

Algorithmenentwurf HA 6

LUKAS BRANDT: 7011823, CLEMENS DAMKE: 7011488, LUKAS GIESEL: 7011495
26. MAI 2016

Aufgabe 9

a. Rekursive Definition

$$\tau(i, j) := \mathbb{N}_0 \cap \begin{cases} [i, j] & \text{if } i \leq j \\ [0, n-1] \setminus]j, i[& \text{if } i > j \end{cases}$$
$$t[i, j] := \begin{cases} 0 & \text{if } \min\{q \mid j \equiv i + q \pmod{n}\} \leq 1 \\ \min\{t[i, k] + \Delta(i, k, j) + t[k, j] \mid k \in \tau((i+1 \pmod{n}), (j-1 \pmod{n}))\} & \text{else} \end{cases}$$

b. Algorithmus

Algorithm 1 Minimales Gewicht Triangulierung Problem

```
1: function MINIMALESGEWICHTTRIANGULIERUNG( $\langle v_0, \dots, v_{n-1} \rangle$ )
2:   if  $n \leq 2$  then return 0  $\triangleright \mathcal{O}(1)$ 

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

5:   for  $l \leftarrow 2$  to  $n-1$  do  $\triangleright \mathcal{O}(n)$ 
6:     for  $i \leftarrow 0$  to  $\begin{cases} \text{if } l \neq n-1: & n-1 \\ \text{else:} & 0 \end{cases}$  do  $\triangleright \mathcal{O}(n) \cdot \mathcal{O}(n) = \mathcal{O}(n^2)$ 
7:        $m \leftarrow \infty$ 
8:        $j \leftarrow i + l \pmod{n}$ 
9:       for  $d \leftarrow 1$  to  $l-1$  do  $\triangleright \mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$ 
10:         $k \leftarrow i + d \pmod{n}$ 
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

$$D := \{d_1, \dots, d_n\}$$
$$t[i, j] := \begin{cases} 0 & \text{if } i > \max D \\ t[i+1, \max\{j-1, 0\}] & \text{if } i \leq \max D \wedge (j > 0 \vee i \notin D) \\ \min\{c_r + t[i+1, \ell_r - 1] \mid r \in \{1, \dots, k\}\} & \text{else} \end{cases}$$

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

b. Algorithmus

Algorithm 2 Parkschein Problem

Require: $D = \{d_1, \dots, d_n\}$

Require: $L = \{\ell_1, \dots, \ell_k\}$

Require: $C = \{c_1, \dots, c_k\}$

```
1: function PARKSCHEIN( $D, L, C$ )
2:    $m_d \leftarrow \max D$   $\triangleright \mathcal{O}(|D|) = \mathcal{O}(n)$ 
3:    $m_\ell \leftarrow \max L$   $\triangleright \mathcal{O}(|L|) = \mathcal{O}(k)$ 

4:   for  $j \leftarrow 0$  to  $m_\ell - 1$  do  $\triangleright \mathcal{O}(m_\ell)$ 
5:      $t[m_d + 1, j] \leftarrow 0$ 

6:   for  $i \leftarrow m_d$  to  $0$  do  $\triangleright \mathcal{O}(m_d)$ 
7:     for  $j \leftarrow 1$  to  $m_\ell - 1$  do  $\triangleright \mathcal{O}(m_d \cdot m_\ell)$ 
8:        $t[i, j] \leftarrow t[i+1, j-1]$ 

9:     if  $i \in D$  then  $\triangleright \mathcal{O}(m_d \cdot 1)$  mit  $D$  als Hashset
10:       $m \leftarrow \infty$ 
11:      for  $r \leftarrow 1$  to  $k$  do  $\triangleright \mathcal{O}(m_d \cdot k)$ 
12:         $m_0 \leftarrow c_r + t[i+1, \ell_r - 1]$ 
13:        if  $m_0 < m$  then  $m \leftarrow m_0$ 
14:       $t[i, 0] \leftarrow m$ 
15:    else
16:       $t[i, 0] \leftarrow t[i+1, 0]$ 

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

$$\pi_i := j, \text{ sodass } \sigma_i = \nu_j$$
$$p[i, s] := \begin{cases} 0 & \text{if } i > t \\ \min\{d(\nu_s, \sigma_i) + p[i+1, s], D \cdot d(\nu_s, \sigma_i) + p[i+1, \pi_i]\} & \text{else} \end{cases}$$

Optimale Lösung ($\hat{=}$ minimale Kosten) ist $p[1, 1]$.

b. Algorithmus

Algorithm 3 Page Migration Problem

```
1: function PAGEMIGRATION( $\langle \nu_1, \dots, \nu_n \rangle, \langle \sigma_1, \dots, \sigma_t \rangle, D, d$ )
2:   for  $s \leftarrow 1$  to  $n$  do  $\triangleright \mathcal{O}(n)$ 
3:      $p[t+1, s] \leftarrow 0$ 

4:   for  $i \leftarrow t$  to  $1$  do  $\triangleright \mathcal{O}(t)$ 
5:     for  $j \leftarrow 1$  to  $n$  do  $\triangleright \mathcal{O}(t \cdot n)$ 
6:       if  $\sigma_i = \nu_j$  then
7:          $\pi_i \leftarrow j$ 
8:         break

9:   for  $s \leftarrow 1$  to  $n$  do  $\triangleright \mathcal{O}(t \cdot n)$ 
10:     $p_0 \leftarrow d(\nu_s, \sigma_i) + p[i+1, s]$ 
11:     $p_1 \leftarrow D \cdot d(\nu_s, \sigma_i) + p[i+1, \pi_i]$ 
12:     $p[i, s] \leftarrow \min\{p_0, p_1\}$   $\triangleright \mathcal{O}(t \cdot n \cdot 1)$ 

13: return  $p[1, 1]$ 
```

c. Laufzeit

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