Aufgabe 1. Kleinaufgaben [19 Punkte]

a. Geben Sie eine Familie von DAGs an, in der alle DAGs $\Omega(n^2)$ Kanten haben bei n Knoten. [2 Punkte]

b. Angenommen, Sie wollen eine Folge von n ganzen Zahlen sortieren. Nennen Sie einen vergleichsbasierten Sortieralgorithmus, der im O-Kalkül optimales Worst-Case-Laufzeitverhalten aufweist. Würden Sie diesen Algorithmus auch in der Praxis verwenden? Begründen Sie kurz. [2 Punkte]

c. Lösen Sie die beiden folgenden Rekurrenzen im Θ-Kalkül:

$$\begin{aligned} T(n) &= 17n + 2T(n/3), \quad T(1) = 3 \\ S(n) &= 3n + 9S(n/3), \quad S(1) = 23 \end{aligned}$$

mit $n = 3^k$ und $k \in \mathbb{N}_{>0}$. [2 Punkte]

d. Zu einem gerichteten Graph $G = (V,E)$ definieren wir $\overline{G} := (V,\overline{E})$ mit

$$\overline{E} := \left\{ (u,v) \in V^2 \,\middle|\, (u,v) \notin E \right\} .$$

Sei G als Adjazenzfeld gegeben. Nun will man $\overline{G}$ aus G erzeugen, wobei auch $\overline{G}$ am Ende in Form eines Adjazenzfeldes vorliegen soll. Ein solcher Vorgang benötigt mindestens $\Omega(|V|^2)$ Zeit. Erklären Sie kurz warum das so ist. [2 Punkte]

(weitere Teilaufgaben auf den nächsten Blättern)

Fortsetzung von Aufgabe 1

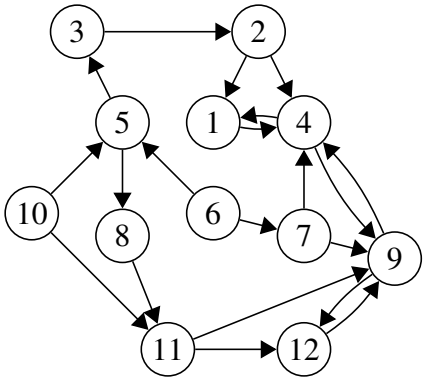
e. Skizzieren Sie kurz, wie man *locateLocally* in (a,b)-Bäumen so verändern kann, dass die *locate* Operation $O(\log b \log n)$ Zeit braucht statt $O(b \log n)$ Zeit. [2 Punkte]

f. Gegeben sei der unten abgebildete gerichtete Graph. Auf diesem Graphen soll eine Breitensuche durchgeführt werden. Wählen Sie den Startknoten so, dass bei der Breitensuche **keine Rückwärtskanten** auftreten.

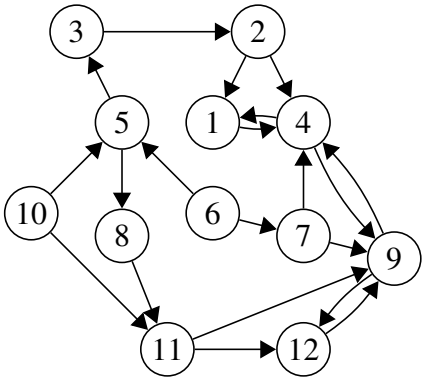
Markieren Sie den Startknoten und den zugehörigen Breitensuchbaum im Bild.

Hinweis: Der Startknoten darf auch so gewählt werden, dass die Breitensuche nicht alle Knoten des Graphen erreicht.

Für die **Lösung:**



Kopie zum **Rechnen:**



[3 Punkte]

g. Zeigen Sie oder widerlegen Sie, dass $2^{2^{n+1}} = O\left(2^{2^n}\right)$ gilt. [2 Punkte]

(weitere Teilaufgaben auf dem nächsten Blatt)

Fortsetzung von Aufgabe 1

h. Zeigen oder widerlegen Sie, dass $n^{\log_2 \log_2 n} = \Omega(n^2)$ gilt. [2 Punkte]

i. Nennen Sie eine Operation, die auf unbeschränkten Arrays im besten Fall $\Omega(n)$ Zeit braucht und auf doppelt verketten Listen im schlimmsten Fall $O(1)$ Zeit braucht (n sei die Anzahl der jeweils enthaltenen Elemente). [1 Punkte]

j. Nennen Sie eine Operation, die auf einfach verketteten Listen im schlimmsten Fall $\Omega(n)$ Zeit braucht und auf unbeschränkten Arrays im schlimmsten Fall $O(1)$ Zeit braucht (n sei die Anzahl der jeweils enthaltenen Elemente). [1 Punkte]

