Netzwerkalgorithmen
SoSe 20

Benedikt Lüken-Winkels

13. Oktober 2020

Inhaltsverzeichnis

1 Bellman/Ford - Allgemeiner Graph

Geben Sie den Bellman/Ford Algorithmus an und analysieren Sie seine Laufzeit.

Algorithmus 1 : Bellman/Ford

```
input : Graph G(V, E)
        Knoten s
        Kostenfunktion c
        Knotenarray DIST
        Knotenarray PRED
Queue Q;                                   //Kandidatenmenge
Nodearray count ← (G, 0) ;
for v ∈ G do
    DIST[v] ← ∞;
    PRED[v] ← NULL;
DIST[s] ← 0;
Q ← Q ∪ s;
while Q ≠ ∅ do
    u ← Q.pop();
    count[u] + +;
    if count[u] > |V| then
        return negativer Zyklus gefunden;
    for e ∈ u.out_edges() do
        v ← e.target();
        if DIST[u] + c(e) < DIST[v] then
            DIST[v] = DIST[u] + c(e);
            PRED[v] = e;
            Q ← Q ∪ v;
return kein negativer Zyklus gefunden ;
```

$PRED$ wird dazu verwendet, um nachher den kürzesten Weg zu einem Knoten nachverfolgen zu könnten. $PRED$-Verweise ändern sich daher, wenn sich der kürzeste Weg ändert.

Laufzeit

$$n \cdot \underbrace{(\text{Iterationen über alle Knoten}|\text{ausgehende Kanten})}_{O(\underbrace{\sum_{v \in V}(1 + outdeg(v))}_{n+m})}$$

$\Rightarrow O(n(n+m))$ als Gesamtlaufzeit. Ist der Graph zusammenhängend, so gilt $m \geq n-1$ und die Laufzeit wird von $m \cdot n$ dominiert $\Rightarrow O(nm)$

2 SSSP-Algorithmus ohne neagtive Zyklen - Dijkstra

Geben Sie einen SSSP-Algorithmus an, der O(n + m) als Laufzeit hat und auf nicht-negativen Netzwerken funktioniert.

Algorithmus 2 : dist - verfeinert

for $v \in G$ **do**
  $DIST[v] \leftarrow \infty$;
  $PRED[v] \leftarrow NULL$;
DIST[s] ← 0 ;
Priority Queue $PQ.insert(s, 0)$;
while $PQ \neq \emptyset$ **do**
  wähle Knoten $u \in PQ$ mit $DIST[u]$ minimal: $PQ.del\_min()$;
  for $v \in V$ *mit* $e = (u,v) \in E$ **do**
    $d \leftarrow DIST[u] + c(e)$;
    if $DIST[v] > d$ **then**
      if $DIST[v] = \infty$ **then**
        $PQ.insert(v, d)$;
      else
        $PQ.decrease\_p(v, d)$;
      $DIST[v] \leftarrow d$;
      $PRED[v] \leftarrow e$

Priority = Distanz

Laufzeit $O(n + m)$ durch perfekte Wahl.

- Operationen auf dem Graphen: $O(n + m)$
- Priority Queue

⇒ Gesamtlaufzeit $O(n \cdot (T_{insert}(n) + T_{delmin}(n) + T_{empty}(n)) + m \cdot T_{decrease}(n))$
m ist der Flaschenhals des Algorithmus.

Realisierung der Datenstruktur

- binärer Min-Heap
- balancierter Baum

⇒ $O((n + m) \log n)$

Mit Fibonacci Heap: Insert $O(1)$, Delmin $(\log n)$) ⇒ $O(n \log(n) + m)$ (Decrease $O(1)$,

3 SSSP-Algorithmus azyklische Netzwerke

Geben Sie einen SSSP-Algorithmus an, der O(n + m) als Laufzeit hat und auf azyklischen Graphen funktioniert.

Azyklische Graphen besitzen eine topologische Sortierung.

Algorithmus 3 : SSSP-Alogrithmus mit Topsort

input : Graph G, Knoten s, Kostenfunktion c, Knotenarray $DIST$

Knotenarray $INDEG(G, 0)$;

Laufzeit $O(n + m)$ durch perfekte Wahl mit topologischem Sortieren

4 SSSP-Algorithmus mit negativen Zyklen

Geben Sie einen SSSP-Algorithmus an, der negative Zyklen erkennt und ihn ausgibt. Wie ist die Laufzeit?

Bellman/Ford

5 Maxflow Labeling

Geben Sie einen Labeling-Algorithmus zur Lösung des Maxflow-Problems in Pseudo-Code an und analysieren Sie die Laufzeit.

6 Capacity-Scaling

Geben Sie CAPACITY-SCALING unter Verwendungen des Maxflow Labeling Algorithmus an. Begründen Sie die Laufzeit.

7 Feasible Flow

7.1 Berechnung des Feasible Flows

Definieren Sie feasible flow. Formulieren Sie einen Algorithmus zur Berechnung eines feasible flows.

7.2 Verwendung bei Maxflow

Wie kann man Maxflow damit lösen?

8 Min-Cost-Flow-Problem

Wie lautet die reduzierte Kosten-Optimalität für das Min-Cost-Flow-Problem? Folgern Sie aus ihrer Gültigkeit, dass im Restnetzwerk keine negativen Zyklen existieren.