Universität Bremen

FB3: Mathematik/Informatik

Prof. Dr. Daniel Neuen
Jens Schlöter

Wintersemester 2023/2024

Algorithmentheorie

Präsenzübung 3

Präsenzübung 3.1

Beweist folgende Sätze aus der Vorlesung.

Satz (Handshake Lemma). In einem einfachen ungerichteten Graphen $G = (V, E)$ gilt

$$\sum_{v \in V} \mathrm{d}(v) = 2|E|.$$

Satz. In einem einfachen ungerichteten Graphen ist die Anzahl der Knoten mit ungeradem Grad gerade.

Präsenzübung 3.2

Gebt einen Algorithmus an, der für einen gegebenen ungerichteten Graphen $G = (V, E)$ entscheidet, ob dieser kreisfrei ist. Die Laufzeit des Algorithmus soll linear in der Eingabegröße des Graphen ($\mathcal{O}(|V| + |E|)$) sein. Zeigt, dass Euer Algorithmus korrekt ist und die gewünschte Laufzeit erzielt.

Präsenzübung 3.3

(a) Gegeben seien natürliche Zahlen $x_1, x_2, \ldots, x_n \in [0, M]$ für ein ebenfalls gegebenes $M \in \mathbb{N}$. Wir suchen eine Teilmenge $S \subseteq [0, M]$ mit kleinstmöglicher Kardinalität, so dass für jedes $i \in \{1, \ldots, n\}$ ein $s \in S$ existiert mit $|s - x_i| \leq 5$. Gebt einen Algorithmus an, der dieses Problem in Zeit $\mathcal{O}(n \cdot \log n)$ löst. Zeigt, dass euer Algorithmus korrekt ist und die gewünschte Laufzeit erzielt.

(b) Gegeben sei eine Menge von natürlichen Zahlen $X \subseteq [0, M]$ für ein ebenfalls gegebenes $M \in \mathbb{N}$. Wir suchen eine Teilmenge $S \subseteq X$ mit kleinstmöglicher Kardinalität, so dass die folgenden Eigenschaften gelten:

   (i) Es gibt ein $s \in S$ mit $s \leq 20$.

   (ii) Es gibt ein $s \in S$ mit $M - s \leq 20$.

   (iii) Seien $s_1, \ldots, s_{|S|}$ die Zahlen in S in aufsteigender Sortierung. Dann gilt $s_i - s_{i-1} \leq 20$ für alle $i \in \{2, \ldots, |S|\}$.

   Gebt einen Algorithmus an, der dieses Problem in Zeit $\mathcal{O}(n \cdot \log n)$ löst. Zeigt, dass euer Algorithmus korrekt ist und die gewünschte Laufzeit erzielt.

Lösung:

(a) **Greedy Algorithmus:**

Algorithmus 1 : Algorithm

```
1 S ← ∅;
2 current ← −∞;
3 sort x_1, ..., x_n (aufsteigend);
4 for i=1 to n do
5 |   if current ∉ [x_i − 5, x_i + 5] then
6 |   |   current ← min{x_i + 5, M};
7 |   |   S ← S ∪ {current};
8 return S;
```

Laufzeit: Abgesehen vom Sortieren in Zeile 3 haben die elementaren Operationen des Algorithmus Laufzeit $\mathcal{O}(1)$. Die Gesamtlaufzeit wird also entweder von der Laufzeit des Sortierens ($\mathcal{O}(n \log n)$) oder der Anzahl der Schleifendurchläufe dominiert ($\mathcal{O}(n)$). Insgesamt ergibt sich also eine Laufzeit von $\mathcal{O}(n \log n)$.

Terminierung: Der Algorithmus terminiert nach n Iterationen der for-Schleife.

Korrektheit: Sei S die Lösung des Algorithmus. Per Definition von Zeile 6 und 7 gilt $s \in [0, M]$ für alle $s \in S$. Wir müssen argumentieren, dass es für jedes x_i mit $i \in \{1, \ldots, n\}$ ein $s \in S$ gibt mit $|x_i - s| \leq 5$. Wir sagen ein $s \in S$ *überdeckt* ein x_i wenn $|x_i - s| \leq 5$.

- Für jedes x_i wird in Zeile 5 direkt überprüft, ob es vom zuletzt ausgewählten Element überdeckt wird. Falls ja, dann gibt es ein $s \in S$ das x_i überdeckt.
- Falls nicht, wird $s = \min\{x_i + 5, M\}$ in Zeile 6 zu S hinzugefügt. Dann gibt es also auch ein $s \in S$, das x_i überdeckt.
- Da die Schleife über alle x_i iteriert, müssen am Ende also auch alle x_i abgedeckt sein.

