DAP2 – Heimübung 6

Ausgabedatum: 19. 5. 17 — Abgabedatum: Montag 29. 5. für alle Gruppen, 12 Uhr

Schreiben Sie unbedingt immer Ihren **vollständigen Namen**, **Ihre Matrikelnummer** und **Ihre Gruppennummer** auf Ihre Abgaben!

Aufgabe 6.1 (5 Punkte): (Gierige Algorithmen)

Erik ist ein Fan von BVB und will die ℓ Kilometer lange Strecke von Dortmund bis Berlin mit dem Fahrrad fahren, um das Pokalfinale zu sehen. Dazu braucht er mehrere Tage und er hat sicherheitshalber beschlossen, nur tagsüber zu fahren. Täglich kann er k Kilometer fahren, und er darf nur in bekannten Raststätten übernachten. Die i-te Raststätte, $1 \leq i \leq n$, befindet sich am $A[i]$-ten Kilometer ab Dortmund, und es gilt dass $0 < A[i+1] - A[i] \leq k$ ist, für alle $1 \leq i < n$. Erik möchte, dass seine Reise möglichst wenige Übernachtungen enthält.

a) Beschreiben Sie mit eigenen Worten einen **gierigen** Algorithmus, der bei Eingabe des Arrays A eine minimale Menge der Indizes der Raststätte berechnet und zurückgibt, sodass Eriks Reise möglich ist. Geben Sie den Algorithmus auch in Pseudocode an. Für die volle Punktzahl wird ein Algorithmus erwartet, dessen Worst-Case-Laufzeit durch $\mathcal{O}(n)$ beschränkt ist.

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

c) Beweisen Sie, dass Ihr Algorithmus die optimale Lösung zurückgibt.

Lösung:

a) Eine gierige Idee ist, immer so weit tagsüber zu fahren, wie man kann, und zu übernachten, wenn die nächste Raststätte nicht erreicht werden kann. Diese Strategie erweist sich als optimale Strategie.

In Pseudocode erhalten wir folgenden Algorithmus.

```
FahrradFahrt (Array A[1..n], int k, int ℓ):
1. p ← 0                                  /* die letzte Übernachtung bei p km */
2. R ← ∅                                  /* die Menge der Übernachtungen */
3. A[n+1] ← ℓ                             /* Destination als letzte Raststätte */
4. for i ← 1 to n do
5.     if A[i+1] − p > k then
6.         R ← R ∪ {i}
7.         p ← A[i]
8. return R
```

b) Die Laufzeit dieses Algorithmus ist durch die For-Schleife in den Zeilen 4-7 bestimmt, da diese Schleife n Wiederholungen enthält, und jede Wiederholung die konstante Zeit benötigt – zusammen $\mathcal{O}(n)$. Die restliche Anweisungen in den Zeilen 1, 2, 3 und 8 benötigen konstante Zeit. Insgesamt ergibt sich die Laufzeit $\mathcal{O}(n)$.

c) Sei R unsere gierige Lösung und sei Q eine optimale Lösung, sodass $|R| > |Q|$ ist. Wir beweisen die Optimalität unserer gierigen Lösung in zwei Schritten, und zwar:

   i) die gierige Lösung R bleibt immer vor der optimalen Lösung Q, bezüglich der letzten erreichten Raststätte, und

   ii) die optimale Lösung Q kann nicht weniger Übernachtungen als R enthalten, was wird der Optimalität von Q widersprechen.

Hilfsaussage – Behauptung: Seien die Abstände von Dortmund der Raststätten in R $\{A[r_1], \ldots, A[r_s]\}$, und die Abstände der Raststätten in Q: $\{A[q_1], \ldots, A[q_t]\}$, mit $s > t$. Es gilt $A[r_i] \geq A[q_i]$, für alle $1 \leq i \leq t$.

Mit anderen Worten, die Raststätten in R liegen mindestens so weit wie die Raststätten in Q.

Beweis 1: (etwa informal)

Sei i der erste Index, so dass $A[r_i] \neq A[q_i]$ ist (und $A[r_{i-1}] \neq A[q_{i-1}]$). Aufgrund gieriger Auswahl muss gelten, dass $A[r_i] > A[q_i]$ ist (da die gierige Strategie immer so weit geht, wie es möglich ist). Da die optimale Lösung am nächsten Tag die Raststätte auf Kilometer $A[q_{i+1}]$ erreichen wird, mit $A[q_{i+1}] - A[q_i] \leq k$ und $A[r_i] > A[q_i]$, muss auch gelten dass $A[q_{i+1}] - A[r_i] < k$ ist. Damit folgt dass die optimale Lösung auch statt $A[q_i]$ die Raststätte auf $A[r_i]$ verwenden könnte, was widerspricht der Behauptung, dass i der erste unterschiedliche Raststätte ist.