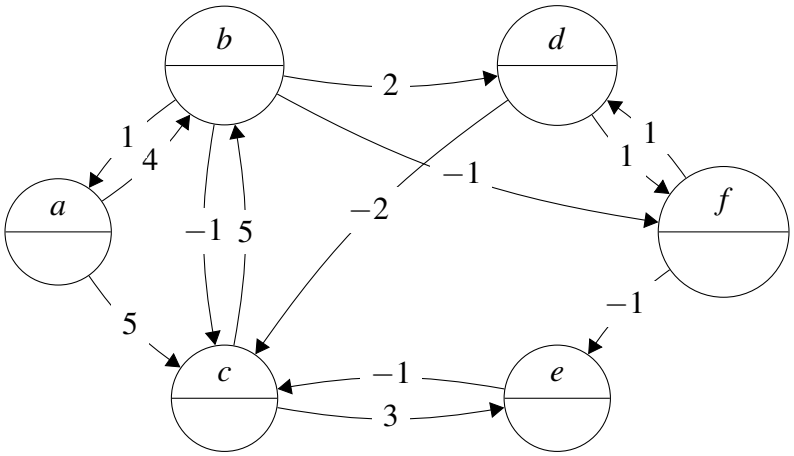
Aufgabe 2. Bellman-Ford Algorithmus [6 Punkte]

Gegeben sei der abgebildete gerichtete Graph mit Kantengewichten. Auf diesem Graph soll der Bellman-Ford Algorithmus mit *a* als Startknoten ausgeführt werden.

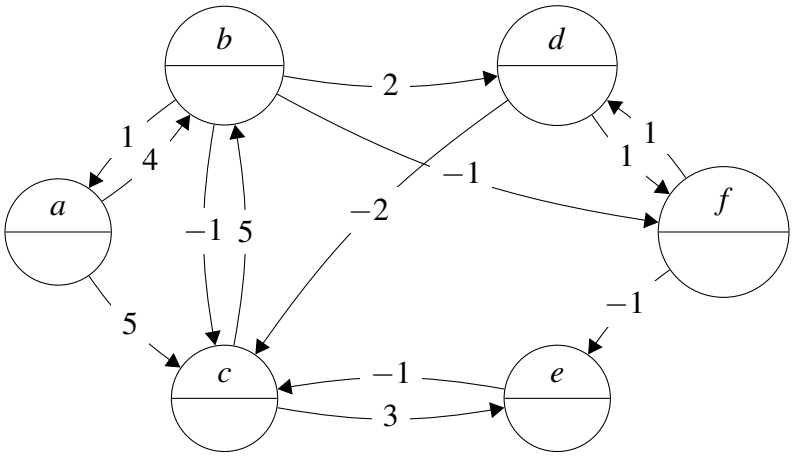
a. Tragen Sie in jeden Knoten jeweils die kürzeste Distanz von *a* zu diesem Knoten ein (in der unteren Hälfte jedes Knoten wurde dafür Platz frei gelassen). [2 Punkte]

b. Zeichnen Sie den vom Bellman-Ford Algorithmus berechneten Baum kürzester Wege in den Graph ein. [2 Punkte]

Tragen Sie die **Lösungen** von **a.** und **b.** bitte in **diesen** Graphen ein:



Kopie des obigen Graphen zum **Rechnen:**



(Teilaufgabe **c.** auf dem nächsten Blatt)

Fortsetzung von Aufgabe 2

c. Geben Sie für den Bellman-Ford Algorithmus eine Bearbeitungsreihenfolge der Kanten an, so dass für den obigen Graph mit *a* als Startknoten eine einzige Runde des Algorithmus ausreicht, um die kürzesten Distanzen von *a* zu jedem Knoten zu berechnen. [2 Punkte]

Aufgabe 3. Entwurf einer Datenstruktur [8 Punkte]

Eine Datenstruktur D soll Paare der Form $(\mathit{ElementID},\ \mathit{Bewertung})$ speichern (sowohl *ElementID* und *Bewertung* seien Zahlen aus $\mathbb{Z}$). Paare (x,c) und (y,c') mit $x = y$ dürfen in D **nicht gleichzeitig** vorkommen. Weiter soll D folgende Operationen mit jeweils gegebenem Laufzeitverhalten unterstützen (n bezeichne dabei die Anzahl der in D enthaltenen Paare):

- $\mathit{insert}(x : \mathit{ElementID},\ c : \mathit{Bewertung})$
  fügt (x,c) in D ein. Ist schon ein Paar (y,c') mit $y = x$ in D vorhanden, so wird D nicht verändert. Der Zeitbedarf sei erwartet $\mathrm{O}(\log n)$.

