# Algorithmen & Datenstrukturen

## Übungsstunde 8

Michelle Honegger - CAB G59 - 10.11.2024

# Miniquiz 8

- Passwort: **graphs21**
- 9 Fragen
- 10 Minuten
- 0.111 Punkte pro richtige Antwort (denke ich)

# Programm

- Miniquiz
- Theory Recap
- Prüfungsaufgaben
- Besprechung Aufgaben

# Besprechung Miniquiz

For the longest ascending subsequence (LAT), which partial problem corresponds to the following recursion?

$$DP(i,\ell) = \begin{cases} 1, & \text{if there is } j < i \text{ with } DP(j,\ell-1) = 1 \text{ and } A[j] < A[i] \\ 0, & \text{otherwise} \end{cases}.$$

- a. $DP(i,\ell) =$ "there is an ascending subsequence of length $\ell$ ending in $i$".
- b. $DP(i,\ell) =$ "$A[i]$ is the smallest ending of an ascending subsequence of length $\ell$ in $A[1 \ldots i]$".

Recall the setting of the Subset Sum problem. We are given a list of $n$ integers and a number $b$ and are asked to answer whether there exists a subset in the list that sums to $b$.

If $b$ is at most $10n$, there exists a $O(n^2)$ time algorithm that solves the problem.

- Wahr
- Falsch

Consider the Subset Sum problem. Let $x$ be the number of different values $s \geq 0$ which can be expressed as a subset sum of $A[1..i]$. Similarly, let $y$ be the number of different values $s \geq 0$ which can be expressed as a subset sum of $A[1..i+1]$. Then, $y \leq 2x$.

- Wahr
- Falsch

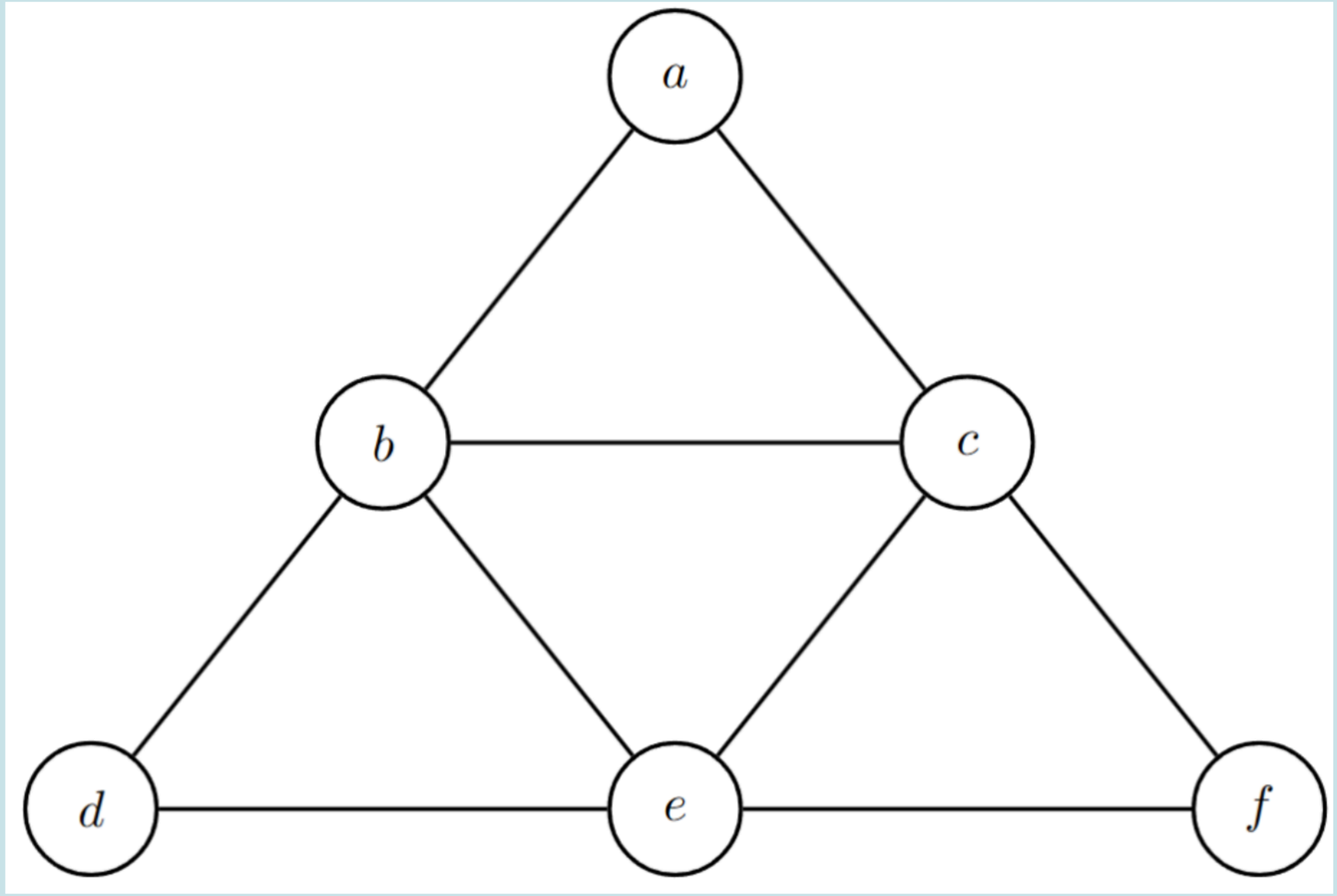
Consider the approximation algorithm we saw in the lecture for Knapsack. Let $OPT$ be an optimal solution (i.e. set of elements chosen) for the original problem and let $\widetilde{OPT}$ be an optimal solution for the problem with rounded profits. What does the algorithm compute?

