QUIZ-NACHBESPRECHUNG

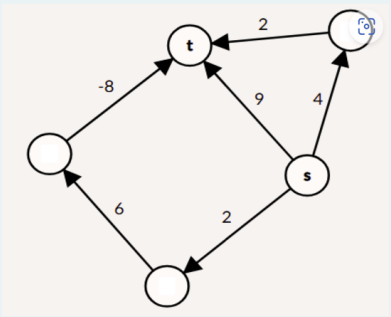
Angenommen während der Ausführung des Dijkstra-Algorithmus zum Finden eines Pfades von s nach t in einem Graphen mit nicht-negativen Kantengewichten sieht das Array d der Distanzen wie folgt aus:

v	s	v_1	v_2	v_3	v_4	t
$d(v)$	0	3	2	4	8	9

Welche der folgenden Aussagen muss wahr sein, **nachdem der Dijkstra-Algorithmus beendet ist**?

Wählen Sie eine Antwort:

- a. $d(t) < 9$
- b. $d(v_2) = 2$
- c. $d(v_4) = 8$
- d. Alle der oben genannten.

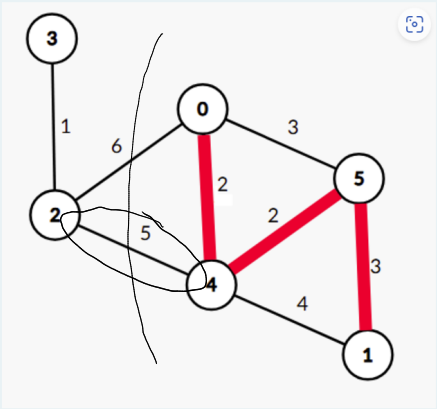


Angenommen wir wenden den Bellman-Ford-Algorithmus auf den oben dargestellten gewichteten, gerichteten Graphen an, um einen kürzesten Pfad von s nach t zu finden.

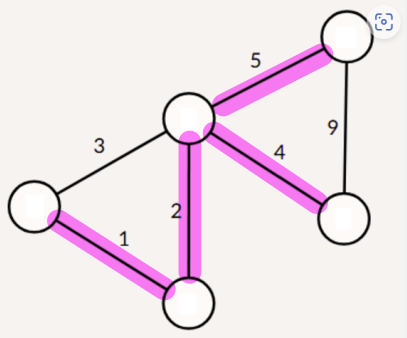
Wahr oder falsch: Wir haben $d(t) = 6$ nachdem die $\mathrm{Schranken\ verbessern}$-Funktion zweimal ausgeführt wurde.

Wählen Sie eine Antwort:

- a. Wahr.
- b. Falsch.
- c. Das hängt davon ab in welcher Reihenfolge die Knoten in der $\mathrm{Schranken\ verbessern}$-Funktion bearbeitet werden.



Im oben dargestellten gewichteten Graphen sind die Kanten eines (unvollständigen) minimalen Spannbaums, der während der Ausführung vom Prim-Algorithmus konstruiert wurde, markiert *(in dick und rot)*.
Welche Kante wird als Nächstes zum Baum hinzugefügt?



4

Der Boruvka-Algorithmus findet einen minimalen Spannbaum durch schrittweises Hinzufügen von Gruppen von Kanten.
Wie viele Kanten fügt der Boruvka-Algorithmus im ersten Schritt hinzu, wenn er auf den obigen gewichteten Graphen angewendet wird?

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph, und sei $F \subseteq E$ eine aufspannende Teilmenge *(d.h., für jeden Knoten $v \in V$ gibt es eine Kante $f \in F$, die v als Endpunkt hat).*

Wahr oder falsch: F kann keinen Kreis enthalten.