Beweis 2: (Induktiver Beweis)

IA Wenn $i = 1$ ist, muss die erste Übernachtung der gierigen Auswahl $A[r_1]$ mindestens so weit ab Abfahrt sein, wie $A[q_1]$ – damit folgt $A[r_1] \geq A[q_1]$.

IV Es gelte die Behauptung für alle $j < m$: $A[r_j] \geq A[q_j]$.

IS Wir zeigen die Behauptung für $A[r_m]$ und $A[q_m]$. Da $A[r_{m-1}] \geq A[q_{m-1}]$ (nach I.V.), $A[r_m] - A[r_{m-1}] \leq k$ und $A[q_m] - A[q_{m-1}] \leq k$ (nach der Behauptung, dass Q und R zulässige Strategien sind), folgt es dass $A[q_m] - k \leq A[q_{m-1}] \leq A[r_{m-1}]$ ist (aus erster und dritter Ungleichung). D.h. $A[q_m] - A[r_{m-1}] \leq k$ und Erik könnte ab $A[r_{m-1}]$ auch bis $A[q_m]$ an einem Tag fahren. Da er aber bis $A[r_m]$ gierig fährt, muss es gelten dass $A[r_m]$ mindestens so weit von $A[r_{m-1}]$ wie $A[q_m]$ ist, d.h. $A[r_m] \geq A[q_m]$. Somit gilt die Behauptung für alle natürlichen Zahlen m.

Widerspruchbeweis der zweiten Aussage: Da $t < s$, muss es gelten dass für die Endstelle $A[n+1]$ gilt $A[n+1] - A[r_t] > k$ (ansonsten brauchen wir nicht die Übernachtungen $t+1, t+2, \ldots, s$).

Aus bewiesener Behauptung für $m = t$ folgt es, dass es auch $A[r_t] \geq A[q_t]$ gilt, für die letzte ausgewählte Raststätte in Q. Da von $A[q_t]$ innerhalb eines Tages die Endstelle $A[n+1]$ erreichbar ist, gilt es $A[n+1] - A[q_t] \leq k$. Davon folgt aber auch $A[n+1] - A[r_t] \leq k$. Das widerspricht der Behauptung, dass $t < s$ ist. Somit kann dies nicht gelten, und die Optimalität unserer gierigen Lösung ist bewiesen.

Aufgabe 6.2 (5 Punkte): (Gierige Algorithmen)

Simon sammelt Münzen, und hat über eine neue Münzserie von n unterschiedlichen Silbermünzen erfahren, die er kaufen möchte und die im Folgenden mit den Zahlen 1 bis n identifiziert werden. Allerdings kann er sich nur leisten, pro Monat eine Münze im numismatischen Laden zu kaufen.

Zu Beginn hat jede Münze einen Preis von 20€. Jeden Monat steigt der Preis jeder Münze i, $1 \leq i \leq n$, um einen bestimmten Faktor $p_i > 1$. Der neue Preis lässt sich aus dem vorherigen Preis durch Multiplikation mit Faktor p_i berechnen. Das heißt, nach m Monaten kostet die Münze i nun $20 \cdot p_i^{m-1}$. Simon muss sich entscheiden, in welcher Reihenfolge er die Münzen kaufen sollte, damit er den ausgegebenen Gesamtgeldbetrag minimiert.

a) Beschreiben Sie mit eigenen Worten einen **gierigen** Algorithmus, der bei Eingabe der Faktoren $p_1, \ldots, p_n$ den minimalen Gesamtgeldbetrag, den Simon ausgeben muss um alle n Münzen zu kaufen, berechnet und zurückgibt. Geben Sie den Algorithmus auch in Pseudocode an. Für die volle Punktzahl wird ein Algorithmus erwartet, dessen Worst-Case-Laufzeit durch $\mathcal{O}(n \cdot \log n)$ beschränkt ist.

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

c) Beweisen Sie, dass Ihr Algorithmus eine optimale Lösung berechnet.