- a. $OPT$
- b. $\widetilde{OPT}$
- c. None of the above.

Let $G$ be a graph with $x$ connected components.

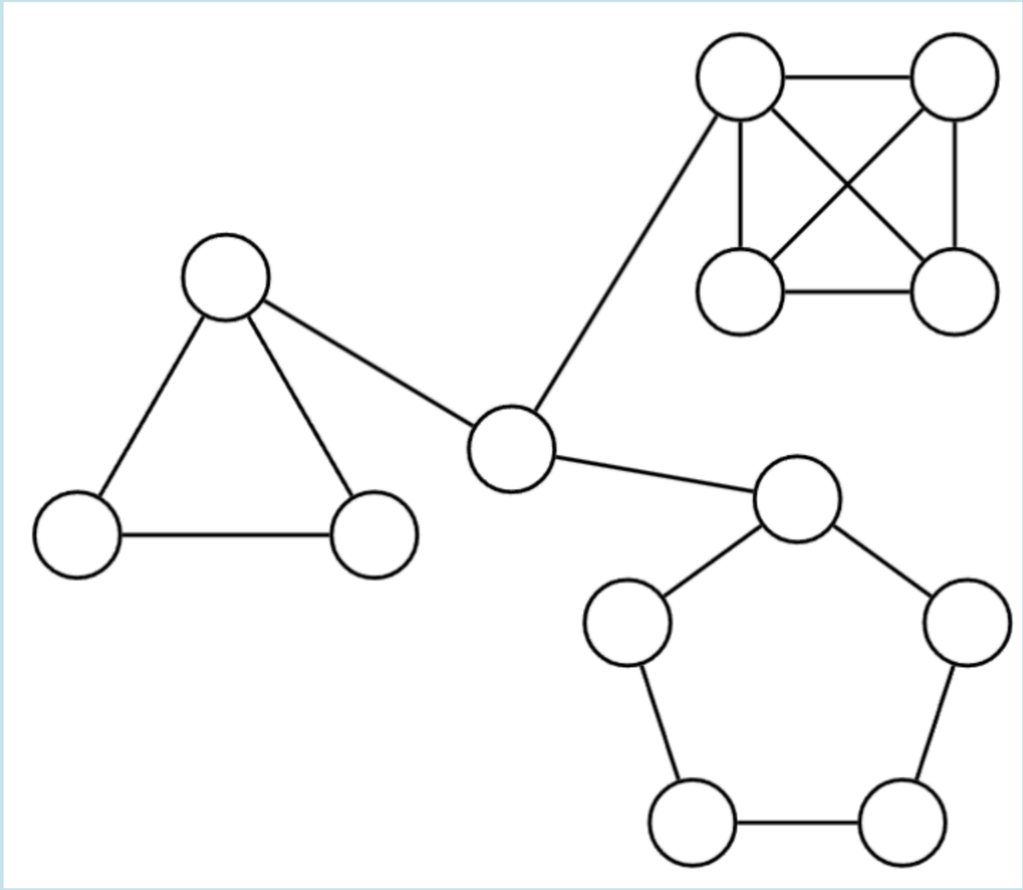
| True | False | |
|---|---|---|
| ○ | ○ | Removing any edge yields a graph with at most $x + 1$ connected components. |
| ○ | ○ | Removing any vertex (along with the incident edges) yields a graph with at most $x + 1$ connected components. |
| ○ | ○ | Removing any vertex (along with the incident edges) with degree at most 4 yields a graph with at most $x + 3$ connected components. |

Bewertung: **Teilpunkte** ?




The above graph contains a closed Eulerian walk.

- Wahr
- Falsch



| True | False | | |
|---|---|---|---|
| ○ | ○ | The above graph has a Hamilton path. | |
| ○ | ○ | It is possible to add one edge so that the graph contains a Hamilton path. | |

Bewertung: **Teilpunkte**

Answer the following True/False questions.

| True | False | |
|---|---|---|
| ○ | ○ | A polynomial time algorithm is known for deciding whether a closed Eulerian walk exists in a graph. |
| ○ | ○ | A polynomial time algorithm is known for deciding whether a Hamilton path exists in a graph. |

Bewertung: **Teilpunkte** ?