Bitte wählen Sie eine Antwort:

- Wahr
- Falsch

SERIE 9 - COMMON MISTAKES

Welche Laufzeit hat diese DP-Rekursion?

$$DP[1] = 0$$

$$DP[i] = \begin{cases} 0 & \text{falls } v_i \text{ Quelle} \\ 1 + \max_{(v_j, v_i) \in E} DP[j] & \text{sonst} \end{cases}$$

→ um in $O(|V|+|E|)$ zu liegen, muss man die Adjazenzliste umkehren

BELLMANN-FORD $O(|V| \cdot |E|)$

Idee: · Finden von kürzesten Wegen von einem Knoten zu allen anderen mit negativen Kantengewichten
· Finden von negativen Zyklen

Invariante: Nach der k-ten Iteration sind kürzeste Wege der Länge $\leq k$ korrekt berechnet

Pseudocode:

Startknoten

```
for v∈V\{s} do
    d[v] ← ∞; p[v] ← null      //initialisiere alle mit Distanz ∞ und noch keinen parent im shortest path tree
d[s] ← 0; p[s] ← null
for i←1...n-1 do
    for (u,v)∈E do
        if d[v] > d[u] + w((u,v)) then
            d[v] ← d[u] + w((u,v))
            p[v] ← u
for (u,v)∈E do
    if d[u] + w((u,v)) < d[v] then
        print("negativer Zyklus")
```

```java
//BELLMAN_FORD
int[] d = new int[n];
for(int i=0; i<n; i++){
  d[i] = Integer.MAX_VALUE;
}
d[s] = 0;
for(int i =0 ; i<Ev.get(s).size(); i++){
  d[Ev.get(s).get(i)] = Ew.get(s).get(i);
}

for(int k=0; k<n-1; k++){
  for(int i =0; i<n; i++){
    for(int j=0; j<Ev.get(i).size(); j++){
      d[Ev.get(i).get(j)] = Math.min(d[Ev.get(i).get(j)], d[i]+Ew.get(i).get(j));
    }
  }
}
```

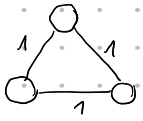
Minimale Spannbäume

Baum, der den Graphen „aufspannt", also mit minimalem Gewicht alle Knoten verbindet
→ immer $|V|-1$ Kanten

falls alle Kantengewichte paarweise verschieden sind ist der MST eindeutig,
ansonsten nicht unbedingt

Aufgabe

(b) Prove or disprove: For all vertices u, v of a graph G, the only path between u and v in an MST T of G is a shortest path between u and v in G.

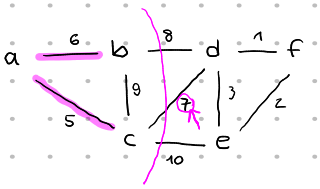


PRIM

$O((|V|+|E|)\cdot \log|V|)$

Idee: wähle Kante nach minimalem Gewicht nach „Schnittprinzip"

Pseudocode: Prim(G,s)

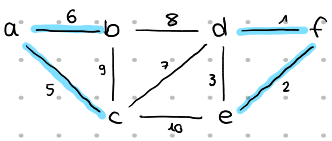


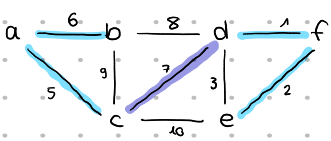

```
H ← makeHeap(V), S ← ∅
d[s] ← 0, d[v] ← ∞   ∀v ∈ V\{s}
decrease_key(H,s,0)
while H ≠ ∅:
    v* ← extract_min(H)
    S ← S ∪ {v*}
    for (v*,v) ∈ E, v ∉ S:
        d[v] ← min{d[v], w((v*,v))}
        decreasekey(H,v,d[v])
```

BORUVKA

$O((|V|+|E|)\log|V|)$

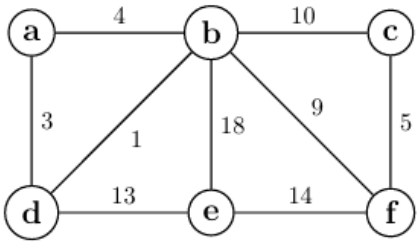
Idee: 1) finde für jede Komponente die günstigste Kante, die sie verlässt
2) füge die Komponenten zusammen





