QUIZ-NACHBESPRECHUNG

Sei $G = (V, E)$ ein gerichteter, gewichteter Graph mit positiven Gewichten $w_e > 0$ für $e \in E$.
Seien $s, t \in V$. Wir haben gesehen, dass der Algorithmus von Dijkstra die Länge eines kürzesten Pfades von s nach t in Zeit $O((n+m)\log n)$ berechnet, wobei $n = |V|$ und $m = |E|$ ist.

Welche der folgenden Aussagen ist korrekt?

Wählen Sie eine Antwort:

- a. Mit dem Algorithmus von Dijkstra können wir die Länge der kürzesten Pfade zwischen s und t für alle $t \in V$ in Zeit $O((n+m)\log n)$ berechnen.
  (Insgesamt n Pfade)
- b. Mit dem Algorithmus von Dijkstra können wir die Länge der kürzesten Pfade zwischen s und t für alle $s \in V$ und alle $t \in V$ in Zeit $O(n \cdot (n+m)\log n)$ berechnen.
  (Insgesamt n^2 Pfade)
- c. (a) und (b) sind beide wahr.
- d. (a) und (b) sind beide falsch.

Sei $G = (V, E)$ ein gerichteter, gewichteter Graph mit $V = \{1, 2, 3, \ldots, n\}$.
Der Floyd-Warshall-Algorithmus verwendet Dynamische Programmierung mit dem Teilproblem:

$$d_{uv}^i = \text{Länge eines kürzesten Weges zwischen } u \text{ und } v, \text{ der nur Zwischenknoten aus } 1, 2, \ldots, i \text{ verwendet.}$$

Angenommen, G enthält einen negativen Zyklus. Welche der folgenden Aussagen muss wahr sein?

Wählen Sie eine Antwort:

- a. Es gibt ein $v \in V$ mit $d_{vv}^3 > 0$.
- b. Es gibt ein $v \in V$ mit $d_{vv}^n < 0$.
- c. Für alle $v, w \in V$ mit $v \neq w$ gilt $d_{vw}^n < 0$.
- d. Es gibt ein $v \in V$ mit $d_{vv}^i < 0$ für alle $i = 2, 3, \ldots n$.

Sei $G = (V, E)$ ein gerichteter, gewichteter Graph mit positiven Gewichten $w_e > 0$ für $e \in E$.
Seien $s, t \in V$, und sei P ein kürzester Pfad von s nach t in G.

Wahr oder falsch: Nach Erhöhung aller Gewichte w_e um eine Konstante $c > 0$, ist P weiterhin ein kürzester Pfad von s nach t.

c = 2

Bitte wählen Sie eine Antwort:

- Wahr
- Falsch

Sei $G = (V, E)$ ein gerichteter, gewichteter Graph ohne negative Zyklen. Seien $n = |V|$ und $m = |E|$.
Sowohl der Algorithmus von Johnson als auch der Floyd-Warshall-Algorithmus können verwendet werden, um kürzeste Pfade zwischen allen Paaren von Knoten in G zu finden.

Wahr oder falsch: Wenn $m = O(n)$, dann hat der Algorithmus von Johnson eine schlechtere asymptotische Laufzeit als der Floyd-Warshall-Algorithmus.

$O(n(m+n)\log n)$
$= O(n^2 \log n)$

$O(n^3)$

Bitte wählen Sie eine Antwort:

- Wahr
- Falsch

Sei $G = (V, E)$ ein gerichteter Graph mit den Knoten $V = \{1, 2, 3, 4\}$.
Sei A_G die *Adjazenzmatrix* von G. Angenommen, $(A_G)^3$ (d.h., *die dritte Potenz der Adjazenzmatrix)* ist gleich:

0	1	1	3
0	0	0	1
1	3	0	2
0	1	0	0

Wie viele gerichtete Zyklen mit genau der Länge 3 gibt es in G?

Antwort: 0

FLOYD-WARSHALL $O(n^3)$

Anwendung: finden von allen kürzesten Wegen, negative Zyklen finden

Idee: DP!

$d^i_{u,v}$:= Länge des kürzesten Weges von u nach v, dessen Zwischenknoten aus der Menge $\{1,\dots,i\} \subseteq V$ stammen

Rekursion: $d^i_{u,v} = \min\{ d^{i-1}_{u,v}, d^{i-1}_{u,i} + d^{i-1}_{i,v} \}$

- $d^{i-1}_{u,v}$: Knoten i kommt im kürzesten Weg nicht vor
- $d^{i-1}_{u,i} + d^{i-1}_{i,v}$: Knoten i kommt im kürzesten Weg vor

Base Case:

$$d^0_{u,v} = \begin{cases} 0 & \text{falls } u=v \\ c(u,v) & \text{falls } (u,v)\in E \\ \infty & \text{sonst} \end{cases}$$

G enthält negativen Zyklus $\Leftrightarrow \exists v \in V: d^n_{v,v} < 0$

FLOYD-WARSHALL$(G=(V,E),c)$

---

1 $d^0_{u,v} \leftarrow \begin{cases} 0 & \text{falls } u=v \text{ ist,} \\ c(u,v) & \text{falls } (u,v)\in E \text{ ist,} \\ \infty & \text{sonst} \end{cases}$ ▷ *Initialisierung*
2 **for** $i \leftarrow 1,\dots,n$ **do**
3 **for** $u \leftarrow 1,\dots,n$ **do**
4 **for** $v \leftarrow 1,\dots,n$ **do**
5 $d^i_{u,v} \leftarrow \min(d^{i-1}_{u,v}, d^{i-1}_{u,i} + d^{i-1}_{i,v})$
6 **return** d

```
//FLOYD_WARSHALL
//Initialisierung und Base Case
int[][] d = new int[n][n];
for(int i=0; i<n; i++){
  for(int j=0; j<n; j++){
    d[i][j] = Integer.MAX_VALUE/2;
    if(i==j) d[i][j] = 0;
  }
}
for(int i=0; i<n; i++){
  for(int j=0; j<Ev.get(i).size(); j++){
    d[i][Ev.get(i).get(j)] = Ew.get(i).get(j);
  }
}
//Rekursion
for(int k=0; k<n; k++){
  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      d[i][j] = Math.min(d[i][j], d[i][k] + d[k][j]);
    }
  }
}
```