Let $G = (V, E)$ be a graph with 5 vertices $v_1, v_2, v_3, v_4, v_5$. Suppose that

$$\deg(v_1) = 2,\ \deg(v_2) = 2,\ \deg(v_3) = 3,\ \deg(v_4) = 3,\ \deg(v_5) = 4.$$

How many edges does $G$ have? *(Your answer should consist of a single integer.)*

Answer:

# Theory Recap

# Graphen

- $G = (V, E, (c))$
  - V sind Knoten um Knoten anzugeben nutzen wir meist $v_i \in V$ o.ä.
  - E sind Kanten wir nutzen meist $e \in E$
    - Können gerichtet $(u, v)$ oder ungerichtet $\{u, v\}$ sein
  - c ist eine allfällige Gewichtsfunktion, die jeder Kante ein Gewicht zuweist

# Graphen

- Zwei Knoten sind **adjazent**, wenn sie durch eine Kante verbunden sind
- Eine Kante ist **inzident** zu einem Knoten, wenn der Knoten ein Endpunkt der Kante ist
  - $deg(v) =$ #inzidente Kanten an Knoten v
  - Handschlaglemma: $\sum_{v \in V} deg(v) = 2|E|$

# Graphen

- u ist von v erreichbar, wenn es einen Weg von v nach u gibt mit Kanten im Graphen
- Eine **Zusammenhangskomponente (ZHK)** ist ein Teilgraph, in welchem alle Knoten gegenseitig erreichbar sind
- Ein Graph ist **zusammenhängend**, wenn er aus nur einer ZHK besteht

# Graphen

- Ein **Weg** ist eine Folge benachbarter Knoten
- Ein **Pfad** ist ein Weg ohne wiederholte Knoten
- Ein **Zyklus** ist ein Weg mit $v_0 = v_l, l \geq 2$

# Eulerwege

- durchlaufe jede Kante genau 1 mal
- Wenn Eulerweg existiert dann ≤2 Knotengrade ungerade
  - Bei jedem Knoten rein und raus ausser ev. Start- und Endknoten

# Eulerzyklen

- Durchlaufe jede Kante genau 1 mal und lande wieder beim Startknoten
- ∃Eulerzyklus ⇔ Alle Knotengrade sind gerade und alle Kanten in 1 ZHK
  - ⇒: Gleich wie Eulerwege
  - ⇐: Via Walk-Algorithmus

# Walk-Algorithmus

- Berechne Zyklen ohne wiederholte Kanten bis alle Kanten aufgebraucht
- Walk(u)
  if $\exists v.\ uv \in E$, nicht markiert
    markiere $uv$
    Walk(v)

# Hamiltonkreise

- Besuchen jeden Knoten exakt 1 mal
- haben selben Start- und Endknoten
- NP-hard Problem

**Exercise 7.5** *Zebra arrays* **(1 point)**.

Zebra array is square

A square two-dimensional array $Z[1 \dots k][1 \dots k]$ with entries in $\{0, 1\}$ is called a *zebra array* if no two adjacent entries of $Z$ are equal. We say two distinct entries $Z[i_1][j_1]$ and $Z[i_2][j_2]$ in $Z$ are adjacent if

- $i_1 = i_2$ and $|j_1 - j_2| \leq 1$; or
- $|i_1 - i_2| \leq 1$ and $j_1 = j_2$.

Aufzeichnen, ausser absolut klar beim draufschauen

Describe a DP algorithm that, given a two-dimensional array $A[1 \dots n][1 \dots m]$ with entries in $\{0, 1\}$, outputs the size of a largest zebra array contained in $A$. That is, the largest $k$ such that, for some $1 \leq i \leq n - k + 1, 1 \leq j \leq m - k + 1$, the array $A[i \dots i + k - 1][j \dots j + k - 1]$ is a zebra array. Your algorithm should have asymptotic runtime complexity at most $O(nm)$.

In your solution, address the following aspects:

1. *Dimensions of the DP table*: What are the dimensions of the $DP$ table?
2. *Subproblems*: What is the meaning of each entry?
3. *Recursion*: How can an entry of the table be computed from previous entries? Justify why your recurrence relation is correct. Specify the base cases of the recursion, i.e., the cases that do not depend on others.
4. *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
5. *Extracting the solution*: How can the solution be extracted once the table has been filled?
6. *Running time*: What is the running time of your solution?

***Hint:*** Use a DP table $B[1 \dots n][1 \dots m]$. The meaning of entry $B[i][j]$ is

$$B[i][j] = \text{size of the largest zebra array in } A \text{ whose bottom-right entry is } A[i][j].$$

***Hint:*** Your recursion to compute $B[i][j]$ should involve the entries $B[i][j-1]$, $B[i-1][j]$, $B[i-1][j-1]$ and also the entries $A[i][j]$, $A[i][j-1]$, $A[i-1][j]$, $A[i-1][j-1]$.

# Exercise Recommendations

## Aufgabenblatt 8

- 8.1 Bonus
- 8.2 Interessant sich zu überlegen, jedoch keine hohe Priorität
- 8.3 Bonus
- 8.4 Sehr nützliche Aufgabe, machen wenn Zeit vorhanden. Bringt auch für AnW im nächsten Semester was
- 8.5 Semi interessant, keine hohe Priorität

# Peergrading

- Aufgabe 7.1
- Grading Scheme in Lösungen beachten