Lösung:

a) Um den Geldbetrag zu minimieren, macht es Sinn zuerst die Münzen mit schnellem Preiswachstum zu kaufen (diejenige, deren Preis am schnellsten wächt). Deswegen sortieren wir die Münzen absteigend nach Preis, und kaufen die Münzen in dieser Reihenfolge, wobei i-te Münze für $20 \cdot p_i^{i-1}$ €. Den auszugebende Betrag ergibt es als Summe der Einzelmünzen.

In Pseudocode erhalten wir folgenden Algorithmus.

```
Muenzen (Array p[1..n]):
1. Sortiere P absteigend mit Mergesort                    /* in O(n log n) Zeit */
2. S ← 0
3. for i ← 1 to n do
4.     S ← S + 20 · p[i]^{i−1}      /* angenommen arithmetische Operationen in linearer
       Zeit */
5. return S
```

b) Die Laufzeit $T(n)$ des Algorithmus ist durch die Zeit für Sortieren (Zeile 1) dominiert – $O(n \log n)$. Die Zeilen 2 und 5 benötigen konstante Zeit, und die Schleife in der Zeilen 3 und 4 benötigt lineare Zeit – $O(n)$. Insgesamt benötigt der Algorithmus die Zeit $O(n \log n)$.

c) Um zu beweisen, dass der Betrag $S = 20 \cdot \sum_{i=1}^{n} p_i^{i-1}$, der vom Algorithmus `Muenzen` zurückgegeben wird, ist optimal, nehmen wir an, dass es eine optimale Summe Q existiert, sodass $Q < S$ ist, und dass mit dem Betrag Q alle Münzen gekauft werden können. Aus gieriger Auswahl folgt, dass $p_1 > p_2 > \ldots > p_n$.

Seien die Faktoren der Münzen in optimaler Lösung Q: $q_1, \ldots, q_n$. Dabei ist $(q_1, \ldots, q_n)$ eine Permutation von $(p_1, \ldots, p_n)$ und es gilt $Q = 20 \cdot \sum_{i=1}^{n} q_i^{i-1}$. Eine Ordnungsbehauptung über Reihenfolge der Faktoren in Q dürfen wir nicht machen.

Sei i der erste Index, wo sich die Permutationen unterscheiden: $p_i \neq q_i$. Wegen gieriger Auswahl gilt $p_i > q_i$. Dann muss der Faktor p_i irgendwo später in der Permutation die Q erzeugt, auftauchen, und es gilt $q_j = p_i$ mit $j > i$. Für p_j (die Münze, die im Monat j in gieriger Lösung gekauft wird) gilt $p_j < p_i$ wegen gieriger Auswahl.

Erstellen wir jetzt aus der Permutation T: $(q_1, \ldots, q_i, \ldots, q_j, \ldots, q_n)$ die die Summe Q ergibt, die Permutation T': $(q_1, \ldots, q_{i-1}, p_i, q_{i+1}, \ldots, q_{j-1}, q_j, q_{j+1}, \ldots, q_n)$. Sei der Geldbetrag, impliziert von der Permutation T', mit Q^* bezeichnet. Es gilt dass

$$\begin{aligned} Q - Q^* &= 20 \cdot \left[q_i^{i-1} + q_j^{j-1} - p_i^{i-1} - p_j^{j-1}\right] \\ &= 20 \cdot \left[p_j^{i-1} + p_i^{j-1} - p_i^{i-1} - p_j^{j-1}\right] \\ &= 20 \cdot \left[p_j^{i-1} \cdot \left(1 - p_j^{j-i}\right) + p_i^{i-1} \cdot \left(p_i^{j-i} - 1\right)\right] \end{aligned}$$

Zu zeigen dass $Q - Q^* > 0$ ist, ist äquivalent zu

$$p_i^{i-1} \cdot \left(p_i^{j-i} - 1\right) > p_j^{i-1} \cdot \left(p_j^{j-i} - 1\right).$$

Diese Ungleichung ist korrekt, weil $p_i > p_j > 1$ ist, und somit auch $p_i^{i-1} > p_j^{i-1}$ und $p_i^{j-i} - 1 > p_j^{j-i} - 1$. Damit wäre Q^* ein noch kleiner Geldbetrag, mit dem man alle Münzen kaufen könnte, was widerspricht der Optimalität von Q. Somit muss unsere Lösung S optimal sein. □