Optimalität: Wir zeigen, dass die Lösung S des Algorithmus auch tatsächlich optimal ist. D.h., es gibt keine Lösung S' mit $|S'| < |S|$, so dass alle x_i von S' überdeckt werden. Via Widerspruch:

(i) Sei $s_1 < \ldots < s_k$ die Lösung des Algorithmus. Seien $x_1 < \ldots < x_n$ die sortierten x_i.

(ii) Nehme an, die Lösung S ist nicht optimal.

(iii) Dann gibt es eine alternative Lösung $s'_1 < \ldots < s'_l$ mit $l < k$.

(iv) Um auf einen Widerspruch zu kommen, zeigen wir zunächst die folgende Hilfsaussage: Für alle $1 \leq i \leq \ell$ gilt $s'_i \leq s_i$. Per Induktion über i:

- IA $i = 1$. Per Definition des Algorithmus gilt $s_1 = \min\{x_1 + 5, M\}$ für das minimale x_i. Wäre $s'_1 > s_1$, dann gäbe es entweder kein $s' \in S'$ mit $|s' - x_i| \leq 5$ oder $s' > M$. Beide Fälle sind Widersprüche zur Zulässigkeit von S'.
- IV: Wir nehmen an, dass $s'_d \leq s_d$ für alle $d < i$ gilt.
- IS: Betrachte s'_i und s_i.
  - Per Definition des Algorithmus gilt $s_i = \min\{x_j + 5, M\}$ für ein x_j welches von s_{i-1} nicht überdeckt wird.

- Wenn $s_i = M$, dann muss auch $s'_i \leq M = s_i$ gelten da $s'_i \in [0, M]$.
- Betrachte also den Fall, dass $s_i = x_j + 5$.
- Per Sortierung wird x_j dann auch von $s_1, \ldots, s_{i-1}$ nicht überdeckt.
- Per IV gilt $s'_d \leq s_d$ für alle $d \leq i - 1$.
- Damit wird x_j also auch nicht von $s'_1, \ldots, s'_{i-1}$ überdeckt.
- Es folgt $s'_i \leq s_i$, da ansonsten $|x_j - s'_i| > 5$ und x_j von Lösung S' nicht überdeckt werden würde.

(v) Wir wissen jetzt also, dass $s'_i \leq s_i$ für alle $1 \leq i \leq \ell$.

(vi) Per Definition des Algorithmus wird $s_{\ell+1}$ hinzugefügt, weil ein x_j existiert, welches von s_ℓ nicht abgedeckt wird.

(vii) Per Sortierung und Hilfsaussage wird x_j dann auch weder von $s_1 \ldots, s_\ell$ noch $s'_1 \ldots, s'_\ell$ abgedeckt. Ein Widerspruch dazu, dass $S' = \{s'_1 \ldots, s'_\ell\}$ eine Lösung für das Problem ist.

(b) **Greedy Algorithmus:** Sei $X = \{x_1, \ldots, x_n\}$ die gegebene Menge von natürlichen Zahlen $X \subseteq [0, M]$.

Algorithmus 1 : Algorithm

```
1  S ← ∅;
2  current ← 0;
3  sort x_1, ..., x_n (aufsteigend);
4  for i=1 to n − 1 do
5  |  if |current − x_i| > 20 then
6  |  |  return false;
7  |  if |current − x_i| ≤ 20 and |current − x_{i+1}| > 20 then
8  |  |  current ← x_i;
9  |  |  S ← S ∪ {current};
10 if |M − current| > 20 then
11 |  if |M − x_n| > 20 then
12 |  |  return false;
13 |  else
14 |  |  S ← S ∪ {x_n};
15 return S;
```

Laufzeit: Abgesehen vom Sortieren in Zeile 3 haben die elementaren Operationen des Algorithmus Laufzeit $\mathcal{O}(1)$. Die Gesamtlaufzeit wird also entweder von der Laufzeit des Sortierens ($\mathcal{O}(n \log n)$) oder der Anzahl der Schleifendurchläufe dominiert ($\mathcal{O}(n)$). Insgesamt ergibt sich also eine Laufzeit von $\mathcal{O}(n \log n)$.
Terminierung: Der Algorithmus terminiert nach n Iterationen der for-Schleife.
Korrektheit:

- Algorithmus terminiert nach spätestens n Iterationen der Schleife.
- Korrektheit: Sei $s_1 < \ldots < s_k \subseteq X$ die Lösung des Algortihmus (sofern der Algorithmus nicht false zurück gibt) und seien $x_1 < \ldots < x_n$ die Elemente von X. Wir argumentieren, dass diese Lösung die Anforderungen der Aufgabenstellung erfüllt:

