mit DP lösbar)

Färbeproblem (4-Färbbarkeit von Landkarten)

Spiele (Solitär, Sudoku, Schach, Mühle)

Implementierungsansatz

```
backtrack(int a[], int k) {
    falls a eine Lösung ist → verarbeite die Lösung
    sonst
        k := k+1
        konstruiere Kandidatenmenge c basierend auf a
        für alle Kandidaten c[i]
            a[k] := c[i]
            backtrack(a,k)
            falls fertig → return
```

Dabei ist a die Liste der k schon durchgeführten Schritte.

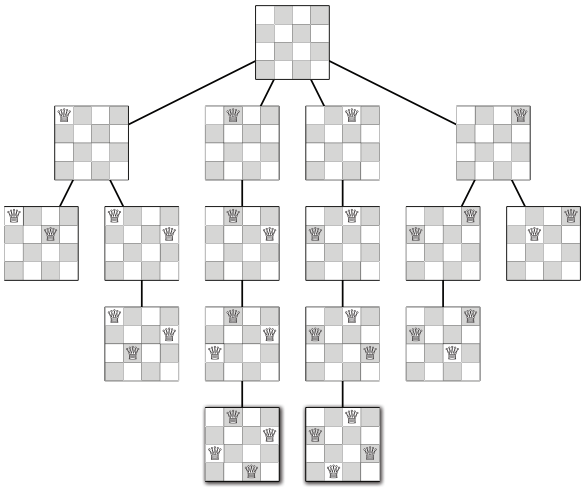
Branch and Bound

Während wir alle Lösungmöglichkeiten ausprobieren, gehen wir durch alle Knoten in einem sogenannten Suchbaum. Häufig lässt sich schon im Voraus erkennen, dass ein ganzer Teilbaum keine Lösung enthalten kann. Damit kann der Suchbaum häufig stark beschnitten werden und somit die Laufzeit verkürzt werden.

Beispiele

Damenproblem

Zähle die Anzahl Möglichkeiten n Damen auf einem n-x-n Schachbrett aufzustellen, sodass sie sich gegenseitig nicht angreifen können, d.h. dass sich in jeder Zeile, Spalte und Diagonale maximal eine Dame befindet.



Berechnungsbaum für das 4x4-Damenproblem [1]

Summieren

Gegeben eine Menge von n Zahlen. Das Summieren zweier Zahlen kostet immer so viel wie deren Summe. Dabei werden diese zwei Zahlen aus der Menge entfernt und die neue Summe wird in die Menge eingefügt. Dies wird wiederholt bis nur noch eine Zahl übrig bleibt. Minimiere die entstehenden Kosten.

Grösstes kleinstes gemeinsames Vielfaches

Gegeben N. Gesucht eine Art N als Summe von positiven Zahlen zu schreiben, sodass das kgV dieser Summanden maximal wird.

Literaturtipps und Bildquelle

[1] Handouts von Sebastian Millius besonders Handout 8 zu Backtracking: http://goo.gl/5DCib

[2] Programming Challenges von Skiena, Revilla