



Vorbereitung

Lies E Kapitel 3 ohne 3.6 und 3.9 und schau das Video der Woche.

Dienstag

Aufgabe 1 (Editieren).

- a) (einfach) Welche Editiersequenz ist hier visualisiert und wie viele Editieroperationen wurden benutzt?

```
P L U T O
P L A N E T
```

- b) (einfach) Füll alle 25 Einträge der Tabelle $\text{Edit}[i, j]$ mit $A = \text{PLUTO}$ und $B = \text{PLANET}$ aus. Gib die optimale Editiersequenz an.
- c) Die Tabelle $\text{Edit}[i, j]$ hat mn Einträge und braucht daher $O(mn)$ Platz. Wie kann das dynamische Programm für die Editiersequenz *platzsparend* gemacht werden, sodass zu jedem Zeitpunkt nur noch $O(\min\{m, n\})$ Platz gebraucht wird? Die Laufzeit soll dabei asymptotisch gleich bleiben.

Aufgabe 2 (Pascalsches Dreieck).

Für die Binomialkoeffizienten gilt folgende Rekursionsformel:

$$\binom{n}{0} = \binom{n}{n} = 1 \quad \text{für alle } n \geq 0, \text{ und}$$
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{für alle } n, k \text{ mit } 1 \leq k \leq n-1.$$

Wir möchten eine Funktion implementieren, die $\binom{n}{k}$ ausrechnet.

- a) Entwirf zunächst eine rekursive Funktion $\text{RECBINOM}(n, k)$, die schlicht die Rekursionsformel rekursiv benutzt. Wie viele rekursive Aufrufe braucht $\text{RECBINOM}(n, k)$? Gib die Antwort in asymptotischer Notation in Abhängigkeit von n und k an.
- b) Entwirf nun mit Hilfe von dynamischer Programmierung eine iterative Funktion $\text{ITERBINOM}(n, k)$, die $O(nk)$ arithmetische Operationen ausführt und dabei $O(nk)$ Zahlen im Speicher hält.
- c) (schwer) Entwirf nun mit Hilfe von platzsparender dynamischer Programmierung eine iterative Funktion $\text{ITERBINOM2}(n, k)$, die ebenfalls $O(nk)$ arithmetische Operationen braucht, dabei aber nur $O(k)$ Zahlen in Speicher halten muss.

Aufgabe 3 (Betriebsfeier). Du organisierst die jährliche Betriebsfeier bei der Algorithmen-Bank. Die Mitarbeiter:innen der Bank sind in einer strikten Hierarchie angeordnet: ein Baum, in dem die Eigentümerin der Bank die Wurzel darstellt. Die allwissende Personalabteilung hat für jede:n Angestellte:n eine reelle Zahl berechnet, die angibt, wie „lustig“ diese Person ist. Damit die Feier entspannt bleibt, gibt es folgende Einschränkung für die Gästeliste: ein:e Mitarbeiter:in darf nicht an der Feier teilnehmen, wenn seine:ihre direkte Vorgesetzte teilnimmt. Andererseits *muss* die Eigentümerin an der Feier teilnehmen, obwohl sie nicht besonders lustig ist; es ist ja schließlich auch ihre Firma. Beschreibe und analysiere einen Algorithmus, der diejenige Gästeliste für die Betriebsfeier berechnet, die die Summe aller „lustig“ Bewertungen der Gäste maximiert.

Donnerstag

Aufgabe 4 (Genaueres Rückgeld). In einem früheren Leben hast du als Kassierer:in in der verlorenen antarktischen Kolonie Nadiria gearbeitet, wo du die meiste Zeit des Tages Rückgeld an deine Kund:innen gegeben hast. Weil Papier ein knappes und wertvolles Gut in der Antarktis ist, gab es das Gesetz, dass Kassierer:innen immer die kleinstmögliche Anzahl an Geldscheinen zurückgeben müssen. Die Währung von Nadiria heißt Traum-dollar und war in den folgenden Scheinen verfügbar: \$1, \$4, \$7, \$13, \$28, \$52, \$91, \$365.

- (einfach) Der gierige Wechselalgorithmus nimmt immer den größten Schein, der kleiner ist als das noch zu gebende Rückgeld. Zum Beispiel, um \$122 mit dem gierigen Algorithmus zurückzugeben, nehmen wir zuerst einen \$91 Schein, dann einen \$28 Schein, und am Ende drei \$1 Scheine. Gib ein Beispiel an, in dem der gierige Algorithmus mehr Traum-dollar-scheine als Wechselgeld gibt als die minimal mögliche Anzahl. *Hinweis: Es könnte einfacher sein, ein kleines Programm zu schreiben, als von Hand ein Beispiel zu konstruieren.*
- Beschreibe und analysiere einen rekursiven Algorithmus, der für eine gegebene Zahl k die kleinste Anzahl an Scheinen berechnet, die benötigt werden, um k Traum-dollar Rückgeld zu geben. (Mach dir erstmal keine Gedanken darüber, wie man den Algorithmus schnell macht; stell einfach sicher, dass er korrekt ist.)
- Beschreibe einen Algorithmus, der mithilfe von dynamischer Programmierung dasselbe algorithmische Problem löst. (Dieser Algorithmus soll schnell sein.)

