Quiz

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. $DP(i,\ell) =$ "$A[i]$ is the smallest ending of an ascending subsequence of length ℓ in $A[1\dots i]$".

b. $DP(i,\ell) =$ "there is an ascending subsequence of length ℓ ending in 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.

True

False

Subset Sum in $O(n\ b) = O(n\ 10n) = O(n^2)$

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, $y \leq 2x$.

True

False

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 $G = (V,E)$ 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: 7

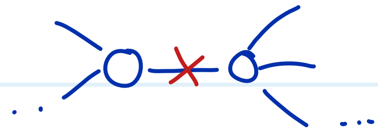
$\sum_{v \in V} \deg(v) = 2+2+3+3+4 = 14$

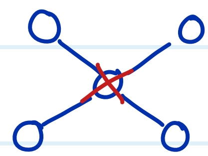
$\sum_{v \in V} \deg(v) = 2|E|$ $\Rightarrow 14 = 2|E|$

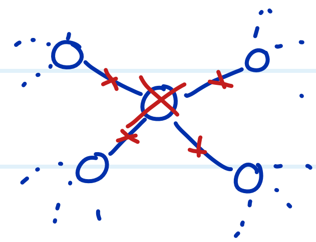
$\Rightarrow |E| = 7$

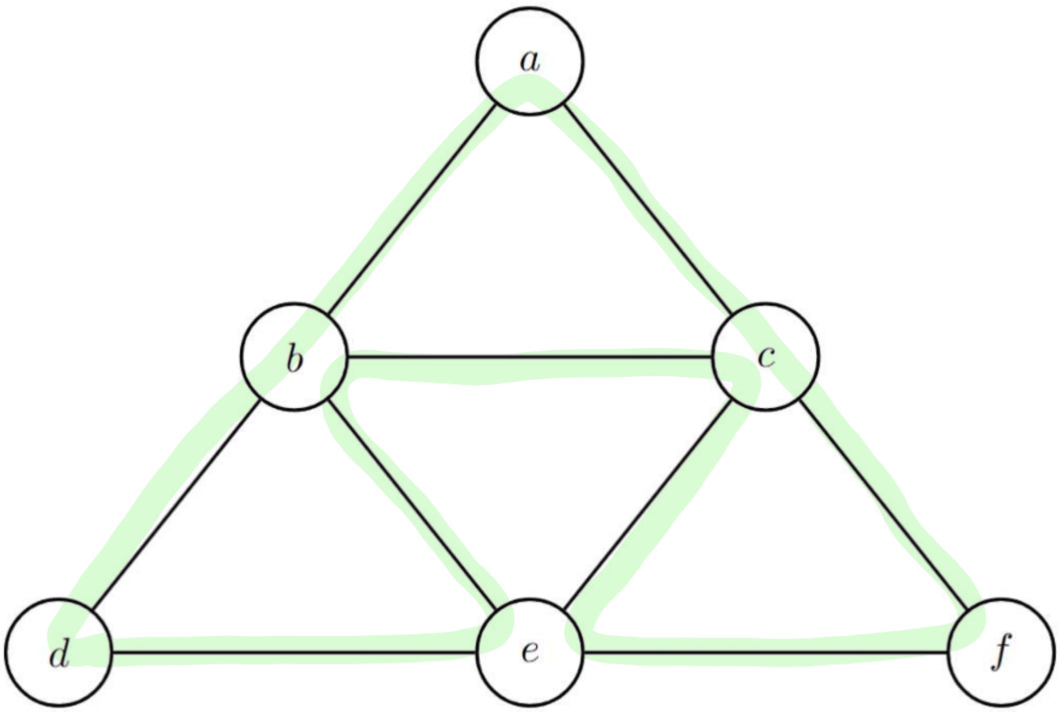
Let G be a graph with x connected components.