- $\mathit{removeMin}() : \mathit{ElementID} \times \mathit{Bewertung}$
  entfernt aus D ein Paar (x,c) mit **minimaler Bewertung** c und liefert das entfernte Paar als Ergebnis. Der Zeitbedarf sei erwartet $\mathrm{O}(\log n)$.

- $\mathit{contains}(x : \mathit{ElementID}) : \mathit{boolean}$
  stellt fest, ob D ein Paar (y,c) mit $y = x$ enthält. Der Zeitbedarf sei erwartet $\mathrm{O}(1)$.

Anwendungsbedingt sei bekannt, dass in D zu jedem Zeitpunkt höchstens m Paare gleichzeitig vorhanden sind, d. h. es gilt stets $n \leq m$. Über die Beschaffenheit der auftretenden Paare wisse man im Voraus aber nichts.

a. Skizzieren Sie, wie Sie diese Datenstruktur und die drei beschriebenen Operationen realisieren würden. [6 Punkte]

(Mehr Platz zum Schreiben und Teilaufgabe **b.** auf dem nächsten Blatt)

Fortsetzung von Aufgabe 3

b. Begründen Sie kurz, warum die drei Operationen in Ihrer Realisierung das geforderte Laufzeitverhalten aufweisen. [2 Punkte]

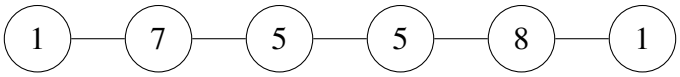
Aufgabe 4. Dynamisches Programmieren [12 Punkte]

Definition: Zu einem ungerichteten Graph $G = (V, E)$ ist eine Menge $U \subset V$ eine *unabhängige Menge* genau dann, wenn keine Knoten aus U in G benachbart sind, d. h. $\forall u, v \in U : \{u, v\} \notin E$.

Definition: Zu einem ungerichteten Graph $G = (V, E)$ mit Knotengewichtsfunktion $c : V \to \mathbb{N}_{>0}$ ist eine Menge $U \subset V$ eine *maximale unabhängige Menge*, wenn U unabhängige Menge ist und unter allen unabhängigen Mengen das größtmögliche Gesamtgewicht hat, d. h. es existiert keine unabhängige Menge $U' \subset V$ mit $\sum_{u \in U'} c(u) > \sum_{u \in U} c(u)$

Festlegung: Im Folgenden sei $G := v_1 - v_2 - \cdots - v_k$ ein Pfad, d. h. G bestehe ausschließlich aus Kanten $\{v_\ell, v_{\ell+1}\}$ für $\ell \in \{1..k-1\}$. Für $\ell \leq k$ sei weiter $G_\ell := v_1 - \cdots - v_\ell$ der Teilpfad von G, der beim Knoten v_1 beginnt und bis einschließlich Knoten v_ℓ geht. G_0 soll als leerer Pfad interpretiert werden.

a. Geben Sie zu dem abgebildeten Pfad die *maximale unabhängige Menge* U an, indem Sie die Knoten markieren, die zu U gehören. Geben Sie außerdem das Gesamtgewicht von U an. Die Knotengewichte stehen in den Knoten. [2 Punkte]



b. Es sei $m(\ell)$ das Gesamtgewicht einer *maximalen unabhängigen Menge* U_ℓ auf G_ℓ. Geben Sie eine Rekurrenz für $m(\ell)$ an, indem Sie auf die Werte der $m(i)$ für $i < \ell$ zurückgreifen! Begründen Sie kurz! [3 Punkte]

$$\begin{aligned} m(0) &= 0 \\ m(1) &= c(v_1) \\ m(\ell) &= \end{aligned}$$

(Teilaufgaben **c. und d.** auf dem nächsten Blatt)

Fortsetzung von Aufgabe 4

c. Skizzieren Sie einen Algorithmus, der eine *maximale unabhängige Menge U* auf einem Pfad $G = v_1 - \cdots - v_k$ mit Knotengewichtsfunktion c **berechnet** und **ausgibt.** Der Algorithmus darf die Laufzeit $O(k)$ nicht überschreiten. [3 Punkte]

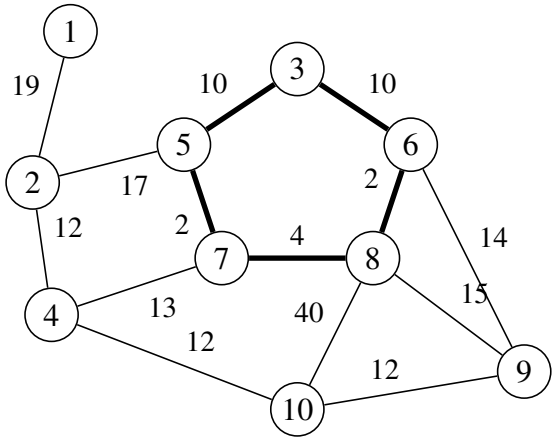
d. Skizzieren Sie kurz einen Linearzeit-Algorithmus, der eine maximale unabhängige Menge auf einem ungerichteten **Baum** mit Knotengewichtsfunktion c **berechnet** und **ausgibt.**
[4 Punkte]