Aufgabe 5 (Maximales Teilfeld II). Erinnere dich zurück an die Aufgabe zu den maximalen Teilfeldern (Aufgabe 6 in Woche 3). Eine Lösung nutzt eine rekursive Funktion $\text{MAXTEILFELD}(j)$, die die Summe eines maximalen Teilfelds ausrechnet, das in j endet. Das heißt, die Funktion liefert unter den Teilfeldern $A[0..j]$, $A[1..j]$, \dots , $A[j..j]$ die maximale Summe. Ohne auf deine Notizen zu schauen, stelle zunächst einen einfachen rekursiven Algorithmus für diese Funktion auf. Wende dann dynamische Programmierung an, um den Algorithmus effizient zu machen. Hier ist noch einmal der Text aus der Aufgabenstellung:

Sei $A \in \mathbb{Z}^n$ als ein Feld $A[0, \dots, n-1]$ gespeichert. Ein Teilfeld von A ist ein genau dann ein *maximales Teilfeld* $A[i..j]$ mit $0 \leq i \leq j \leq n-1$, wenn die Summe $A[i] + A[i+1] + \dots + A[j]$ maximal über alle möglichen Teilfelder ist. Beschreibe und analysiere einen Algorithmus, der mithilfe von dynamischer Programmierung die Summe eines maximalen Teilfelds von A in Zeit $O(n)$ berechnet.

Aufgabe 6 (Palindromische Wege II). Erinnere dich zurück an die Aufgabe zu den palindromischen Wegen (★ in Woche 6). Eine Lösung benutzt eine rekursive Funktion `HASPALINDROMICWALK(u, v, i)`, die feststellt, ob es einen palindromischen Weg von u nach v der Länge i gibt. Ohne auf deine Notizen zu schauen, stelle den rekursiven Algorithmus für diese Funktion erneut auf. Wende dann dynamische Programmierung an, um den Algorithmus effizient zu machen. Hier ist noch einmal der Text aus der Aufgabenstellung:

Ein *Palindrom* über dem Alphabet $\{R, B\}$ ist eine Zeichenkette $s_1, s_2, s_3, \dots, s_n \in \{R, B\}$, sodass $s_i = s_{n-i+1}$ für alle $i \in \{1, \dots, n\}$. Zum Beispiel sind `BRBRB` und `RBBR` Palindrome, aber `RBB` und `BRRR` nicht.

Sei G ein beliebiger gerichteter Graph, in dem jede Kante entweder rot oder blau gefärbt ist, und seien s, t zwei Knoten. Beschreibe einen Algorithmus, der entweder einen Weg von s nach t berechnet, für den die Sequenz von rot und blau entlang der Kanten des Weges ein Palindrom ist, oder korrekterweise feststellt, dass kein solcher Weg existiert.

Zur Erinnerung: Ein Weg von s nach t darf sich selbst überlappen, das heißt, ein Weg muss nicht unbedingt einfach sein.

Aufgabe 7 (RNA). *Ribonucleic acid* (RNA) ist eine lange Kette von Millionen Nukleotiden von vier verschiedenen Typen: Adenin (A), Cytosin (C), Guanin (G), und Uracil (U). Die *Sequenz* eines RNA-Moleküls ist eine Zeichenkette $b[1 \dots n]$, wobei jedes Zeichen $b[u] \in \{A, C, G, U\}$ zu einem Nukleotid korrespondiert. Manchmal mutiert ein Virus und die RNA-Sequenz verändert sich leicht. Deine Aufgabe ist es nun, für zwei gegebene RNA-Sequenzen herauszufinden, wie ähnlich diese sich sind; damit kann man Rückschlüsse ziehen darauf, wie ähnlich sich zwei Virusvarianten sind. Aber nicht jede Änderung in der RNA ist gleich wahrscheinlich! Wir haben nun folgende (fiktive) Ähnlichkeitsmatrix $M[a, b]$ gegeben, die für jede Punktmutation (=Substitution) angibt, wie wahrscheinlich diese ist:

	A	G	C	U
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
U	-4	-3	0	8

Je größer die Zahl ist, desto wahrscheinlicher soll die entsprechende Substitution sein. Zum Beispiel hat eine Substitution von C auf G einen Ähnlichkeitwert von -5, aber ein korrektes Alignment von G mit G hat einen Ähnlichkeitwert von 7. Außerdem definieren wir noch einen Wert $d = -5$, der den Ähnlichkeitwert einer Einfügen oder Löschen Operation auf -5 setzt. Der Ähnlichkeitwert einer Editiersequenz ist die Summe der Ähnlichkeitwerte für jede Spalte.

a) (einfach) Was ist der Ähnlichkeitwert dieser Editiersequenz?

```

G A U
C  C G

```

b) Wie muss das dynamische Programm für die Edit Distance angepasst werden, um eine Editiersequenz zu berechnen, die einen möglichst großen Ähnlichkeitwert erzeugt?