Maximum Sub Sequence Problem

Angenommen, man handelt mit Aktien und möchte rückblickend den grössten möglichen Kursgewinn berechnen, d.h. den optimalen Einstiegs- (Kauf) und Ausstiegszeitpunkt (Verkauf) bestimmen[1]. Oft sieht man die Kurse in einem Graphen. Die folgende Abbildung illustriert die Aufgabe.

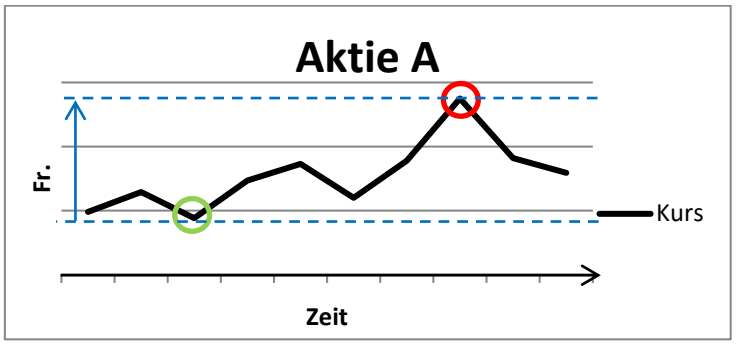

Abbildung 1 - Beispielkurs

Bei einem Kursverlauf wie in Abbildung 1 wäre es optimal, wenn man die *Aktie A* zum grün markierten Zeitpunkt gekauft und zum rot markierten Zeitpunkt wieder verkauft hätte. Der Kursgewinn hätte dann dem blauen Pfeil entsprochen.

Der Kursverlauf der Aktie sei als Liste mit den *Differenzen* von Tag zu Tag gegeben:

[31, -41, 59, 26, -53, 58, 97, -93, -23]

Der Kurswert steigt also zunächst um 31 Punkte, sinkt dann um 41 Punkte, und steigt dann wieder um 59 Punkte. Nach dieser Zeit ist der Wert gesamthaft also um 49 Punkte gestiegen.

Nun sucht man einen Start- und einen End-Index, so dass die Summe zwischen diesen möglichst gross ist. Im Beispiel wären das der Start-Index 2 und der End-Index 6. Die Summe der Werte in diesem Bereich ist 187.

Sinkt der Kurs über die ganze Periode, sind alle Werte in der Liste negativ. Dann ist es natürlich am besten, gar nicht erst zu kaufen und entsprechend auch nichts zu verkaufen. Der Gewinn ist dann 0.

Wer schon eine Lösungsidee hat, kann das Folgende überspringen. Ansonsten hilft vielleicht diese mathematisch formulierte Nachbedingung weiter:[2]

$$result = max\left(\{0\} \cup \bigcup_{start=0}^{a.length-1} \bigcup_{end=start}^{a.length-1} \sum_{i=start}^{end} a[i]\right)$$

Es muss also das Maximum aus den Summen, die sich aus allen möglichen Kombinationen von Start- und End-Indizes ergeben, berechnet werden.

Diese Formel lässt sich direkt in ein paar Programm-Schleifen übersetzen. Damit erhält man eine korrekte Lösung der Aufgabe. Allerdings kann man diese noch um einiges verbessern…

Die Aufgabe:

Programmiert eine Lösung für die oben beschriebene Aufgabe in einer Funktion.

Wie kann man diese Funktion testen? Welche Situationen sollte man testen? Funktioniert die Funktion in allen Fällen korrekt?

Wie lange braucht das Programm dieser Funktion, wenn die Eingabe-Liste immer länger wird? Wie verändert sich die Laufzeit, wenn man die Länge der Eingabe-Liste verdoppelt?

[1] In diesem Beispiel sind wir am absoluten Kursgewinn und nicht am Kursgewinn pro Zeit interessiert.

[2] Das *Resultat* ist der maximale Wert aus der Menge aller Summen von aufeinanderfolgenden Listen-Elementen, deren *Start-Index* zwischen *0* und *len(a)-1* und deren *End-Index* zwischen dem *Start-Index* und *len(a)-1* liegt. und dem Wert 0.