- Per Definition der if-Abfragen in den Zeilen 5 und 7 gilt entweder $s_1 < 20$ oder $x_1 > 20$. Die Lösung erfüllt also entweder Kriterium (i) oder der Algorithmus gibt in Zeile 6 false zurück und es gibt keine Lösung für das Problem, da $x_1 > 20$.
- Per Definition der Zeilen 5-9 gilt für alle s_i mit $i \in \{1, \ldots, k-1\}$ entweder $s_i = \max_{x_j \in X:\, x_j \leq s_{i-1}+20} x_j$ oder es gibt kein $x_j \in X$ mit $x_j \leq s_{i-1} + 20$ und der Algorithmus gibt in false in Zeile 6 zurück. Im ersten Fall erfüllen s_{i-1} und s_i die Anforderung (iii). Anderfalls gibt es x_h und x_{h+1} in X mit $|x_h - x_{h+1}| > 20$. Dies impliziert, dass es keine zulässige Lösung gibt und die false-Antwort des Algorithmus richtig ist.
- Per if-Abfrage in Zeile 10-14 gilt entweder $M - s_k \leq 20$ oder $M - x_n > 20$. Im letzteren Fall gibt der Algorithmus false zurück und es kann keine Lösung geben, die Anforderung (ii) erfüllt, da $x_1 < \ldots < x_n < M - 20$. Im ersten Fall erfüllt die Lösung des Algorithmus Anforderung (ii).

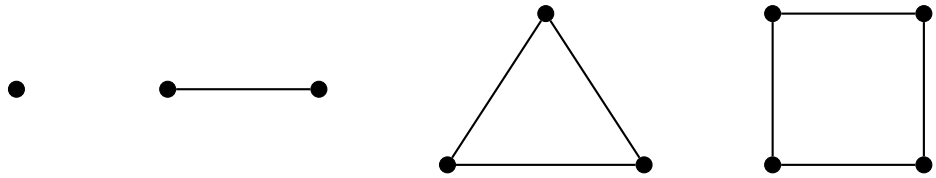
Optimalität: Sei S die Lösung des Algorithmus. Wir zeigen, dass es kein $S' \subseteq X$ mit $|S'| < |S|$ gibt, dass die Anforderungen (i),(ii) und (iii) aus der Aufgabenstellung erfüllt. Via Widerspruch:

1. Seien $x_1 < \ldots < x_n$ die Elemente von X.
2. Sei $S = \{s_1, \ldots, s_k\} \subseteq X$ die Lösung des Algorithmus mit $s_1 < \ldots < s_k$.
3. Nehme an, S ist nicht optimal.
4. Dann gibt es Lösung $S' = \{s'_1, \ldots, s'_\ell\}$ mit $\ell < k$. Nehme an $s'_1 < \ldots, s'_\ell$.
5. Wir zeigen die Hilfsbehauptung $s_i \geq s'_i$ für alle $i \leq \ell$:
   - Per Widerspruch: Nehme an $s_i < s'_i$ für mindestens ein $i \in \{1, \ldots, \ell\}$.
   - Sei i der kleinste Index mit $s_i < s'_i$.
   - Wenn $i = 1$, dann gilt $s_1 = \max_{x_j \leq 20} x_j$ per Definition des Algorithmus und daher $s'_1 > 20$. Dies ist ein Widerspruch zur Zulässigkeit von S', da S' dann (i) nicht erfüllt.
   - Wenn $i > 1$, dann gilt $s_{i-1} \geq s'_{i-1}$ da i der kleinste Index mit $s_i < s'_i$ ist. Wenn s_i in Zeile 9 zu S hinzugefügt wurde, gilt außerdem $s_i = \max_{x_j \in X:\, x_j \leq s_{i-1}+20} x_j$ per Definition des Algorithmus. Wegen $s_{i-1} \geq s'_{i-1}$ und $s_i < s'_i$ gilt dann $s'_i - s'_{i-1} > 20$. Ein Widerspruch zur Zulässigkeit von S', da S' dann (iii) nicht erfüllt.

     Wenn s_i in Zeile 14 hinzugefügt wurde, dann gilt $s_i = x_n$ und $s'_i > s_i$ kann nicht gelten, da x_n der größte Wert in X ist.
6. Betrachte $s_{\ell+1}$. Per Hilfsbehauptung gilt $s_\ell \geq s'_\ell$.
7. Wenn $s_{\ell+1}$ in Zeile 9 hinzugefügt wurde, dann gilt $s_{\ell+1} = \max_{x_j \in X:\, x_j \leq s_\ell+20} x_j$ und es muss ein $x_j \in X$ geben mit $x_j - s_\ell > 20$. Dann gilt aber auch $x_j - s'_\ell > 20$ und damit $M - s'_\ell > 20$. Ein Widerspruch zur Zulässigkeit von S', da (iii) dann nicht erfüllt ist.
8. Wenn $s_\ell = x_n$, dann gilt $M - s_\ell > 20$ per Zeile 10. Damit gilt dann auch $M - s'_\ell > 20$ (da $s_\ell \geq s'_\ell$). Ein Widerspruch zur Zulässigkeit von S', da (ii) dann nicht erfüllt ist.