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







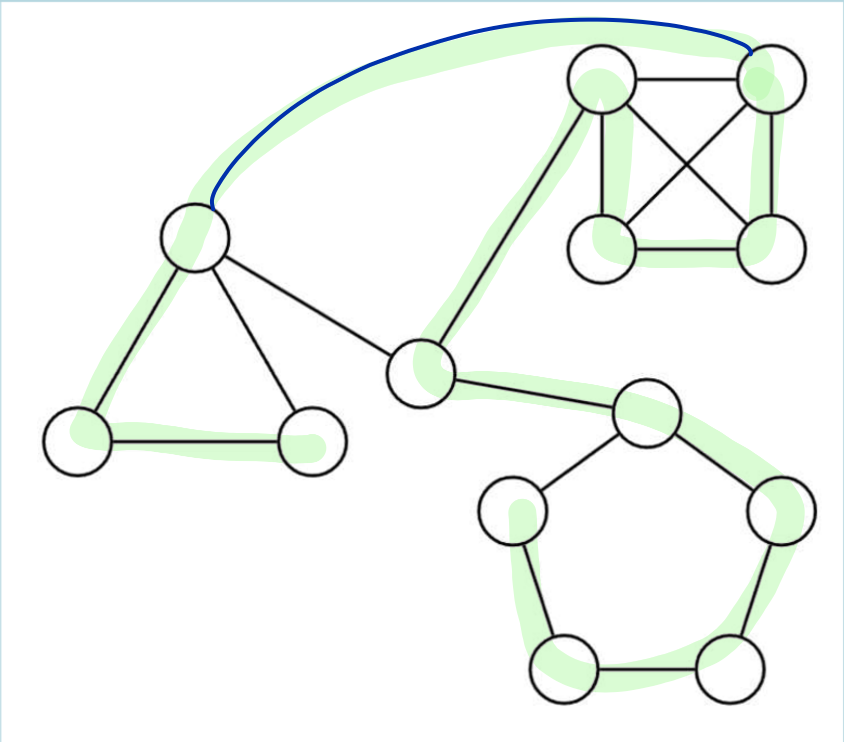


The above graph contains a closed Eulerian walk.

True

False

connected and all vertices have even degree



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

Answer the following True/False questions.

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

"höchstes Schwierigkeitslevel"

Graphentheorie

Graph: $G = (V, E)$

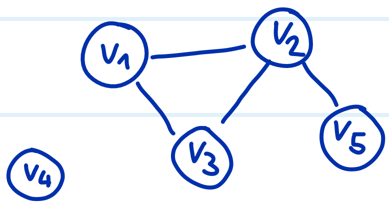
Knotenmenge $V = \{v_1, v_2, \ldots, v_n\}$ meistens ist $|V| = n$

Kantenmenge $E \subseteq \{\{u,v\} \mid u,v \in V\}$ meistens ist $|E| = m$

benachbart/adjazent: $u,v \in V$ sind benachbart, falls es eine Kante zwischen u und v gibt, d.h. $\{u,v\} \in E$

anliegend/inzident: Eine Kante $e \in E$ ist inzident zu Knoten $v \in V$, falls v einer der beiden Endpunkte von e ist

- For $v \in V$, the **degree** $\deg(v)$ of v (german "Knotengrad") is the number of edges that are incident to v.

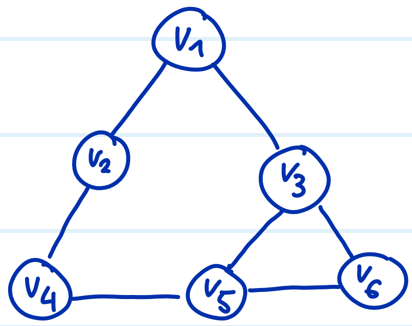


$V = \{a, b, c, d, e\}$

$E = \{\{a,b\}, \{b,c\}, \{c,e\}, \{e,b\}\}$

- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ (with $v_i \in V$ for all i) is a **walk** (german "Weg") if $\{v_i, v_{i+1}\}$ is an edge for each $0 \leq i \leq k-1$. We say that v_0 and v_k are the **endpoints** (german "Startknoten" and "Endknoten") of the walk. The **length** of the walk $(v_0, v_1, \ldots, v_k)$ is k.
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **closed walk** (german "Zyklus") if it is a walk, $k \geq 2$ and $v_0 = v_k$.
  Achtung: Zyklus ≠ cycle
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **path** (german "Pfad") if it is a walk and all vertices are distinct (i.e., $v_i \neq v_j$ for $0 \leq i < j \leq k$).
- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **cycle** (german "Kreis") if it is a closed walk, $k \geq 3$ and all vertices (except v_0 and v_k) are distinct.

How many different paths from v_1 to v_6 are there and what length do they have? 4



How many different walks with endpoints v_1 and v_6 are there? ∞

How many different cycles are there? 3

What are all possible lengths for a walk with endpoints v_1 and v_6? $\mathbb{N} \setminus \{0,1\}$

What are all possible lengths for a path with endpoints v_1 and v_6? 2,3,4,5

- An **Eulerian walk** (german "Eulerweg") is a walk that contains every edge exactly once.
- A **closed Eulerian walk** (german "Eulerzyklus") is a closed walk that contains every edge exactly once.