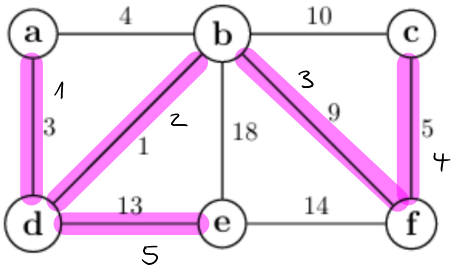
Aufgabe: Finde MST von folgendem Graphen mithilfe von a) Prim von a und b) Boruvka HS23

/ 2 P e) *Minimum Spanning Tree:* Consider the following graph:

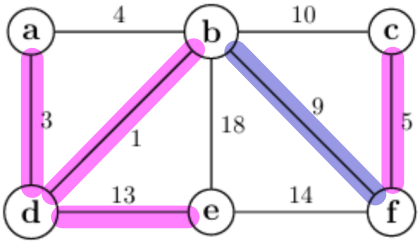


i) Highlight the edges that are part of the minimum spanning tree. (Either in the picture above, or you can recreate the graph below).

Prim von a



Boruvka



ii) Write out all positive integers x such that if we replace the weight 1 of edge $\{b, d\}$ in the above graph with x, then edge $\{b, d\}$ would be in at least one minimum spanning tree of the resulting graph.

1, 2, 3, 4

Riesen Graphenprüfungsaufgabe (T4) [HS23]

Disclaimer: muss in der Prüfung formaler aufgeschrieben werden

Theory Task T4. / 14 P

The city center of Amsterdam is known for its many tourists and many bridges. There are $n \geq 4$ tourist attractions in Amsterdam labeled $\{1, 2, \ldots, n\}$. For some pairs $i, j \in \{1, 2, \ldots, n\}$, the attractions i, j are connected by a two-way bridge, represented by $\{i, j\}$. The set of all bridges is denoted with $B = \{\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_m, j_m\}\}$. You may assume that $m \geq n$. Using the bridges, **any attraction i can be reached from any other attraction j.**

Remark: When asked to describe an algorithm, you need to address the following aspects:

1) the graph algorithm that you use as a subroutine in your solution;
2) the construction of the graph that you run this algorithm on;
3) the correctness of your solution, with a short justification (i.e., how and why does running the chosen graph algorithm on the graph you constructed solve the original problem);
4) the total running time of your solution, with a short justification.

You can directly use the algorithms covered in the lecture material, and you can directly use their running time bounds without proof.

Remark: It is not needed to solve part a) to attempt parts b) and c). It is not needed to solve any earlier parts to attempt part d).

/ 3 P

a) To accommodate for boats, the city government wants to remove some of the bridges. Each bridge $b_k = \{i_k, j_k\}$ has a *length* $L_k \in \mathbb{N}$. Describe an algorithm which outputs a subset $D \subseteq B$ of bridges, of largest possible total length $L(D) := \sum_{k:b_k \in D} L_k$, which can be removed while still making sure that tourists can travel between any pair of attractions $i, j \in \{1, 2, \ldots, n\}$ (without swimming). The running time of your algorithm should be $O(m \log m)$. iiiiiii HEAD
=======

1) Prim

2) ungerichteter, gewichteter Graph (zusammenhängend)
   Knoten ≙ Attraktionen
   Kanten ≙ Brücken mit Gewicht L_k

3) finde MST Prim und entferne diese, um D zu erhalten

4) da $n \leq m+1$ $O(m \log m)$

It turns out the bridges are historical buildings, and so they cannot be removed. Instead, the government decides to only open each bridge during some parts of the day. They divide a day into $N \geq 2$ units of time, labeled $\{0, 1, \ldots, N-1\}$, and give each bridge $b_k = \{i_k, j_k\}$ *opening times* $O_k \subseteq \{0, 1, \ldots, N-1\}$. Each bridge also has a *crossing time* $C_k \in \mathbb{N}$, which represents the number of time-units required to cross it. **Tourists are only allowed to start crossing a bridge when it is open** (but it is not a problem if it closes while they are crossing it). Note that a tourist's travel is allowed to span multiple days. For example, they might start to cross a bridge b_k with crossing time $C_k = 4$ at time $T = N-2$, and finish the next day at time $T' = 2$ (as long as $N - 2 \in O_k$).

