Algorithmenentwurf HA 6

Lukas Brandt: 7011823, Clemens Damke: 7011488, Lukas Giesel: 7011495
26. Mai 2016

Aufgabe 9

a. Rekursive Definition

$$\tau(i,j) := \mathbb{N}_0 \cap \begin{cases} [i,j] & \textbf{if } i \leq j \\ [0,n-1] \setminus]j,i[& \textbf{if } i > j \end{cases}$$

$$t[i,j] := \begin{cases} 0 & \textbf{if } \min\{\, q \mid j \equiv i+q \bmod n \,\} \leq 1 \\ \min\{\, t[i,k] + \Delta(i,k,j) + t[k,j] \mid k \in \tau((i+1 \bmod n),(j-1 \bmod n)) \,\} & \textbf{else} \end{cases}$$

b. Algorithmus

Algorithm 1 Minimales Gewicht Triangulierung Problem

1: **function** MinimalesGewichtTriangulierung($\langle v_0, \ldots, v_{n-1} \rangle$)
2: **if** $n \leq 2$ **then return** 0 ▷ $\mathcal{O}(1)$

3: **for** $i \leftarrow 0$ **to** $n-1$ **do** ▷ $\mathcal{O}(n)$
4: $t[i,(i+1 \bmod n)] \leftarrow 0$

5: **for** $l \leftarrow 2$ **to** $n-1$ **do** ▷ $\mathcal{O}(n)$
6: **for** $i \leftarrow 0$ **to** $\begin{cases} \textbf{if } l \neq n-1\text{:} & n-1 \\ \textbf{else:} & 0 \end{cases}$ **do** ▷ $\mathcal{O}(n) \cdot \mathcal{O}(n) = \mathcal{O}(n^2)$
7: $m \leftarrow \infty$
8: $j \leftarrow i + l \mod n$
9: **for** $d \leftarrow 1$ **to** $l-1$ **do** ▷ $\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$
10: $k \leftarrow i + d \mod n$
11: $m_0 \leftarrow t[i,k] + |v_i v_k| + |v_k v_j| + |v_j v_i| + t[k,j]$
12: **if** $m_0 < m$ **then** $m \leftarrow m_0$
13: $t[i,j] \leftarrow m$

14: **return** $t[0,n-1]$

c. Laufzeit

Der Algorithmus besteht im Wesentlichen aus drei verschachtelten Schleifen, die alle bis zu n mal ausgeführt werden. Die Operationen in den Schleifen benötigen alle nur konstante Zeit. Insgesamt ergibt sich also eine Laufzeit von $\mathcal{O}(n^3)$.

Aufgabe 10

a. Rekursive Definition

$$D := \{d_1, \ldots, d_n\}$$

$$t[i,j] := \begin{cases} 0 & \textbf{if } i > \max D \\ t[i+1, \max\{j-1, 0\}] & \textbf{if } i \leq \max D \wedge (j > 0 \vee i \notin D) \\ \min\{\, c_r + t[i+1, \ell_r - 1] \mid r \in \{1, \ldots, k\} \,\} & \textbf{else} \end{cases}$$

Optimale Lösung ($\hat{=}$ minimale Gesamtausgaben) ist $t[0,0]$.

b. Algorithmus

Algorithm 2 Parkschein Problem

Require: $D = \{d_1, \ldots, d_n\}$
Require: $L = \{\ell_1, \ldots, \ell_k\}$
Require: $C = \{c_1, \ldots, c_k\}$

```
1:  function PARKSCHEIN(D, L, C)
2:      m_d ← max D                                   ▷ O(|D|) = O(n)
3:      m_ℓ ← max L                                   ▷ O(|L|) = O(k)

4:      for j ← 0 to m_ℓ − 1 do                       ▷ O(m_ℓ)
5:          t[m_d + 1, j] ← 0
6:      for i ← m_d to 0 do                           ▷ O(m_d)
7:          for j ← 1 to m_ℓ − 1 do                   ▷ O(m_d · m_l)
8:              t[i, j] ← t[i + 1, j − 1]
9:          if i ∈ D then                             ▷ O(m_d · 1) mit D als Hashset
10:             m ← ∞
11:             for r ← 1 to k do                     ▷ O(m_d · k)
12:                 m_0 ← c_r + t[i + 1, ℓ_r − 1]
13:                 if m_0 < m then m ← m_0
14:             t[i, 0] ← m
15:         else
16:             t[i, 0] ← t[i + 1, 0]

17:     return t[0, 0]
```

c. Laufzeit

Der Algorithmus ist pseudopolynomiell, da $m_d = \mathcal{O}(2^n)$ und $m_\ell = \mathcal{O}(2^k)$.
Wegen Zeile 7 ergibt sich also insgesamt $\mathcal{O}(\max D \cdot \max L) = \mathcal{O}(2^{n+k})$.

Aufgabe 11

a. Rekursive Definition

$$p[i,s,r] := \begin{cases} d(\sigma_s,\sigma_r)\cdot D & \textbf{if } i > t \\ \min\{p[i+1,s,r] + d(\sigma_s,\sigma_i), & \textbf{else} \\ \qquad p[i+1,i,r] + d(\sigma_s,\sigma_i)\cdot D, & \\ \qquad p[i+1,s,i] + d(\sigma_i,\sigma_r)\cdot D\} & \end{cases}$$

Optimale Lösung ($\hat{=}$ minimale Kosten) ist $\min\{\, p[1,1,r] \mid r \in \{1,\ldots,n\}\,\}$.

b. Algorithmus

Algorithm 3 Page Migration Problem

```
1:  function PageMigration(⟨ν_1, ..., ν_n⟩, ⟨σ_1, ..., σ_t⟩, D, d)
2:      for s ← 1 to n do                                          ▷ O(n)
3:          for r ← 1 to n do                                      ▷ O(n^2)
4:              p[t+1, s, r] ← d(σ_s, σ_r) · D                     ▷ O(n^2 · 1)

5:      for i ← t to 1 do                                          ▷ O(t)
6:          for s ← 1 to n do                                      ▷ O(t · n)
7:              for r ← 1 to n do                                  ▷ O(t · n^2)
8:                  p_0 ← p[i+1, s, r] + d(σ_s, σ_i)
9:                  p_1 ← p[i+1, i, r] + d(σ_s, σ_i) · D
10:                 p_2 ← p[i+1, s, i] + d(σ_i, σ_r) · D
11:                 p[i, s, r] ← min{p_0, p_1, p_2}                ▷ O(t · n^2 · 1)

12:     m ← ∞
13:     for r ← 1 to n do                                          ▷ O(n)
14:         if p[1, 1, r] < m then m ← p[1, 1, r]

15:     return o
```

c. Laufzeit

Der Algorithmus besteht im Wesentlichen aus drei verschachtelten Schleifen, die jeweils t, n und n mal ausgeführt werden. Die Operationen in den Schleifen benötigen alle nur konstante Zeit. Insgesamt ergibt sich also eine Laufzeit von $\mathcal{O}(t \cdot n^2)$.