} Algorithmus in $O(n+m)$

- A **Hamiltonian path** (german "Hamiltonpfad") is a path that contains every vertex.
- A **Hamiltonian cycle** (german "Hamiltonkreis") is a cycle that contains every vertex.

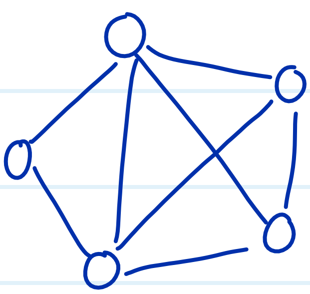
} NP - Vollständig

Sei G zusammenhängend

G hat Eulerweg <=> Alle Knotengrade ausser höchstens zwei sind gerade

(ungerade Knotengrade sind Start- und Endpunkte)

G hat Eulerzyklus <=> Alle Knotengrade sind gerade

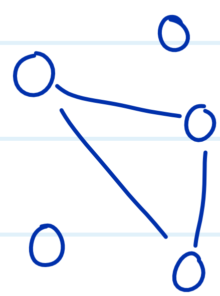


Eulerweg? ✓

Eulerzyklus? ✗

Hamiltonpfad? ✓

Hamiltonkreis? ✓



Eulerweg? ✓

Eulerzyklus? ✓

Hamiltonpfad? ✗

Hamiltonkreis? ✗

Sei G beliebig.

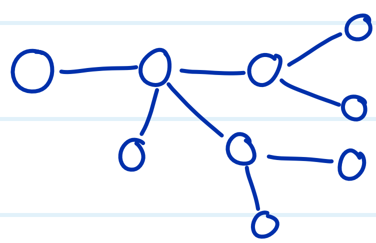
G hat Eulerweg <=> Alle Knotengrade ausser höchstens zwei sind gerade und alle Kanten sind in derselben ZHK

G hat Eulerzyklus <=> Alle Knotengrade sind gerade und alle Kanten sind in derselben ZHK

- For $u, v \in V$, we say u **reaches** v (or v is **reachable** from u; german "u erreicht v") if there exists a walk with endpoints u and v, or equivalently, there exists a path with endpoints u and v.
- A **connected component** of G (german "Zusammenhangskomponente") is an equivalence class of the (equivalence) relation defined as follows: Two vertices $u, v \in V$ are equivalent if u reaches v.
- A graph G is **connected** (german "zusammenhängend") if for every two vertices $u, v \in V$, u reaches v, or equivalently, if there is only one connected component.

- A graph G is a **tree** (german "Baum") if it is connected and has no cycles.

$m = n-1$



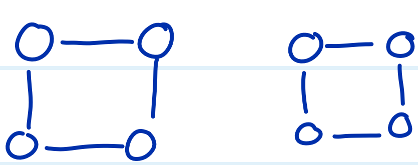
Handshake Lemma: $\boxed{\sum_{v \in V} \deg(v) = 2|E|}$

Beweis: Jede Kante ist zu genau zwei Knoten inzident.

(Wenn wir eine Kante hinzufügen, erhöht sich der Grad von genau zwei Knoten, nämlich den beiden Endpunkten der Kante um 1)

Aussage Jeder Graph G, in dem alle Knoten einen Grad von genau 2 haben, ist zusammenhängend.

Counterexample:



Aussage Sei G ein Graph mit $n \geq 3$ Knoten. Wenn G zusammenhängend ist und $m = n - 1$ Kanten hat, dann muss G einen Knoten vom Grad 1 haben.

Widerspruchsbeweis: Annahme G hat keinen Knoten mit Grad 1

$\Rightarrow \forall v \in V: \deg(v) \geq 2$ (Grad kann auch nicht 0 sein, sonst nicht mehr zusammenhängend)

$\Rightarrow \sum_{v \in V} \deg(v) \geq \sum_{v \in V} 2$

$\Rightarrow 2m \geq 2n$ $(n = |V|)$

$\Rightarrow m \geq n$ (Handschlag Lemma)

$\Rightarrow$ Widerspruch zu $m = n - 1$

Das beweist die Aussage. $\boxtimes$

Aussage Wenn es in einem Graphen einen Weg (walk) von Knoten a nach b gibt und einen Weg von b nach c, dann gibt es immer auch einen Pfad (path) von a nach c.

Beweis: Sei $(a, v_1, v_2, \ldots, v_k, b)$ ein Weg von a nach b

Sei $(b, w_1, w_2, \ldots, w_\ell, c)$ ein Weg von b nach c

Dann ist $(a, v_1, v_2, \ldots, v_k, b, w_1, w_2, \ldots, w_\ell, c)$ ein Weg von a nach c.

Solange ein Knoten $d \in V$ doppelt vorkommt, kürzen wir ab.

D.h. falls der Weg die Form $(\overbrace{x_1, \ldots, x_i}^{\text{Anfang}}, d, \overbrace{y_1, \ldots, y_j}^{\text{Mitte}}, d, \overbrace{z_1, \ldots, z_h}^{\text{Ende}})$ hat

kürzen wir zu $(x_1, \ldots, x_i, d, z_1, \ldots, z_h)$.

So erhalten wir einen Pfad von a nach c.

Aussage Wenn ein Graph G einen Hamiltonpfad enthält, dann muss G zusammenhängend sein.

Beweis Sei $(v_1, v_2, ..., v_n)$ ein Hamiltonpfad in G

Seien $x, y \in V$ beliebig

Fall $x = y$: x erreicht y weil $x = y$ (trivial)

Fall $x \neq y$: $x = v_i$ und $y = v_j$ für $i, j \in \{1, .., n\}$

w.l.o.g sei $i < j$

$(v_i, v_{i+1}, ..., v_{j-1}, v_j)$ ist ein Pfad von $v_i = x$ nach $v_j = y$

x erreicht y

Weil $\forall x, y \in V$: x erreicht y gilt, ist G per Definition zusammenhängend

Aussage Wenn ein Graph G einen Eulerweg (aber keinen Eulerzyklus) enthält, dann hat er genau zwei Knoten mit ungeradem Knotengrad.

Beweis. Es muss mindestens einen Knoten mit ungeradem Grad geben

Nach Handschlaglemma gilt: $\sum_{v \in V} \deg(v) = 2|E|$

Insbesondere ist also die Summe aller Knotengrade gerade

Deshalb muss es einen zweiten Knoten geben mit ungeradem Grad

Alle anderen Knotengrade sind gerade, weil es einen Eulerweg gibt.

Aussage Jeder Graph G mit $n \geq 3$ Knoten, in dem jeder Knoten einen Grad von mindestens $n/2$ hat, muss zusammenhängend sein.

Seien $x, y \in V$ beliebig.

Fall $x = y$: x erreicht y weil $x = y$ (trivial)

Fall $x \neq y$, $\{x,y\} \in E$: x erreicht y weil $\{x,y\} \in E$ (trivial)

Fall $x \neq y$, $\{x,y\} \notin E$:

sei $X = \{v \in V \mid \{x,v\} \in E\}$ die Menge der Nachbaren von x

und $Y = \{v \in V \mid \{y,v\} \in E\}$ die Menge der Nachbaren von y

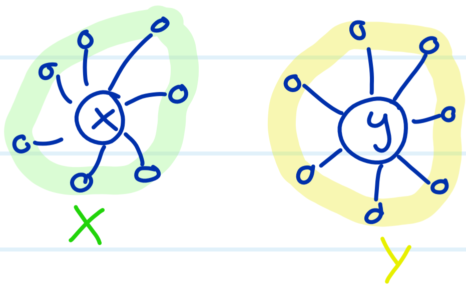
$\Rightarrow |X| \geq \frac{n}{2}$ und $|Y| \geq \frac{n}{2}$ und $x,y \notin X$ und $x,y \notin Y$

Widerspruchsbeweis. Annahme $X \cap Y = \emptyset$

$\Rightarrow |V| \geq |X| + |Y| + \underbrace{2}_{= |\{x,y\}|}$

$\Rightarrow n \geq \frac{n}{2} + \frac{n}{2} + 2$

$\Rightarrow n \geq n + 2$ ↯



Widerspruch, deshalb $X \cap Y \neq \emptyset$

Es gibt also einen Knoten $s \in X \cap Y$

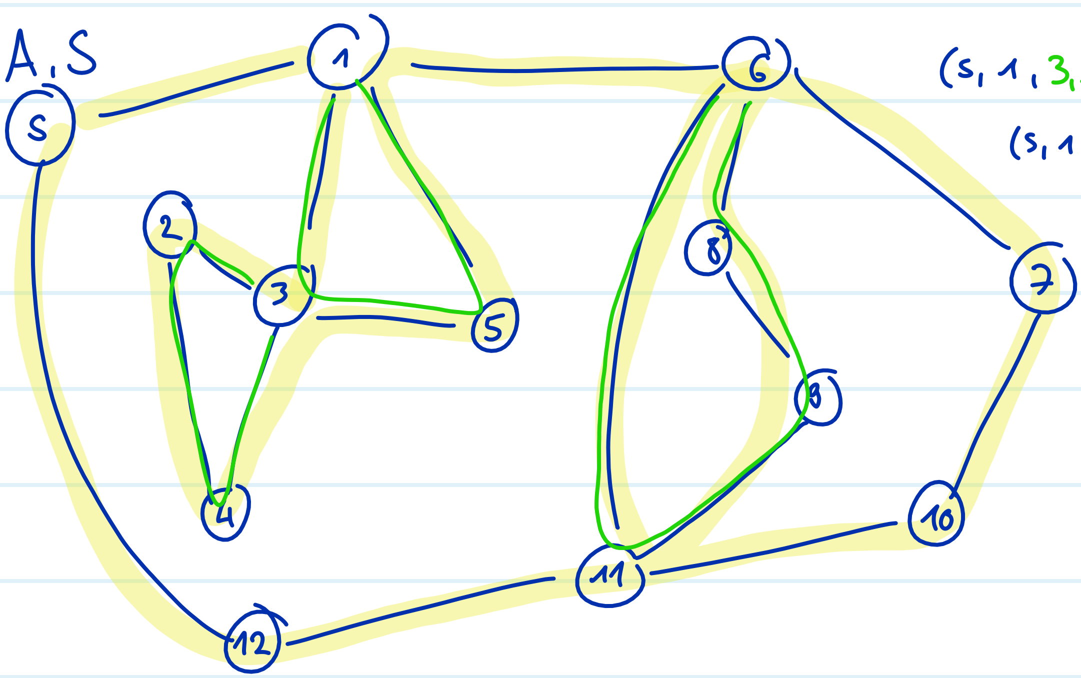
Dann ist (x, s, y) ein Pfad von x nach y, d.h x erreicht y

Weil $\forall x,y \in V$: x erreicht y gilt, ist G per Definition zusammenhängend

Eulerzyklus Algorithmus in O(E)

1. **Start:** Achilles startet am Startknoten s und findet einen beliebigen ersten Zyklus C.
2. **Initialisierung:** Die Schildkröte setzt ihre Haupttour T gleich diesem Zyklus ($T = C$).
3. **Wanderung:** Die Schildkröte beginnt, ihre Tour T ab s abzulaufen.
4. **Prüfung:** An *jedem* Knoten v, den sie auf ihrer Tour T erreicht, prüft die Schildkröte, ob von v noch **unbesuchte** Kanten ausgehen.
    - **Fall A (Keine Abzweigung):** Wenn nein, läuft sie einfach weiter zum nächsten Knoten gemäß T.
    - **Fall B (Abzweigung):** Wenn ja, **stoppt** die Schildkröte bei v.
5. **Neuer Zyklus:** Achilles wird bei v losgeschickt. Er läuft *ausschließlich* über unbesuchte Kanten, bis er einen neuen Zyklus C_{neu} gebildet hat und exakt zu v zurückkehrt.
6. **Einfügen:** Die Schildkröte fügt den neuen Zyklus C_{neu} an der Position v in ihre Haupttour T ein.
    - *(Beispiel: T war* `A -> v -> B`. *C_{neu} ist* `v -> X -> v`. *Neues T ist* `A -> [v -> X -> v] -> B`.*)*
7. **Fortsetzung:** Die Schildkröte setzt ihre Wanderung *sofort* auf dem neu eingefügten Pfad C_{neu} fort. Wenn sie C_{neu} beendet hat, folgt sie dem Rest der ursprünglichen Tour T (im Beispiel: ...nach B).
8. **Ende:** Der Algorithmus ist beendet, sobald die Schildkröte ihre *gesamte* (jetzt erweiterte) Tour T einmal durchlaufen hat, ohne Achilles (in Schritt 4B) erneut losschicken zu müssen.

$T = (s, 1, 6, 7, 10, 11, 12, s)$



$(s, 1, 3, 5, 1\ 6, 7, 10, 11, 12, s)$

$(s, 1, 3, 2, 4, 3, 5, 1, 6, 7, \ldots)$

$\vdots$