/ 4 P

b) We are given two attractions $s, t \in \{1, 2, \ldots, n\}$, and a starting time $T_{\text{start}} \in \{0, 1, \ldots, N-1\}$. Describe an algorithm which decides if it is possible for tourists to travel from s to t, starting at time T_{start}, while respecting the opening times of the bridges, but **without having to wait** at any of the attractions. The running time of your algorithm should be $O(m \cdot N)$.

Hint: *Use a directed graph whose vertex set consists of N copies of the attractions $\{1, 2, \ldots, n\}$, labeled $\{(i, T) : i \in \{1, 2, \ldots, n\}, T \in \{0, 1, \ldots, N-1\}\}$. When should there be an edge from (i, T) to (i', T')?*

1) DFS

2) gerichteter, ungewichteter Graph

Knoten $\hat{=}$ N Kopien der Attraktionen

$V = \{(i,T) \mid i \in \{1,\ldots,n\}, T \in \{0,1,\ldots,N-1\}\}$

Kanten: $((i,T),(i',T')) \in E$ falls $b_k = \{i,i'\} \in B$

mit $T \in O_k$ $\quad T' \equiv_N (T + C_k)$

3) DFS$((s, T_{start}))$ $\rightarrow$ schauen ob (t,T) $T \in \{0,\ldots,N-1\}$ visited

4) $O(|V| + |E|) \rightarrow O(m \cdot N)$

$|V| = n \cdot N$

$|E| = O(mN)$

/ 4 P

c) We are given two attractions $s, t \in \{1, 2, \ldots, n\}$ and a starting time $T_{\text{start}} \in \{0, 1, \ldots, N-1\}$. Describe an algorithm which outputs the minimum amount of time (in time-units) required for tourists to travel from s to t, starting at time T_{start}. Now, **tourists are willing to wait** at an attraction for a bridge to open if this reduces their total travelling time. The running time of your algorithm should be $O(mN \cdot \log(mN))$.

Hint: *Use a weighted, directed graph with the same vertex set as in part b). Which edges should you add to model tourists waiting at an attraction for a bridge to open?* ¿¿¿¿¿¿¿ minor

1) Dijkstra

2) gerichtet, gewichtet
wie b)
Kanten: $((i,T),(i',T')) \in E$ falls Bed. b) mit Kosten C_k
und $\underbrace{((i,T),(i,(T+1)_{\bmod N}))}_{e} \in E$ mit $w(e) = 1$

3) ...

4) Knoten: $n \cdot N$
Kanten: $O(\underbrace{(m+n)}_{O(m)} \cdot N)$
$O(mN \cdot \log(mN))$

Based on negative feedback, the government decides to **open all bridges during the entire day**. In an attempt to steer traffic, they release a traveller's guide which rates how nice the view is from each bridge. As the view depends on the direction in which a bridge $b_k = \{i_k, j_k\}$ is crossed, the guide lists two numbers $R_k^+, R_k^- \in \mathbb{Z}$ (i.e, they are **possibly negative** integers). Assuming $i_k < j_k$, R_k^+ corresponds to crossing b_k from i_k to j_k, and R_k^- corresponds to crossing b_k from j_k to i_k. To avoid tourists getting lost, the government made sure that any walk around Amsterdam which starts and ends in the same attraction has total rating strictly less than 0.

/ 3 P

d) We are given two attractions $s, t \in \{1, 2, \ldots, n\}$. Describe an algorithm which outputs the **maximum** possible total rating of a walk from s to t. The running time of your algorithm should be $O(nm)$.

Hint: *Use a weighted, directed graph whose vertex set consists of the attractions* $\{1, 2, \ldots, n\}$. *(The opening times* O_k *and crossing times* C_k *do not play any role in this part of the exercise!)*

1) Bellman Ford

2) gerichteter, gewichteter Graph (negative Kantengewichte möglich)

Knoten $\hat{=}$ Attraktionen
Kanten $\hat{=}$ Brücken mit $-R_k^+, -R_k^-$

3) BF findet kürzesten s-t-Pfad, da Kantengewichte negiert, finden wir so s-t-Pfad mit maximalem Rating
- Anwendung von BF korrekt, da es keine negativen Zyklen geben kann

4) $O(n \cdot m)$