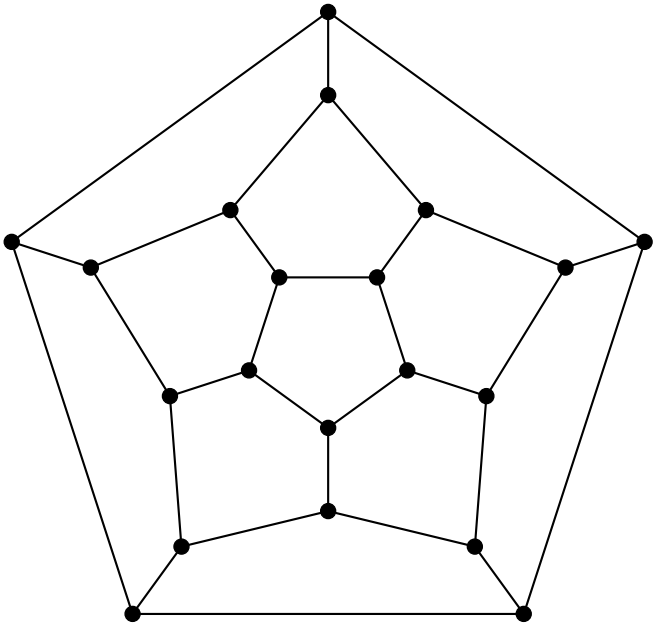
Präsenzübung 3.4

a) Wie viele Kanten müssen mindestens hinzugefügt werden, damit der folgende Graph einen Eulerpfad enthält? Wie viele für eine Eulertour? Stellt Eure Lösung graphisch dar.

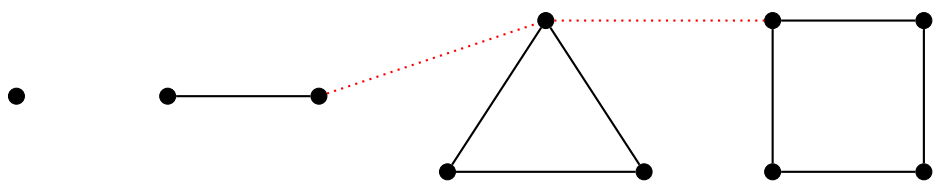


b) Ein Weg P in einem gerichteten oder ungerichteten Graphen $G = (V, E)$ heißt *Hamiltonscher Weg*, wenn P jeden Knoten aus V genau einmal durchläuft. Ist P zusätzlich geschlossen, so nennt man P *Hamiltonkreis*.

Prüft, ob der folgende Graph einen Eulerschen Weg oder eine Eulersche Tour, oder einen Hamiltonschen Pfad oder Kreis enthält und zeichnet gegebenenfalls Eure Lösung.

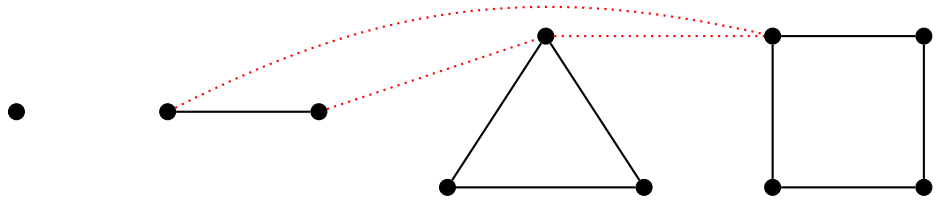


a) Für einen Eulerpfad müssen mindestens zwei Kanten hinzugefügt werden:



Weniger zusätzliche Kanten sind nicht möglich, da wir für einen Eulerpfad die letzten drei Zusammenhangskomponenten verbinden müssen.

Für eine Eulertour wird eine zusätzliche Kante benötigt:



Weniger zusätzliche Kanten sind nicht möglich, da wir durch das Verbinden der Zusammenhangskomponenten mindestens zwei Knoten mit ungeraden Grad erzeugen. Wir benötigen also mindestens eine weitere Kante damit alle Knoten einen gerade Grad haben.

b) Der Graph hat keine Eulertour, da es Knoten mit ungeraden Grad gibt. Der Graph hat keinen Eulerpfad, da es mehr als zwei Knoten mit ungeraden Grad gibt.

Der Graph hat einen Hamiltonkreis: