Übungen zur Vorlesung
Einführung in die Algorithmik WS 15/16

Moritz Eissenhauer

Hasso-Plattner-Institut
Potsdam
4. Januar 2016

Lösung zu Übungsblatt 9

Lösung zu Aufgabe 1

a)

ZZ. Es gibt Instanzen auf denen schlechter-tankplan 3/2 mal so viele Stops einplant wie nötig.

Beweis. Sei $ST(d, T)$ die Anzahl der Stops die von schlechter-tankplan eingeplant werden. Sei $OPT(d, T)$ die optimale Lösung.
Sei $d = 4; T = 0, 3, 4, x, 6; z = 8$ mit $4 < x < 6$ beliebig, fest.
Schlechter-tankplan plant die Stops $0, 3, 6$ ein $\Rightarrow ST(d, T) = 3$. Es gibt die Lösung mit den Stops $0, 4$. Daraus können wir schließen, dass $OPT(d, T) \leq 2$. Schlechter-tankplan plant also mindestens 3/2 mal so viele Stops ein wie die Optimale lösung auf allen Instanzen T. □

b)

Algorithmus:

Tankplan

Input: Tankstellen T, Reichweite d, Ziel z
Output: Tankplan P

füge $T[0]$ zu P hinzu
laststop $\leftarrow T[0]$
entferne $T[0]$ (Indizes rutschen nach)
while *laststop* $+ d < z$ **do**
  candidate $\leftarrow T[0]$
  entferne $T[0]$
  if T leer $\vee$ *laststop* $+ d < T[0]$ **then**
    füge *candidate* zu P hinzu
    laststop $\leftarrow$ *candidate*
  end if
end while
return P

Laufzeit:
Die **while**-Schleife entfernt in jedem Durchgang ein Element aus T und terminiert (durch die Bedingungen aus der Aufgabe) spätestens wenn T leer ist. Da alle Opperationen innerhalb und auserhalb der Schleife $\in \mathcal{O}(1)$ sind ist die Laufzeit damit $\mathcal{O}(n)$.

Korrektheit:
ZZ. Zwei aufeinanderfolgende Stops liegen nie weiter als d auseinander.

Beweis. durch Widerspruch:
Seien p_i, p_{i+1} aufeinanderfolgende Stops in P mit $p_i + d < p_{i+1}$.
Es muss einen Zeitpunkt gegeben haben in dem p_i zu P hinzugefügt wurde. Das heißt, dass dannach $laststop = p_i$ galt. Damit das nächste Element in P p_{i+1} ist, muss $laststop = p_i$ und $candidate = pi + 1$ gelten. Laut Aufgabe gibt es aber in T mindetens ein Element $t > p_i$ vor p_{i+1}, für das gilt $p_i + d \leq t$. Insbesondere gibt es dann ein größtes Element $t \leq t_j < p_{i+1}$ für das $p_i + d \leq t_j$ noch gilt. Für t_{j+1} gilt dann $p_i + d > t_{j+1}$.
Nachdem p_i hinzugefügt wurde, und bevor $p_{i+1} = candidate$ gilt, muss $t_j = candidate$ gewesen sein. Dann ist aber auch $T[0] = t_{j+1}$, und die **if**-Bedingung ist wahr. Das heißt $t_j \neq p_{i+1}$ folgt auf p_i. Das ist aber ein Widerspruch zur Annahme.
Da die Annahme zum Widerspruch führt kann es keine p_i, p_{i+1} geben für die $p_i + d < p_{i+1}$ gilt. □

ZZ. Die Anzahl der Stops ist minimal.

Beweis. Sei P die Lösung meines Algorithmus. Sei OPT eine Lösung mit minimaler Anzahl von Stops. Sei $A(X, z')$ für eine Lösung X und $z' \leq z$ die Anzahl der Stops t für die gilt $t \leq z'$.
Behauptung: Für alle p_i in P gilt $A(P, p_i) \leq A(OPT, p_i)$.
Zeige Behauptung: Induktiv:
IA: $i = 0$: $p_0 = 0 \Rightarrow A(P, 0) = 1$. In OPT muss auch 0 enthalten sein $\Rightarrow A(OPT, 0) \geq 1$.
IV: Behauptung gilt für ein i beliebig.
IS: $A(P, p_{i+1}) = A(P, i) + 1$, da p_{i+1} der nächste Stop nach p_i in P ist. Es gibt keine Tankstelle $t > p_{i+1}$ die noch von p_i aus erreichbar wäre, sonst hätte mein Algorithmus diese gewählt. Insbesondere ist auch keine Tankstelle $t > p_{i+1}$ von $t' \leq p_i$ aus erreichbar. Wenn OPT bei t' stopt, dann muss es noch einen Stop in OPT vor oder bei p_{i+1} geben. Also gilt $A(OPT, p_{i+1}) \geq A(OPT, p_i) + 1$. Da $A(OPT, p_i) \leq A(P, p_i)$ gilt (nach IV), gilt auch $A(OPT, p_{i+1}) \leq A(P, p_{i+1})$.
Mit IA, IV und IS gilt die Behauptung.
Die Behauptung gilt insbesondere auch für den letzten Stop p_k in P. $A(P, p_k)$ ist aber auch die Gesamtzahl aller Stops in P. Die Gesamtzahl aller Stops in OPT ist also mindestens so groß wie die Gesamtzahl aller Stops in P. P ist damit eine minimale Lösung. □

Aus den beiden Teilbeweisen folgt, dass der Algorithmus korrekt ist.

Lösung zu Aufgabe 2

-

Lösung zu Aufgabe 3
Algorithmus:

Tunnel

Input: Liste der X-Koordinaten X, Liste der Y-Koordinaten Y
Output: Y-Koordinate des Tunnels y

1: $y \leftarrow$ Median von Y
2: **return** y

Dabei sei der Median das $\lfloor \frac{n}{2} \rfloor$ kleinste Element.

Laufzeit:
In der Vorlesung haben wir einen Algorithmus besprochen, der den Median in $\mathcal{O}(n)$ bestimmt. Wenn man diesen für die Bestimmung des Median wählt, dann ist die Laufzeit von *Tunnel* in $\mathcal{O}(n)$.

Korrektheit:
ZZ. y ist eine optimale Lösung für die Y-Koordinate des Tunnels.

Beweis. Die Länge der Tunnel bleibt gleich, unabhängig von der Wahl der y-Koordinate.
Seien $Y = \{y_1, y_2, ..., y_n\}$ die Y-Koordinaten. Die Länge aller Nord-Süd Tunnel für Lösung y ist:

$$T = \sum_{i=1}^{n} |y_i - y| \tag{1}$$

Sei $y' = y + d$ mit $d \neq 0$ beliebig eine Alternativlösung.
Fall 1 $d > 0$:
Alle Tunnel von Lagerhallen mit Y-Koordinate $y_l \leq y$ werden um d länger. Davon gibt es $\lfloor \frac{n}{2} \rfloor$. Alle anderen Tunnel werden höchstens um d kürzer. Davon gibt es höchstens $\lfloor \frac{n}{2} \rfloor$. Die Tunnellänge der Alternativlösung ist also $T' \geq T + d\lfloor \frac{n}{2} \rfloor - d\lfloor \frac{n}{2} \rfloor = T$. Das heißt, die Gesamtlänge aller Tunnel bleibt bestenfalls gleich wenn der West-Ost Tunnel nach Norden verschoben wird.
Fall 2 $d < 0$:
Analog zu Fall 1, nur das die nördlichen Tunnel länger werden, und die Südlichen höchstens kürzer. Es werden aber wieder höchstens genausoviele kürzer wie länger, auch diese Lösungen sind also bestenfalls genauso gut.
Da alle Lösungen y' höchstens genauso gut sind, kann es keine bessere Lösung geben. y ist also optimal. □

Lösung zu Aufgabe 4
Algorithmus:
Annahme: es gibt mindestens ein Geschenk.

Geschenkwahl
Input: Liste der Geschenke A
Output: Liste der nicht verdrängten Geschenke

```
1: A ← A sortiert nach Preis (aufsteigend)
2: result ← leere Liste
3: bestK ← k(A[0])
4: result.append(A[0])
5: for i = 1 to len(A) do
6:    if k(A[i]) < bestK then
7:       bestK ← k(A[i])
8:       result.append(A[i])
9:    end if
10: end for
11: return result
```

Laufzeit:
Das Sortieren in Zeile 1 hat eine Laufzeit von $\mathcal{O}(n \log n)$. Zeilen 2, 3, 4, 6, 7, 8 haben jeweils eine konstante Laufzeit. Die Schleife läuft n mal und hat damit eine Laufzeit von $\mathcal{O}(n)$. Die Gesamtlaufzeit ist damit $\mathcal{O}(n \log n)$.
Korrektheit:
ZZ. Keines der gewählten Geschenke wird von einem anderen verdrängt.

Beweis. Induktiv.
IA: Das erste Geschenk der Lösung wird von keinem der anderen Geschenke verdrängt, da es den kleinsten Preis hat.
IV: Das erste bis i-te Geschenk der Lösung wird von keinem anderen Geschenk verdrängt. für ein beliebiges i.
IS: Falls das Geschenk $i+1$ in der Lösung existiert, dann wurde es nur genau dann hinzugefügt, wenn es einen kleineren Kitschfaktor als das i-te Geschenk der Lösung hat. Damit hat es auch einen kleineren Kitschfaktor als alle Geschenke, die links davon in der sortierten Liste A stehen, kann also von keinem dieser verdrängt werden. Da A nach Preis sortiert ist, hat es auch einen niedrigeren Preis als alle Geschenke die rechts stehen und kann von keinem dieser verdrängt werden. Das $i+1$-te Geschenk der Lösung wird also nicht verdrängt.
Aus IA, IV und IS folgt, dass kein gewähltes Geschenk von einem anderen verdrängt wird. □

ZZ. Alle nicht gewählten Geschenke werden von mindestens einem der gewählten verdrängt.

Beweis. Ein Geschenk wird genau dann nicht gewählt, wenn es einen höheren Kitschfaktor als das zuletzt gewählte Geschenk hat. Da es in der sortierten Liste A nach dem zuletzt gewählten Geschenk kommt, hat es auch einen höheren Preis. Es wird also mindestens von dem zuletzt gewählten Geschenk verdrängt. □

Aus den beiden Teilbeweisen folgt, dass der Algorithmus korrekt ist.