

3. Übungsblatt

Jasper Gude Pia Röttgers

7. Mai 2026

/ 25

Aufgabe 1 – Triangulierungen und Dynamische Programmierung

a)

$$T(P) = \begin{cases} 0 & \text{falls } P \text{ nur 2 Ecken hat} \\ \min_{i \in \{2, \dots, n-1\}} [T(p_{1,i}) + T(p_{i,n}) + \text{Diagonalkosten}(i)] & \text{sonst} \end{cases}$$

$$\text{Diagonalkosten}(i) = \begin{cases} d(p_2, p_n) & \text{falls } i = 2 \\ d(p_1, p_{n-1}) & \text{falls } i = n - 1 \\ d(p_1, p_i) + d(p_i, p_n) & \text{sonst} \end{cases}$$

Die Kosten der minimalen Triangulierung $T(P)$ lassen sich aus der minimalsten Summe der kostenminimalsten Triangulierung der entstehenden Teilpolygone $T(p_{1,i})$ und $T(p_{i,n})$ und den entstehenden Diagonalkosten berechnen.

Die Diagonalkosten unterscheiden sich, je nachdem ob eine oder zwei der drei Kanten Polygonkanten sind. Sobald ein (Teil-)Polygon nur noch zwei Ecken hat bricht die Rekursion ab.

/ 2

b) Man nutzt die Idee aus a.

Tabelle $A[i, j]$ speichert die kostenminimalen Triangulierung des Teilpolygons mit den Ecken p_i, \dots, p_j

Für alle i gilt $A[i, i + 1] = 0$, da diese Teilpolygone mit nur zwei Ecken darstellen, welche keine Triangulierung benötigen.

Für $j > i + 1$ gilt:

$$A[i, j] = \min_{k \in \{i+1, \dots, j-1\}} [A[i, k] + A[k, j] + \text{Diagonalkosten}(i, k, j)]$$

$$\text{Diagonalkosten}(i, k, j) = \begin{cases} d(p_k, p_j) & \text{falls } k = i + 1 \\ d(p_i, p_k) & \text{falls } k = j - 1 \\ d(p_i, p_k) + d(p_k, p_j) & \text{sonst} \end{cases}$$

Man berechnet dabei die Einträge nach nach wachsendem Abstand $r = j - i$, also von $r = 2$ bis $r = n - 1$. Dadurch sind die Einträge $A[i, k]$ und $A[k, j]$ immer bereits berechnet wenn man den $A[i, j]$ benötigt, da $k - i < r$ und $j - k < r$ gilt.

Das Ergebnis steht dann in $A[1, n]$.

/ 4

- c) Die polynomielle Laufzeit kann über die Schleifen des DP begründet werden.

Äußere Schleife: r läuft von 1 bis $n - 1$ benötigt also $\mathcal{O}(n)$ Schritte

Mittlere Schleife: für jedes r gibt es $\mathcal{O}(n)$ Paare (i, j) bei welchen $j - i = r$ gilt.

Innere Schleife: Für jedes Paar probiert man alle k zwischen i und j also $\mathcal{O}(n)$ Werte.

Pro Iteration der inneren Schleife benötigt man $\mathcal{O}(1)$ um die Einträge zu berechnen.

Daraus folgt:

$$T(n) = \mathcal{O}(n) \cdot \mathcal{O}(n) \cdot \mathcal{O}(n) \cdot \mathcal{O}(1) = \mathcal{O}(n^3)$$

Diese Schranke ist auch scharf, da jeder der $\mathcal{O}(n^2)$ Tabelleneinträgen auch tatsächlich befüllt wird und für jeden der Einträge im Durchschnitt $\mathcal{O}(n)$ Werte von k durchprobiert werden. Es gilt also $\Theta(n^3)$.

Der Speicherbedarf des Programms ist $\mathcal{O}(n^2)$, da die Tabelle so viele Einträge für die Paare (i, j) mit $1 \leq i < j \leq n$ hat.

/ 2

Aufgabe 2 – TSP mit Wiederholungen

a)

b)

Aufgabe 3 – Metrisches TSP

a)

b)

Aufgabe 4 – Längste Wege

- a)
- b)
- c)