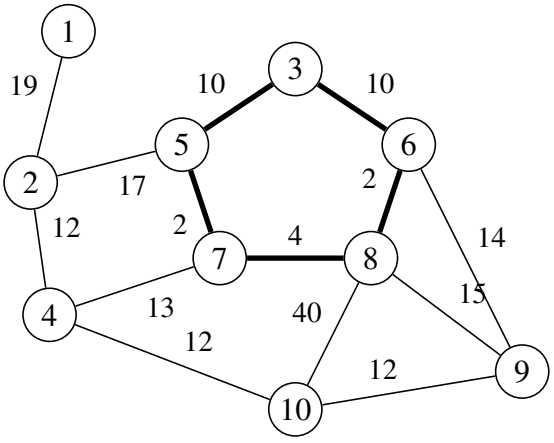
Aufgabe 5. Minimale Spannbäume [9 Punkte]

Gegeben sei ein ungerichteter zusammenhängender Graph G, in dem jede Kante mit einem positiven Gewicht versehen ist. In G sei ein einfacher Kreis markiert, so dass alle Kreiskanten ein Gewicht $< r$ und alle Kanten außerhalb des Kreises ein Gewicht $> r$ haben für ein $r \in \mathbb{R}_+$.
Hinweis: In einem einfachen Kreis haben alle Knoten Grad 2.

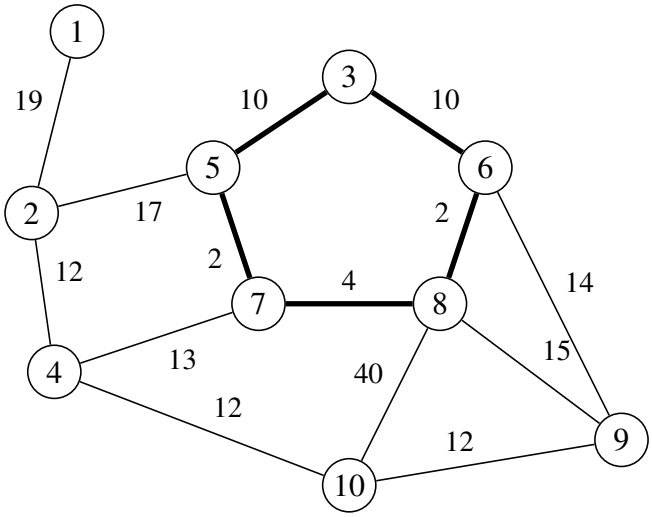
a. Zeichnen Sie in die folgenden beiden identischen Beispielgraphen jeweils einen MST ein. Dabei soll jeder MST mindestens eine Kante des markierten Kreises enthalten, die der andere MST nicht enthält. [3 Punkte]

Für die **Lösung:**





Kopie des Graphen zum **Rechnen:**



(Teilaufgabe **b.** auf dem nächsten Blatt)

Fortsetzung von Aufgabe 5

b. Sei G ein Graph wie in der Einleitung zu dieser Aufgabe beschrieben. Sei k die Anzahl der Kanten auf dem markierten Kreis in G. Zeigen Sie: Jeder MST in G enthält genau $k-1$ Kanten des markierten Kreises. [6 Punkte]

Aufgabe 6. Hashing [6 Punkte]

Betrachten Sie die folgende Hashtabelle:

0	1	2	3	4	5	6	7	8	9
73	75	23	10	18		62	61	83	86

Zur Kollisionsauflösung wird offenes Hashing mit linearer Suche verwendet. Die Hashfunktion, $h(x) = x$ DIV 10, bildet eine Zahl $x \in \{1..99\}$ auf ihre höchstwertige Ziffer ab. Beispiel: $h(62) = 6$, $h(6) = 0$.

a. Sei die Tabelle nun leer. Geben Sie eine Folge von *insert*-Operationen an, so dass die Tabelle nach Ausführen dieser Operationsfolge den obigen Zustand hat. [3 Punkte]

b. Sei die Tabelle in dem oben angegebenen Zustand. Geben Sie den Zustand der Tabelle nach dem Entfernen von 75 an. [3 Punkte]

0	1	2	3	4	5	6	7	8	9

Konzeptpapier (Abgabe freiwillig)