

Algorithmik Blatt 7 Teil 1

Mtr.-Nr. 6329857

Universität Hamburg — 6. Dezember 2019

1 Aufgabe 19

```
1: procedure LISTAUGMENTINGPATHS(maxFlow)
2:   while |maxFlow| > 0 do
3:     path = FindSourceTargetPath(maxFlow)
4:     minEdge = FindLimitingEdge(path)
5:     output path, minEdge.flow
6:     foreach e in path.edges do
7:       ReduceFlow(maxFlow, e, minEdge.flow)
8:
```

2 Terminierung

3 Der Algorithmus Terminiert, weil sowohl die innere als auch die äußere Schleife terminieren:

- 4 • Die innere Schleife iteriert über eine endliche Anzahl Elemente, da der Pfad endliche Länge
5 hat
- 6 • Die äußere Schleife iteriert, weil in jeder iteration mindestens eine Kante (die mit dem nied-
7 rigsten Fluss in dem gefundenen Pfad) aus dem Fluss entfernt wird, der Fluss sich also mit
8 jeder Iteration verringert und nie erhöht.

9 In einem Fluss > 0 kann auch immer ein Pfad von der Quelle zur Senke gefunden werden. In einem
10 solchen Pfad existiert immer mindestens eine Kante, die den Fluss begrenzt.

11 Korrektheit

12 Der Algorithmus ist korrekt, weil der Maximale Fluss in jeder Iteration entlang eines gesamten Pfa-
13 des von der Quelle zur Senke um einen festen verringert wird. Der daraus entstehende Fluss ist also
14 wieder ein gültiger Fluss. Bezüglich des neuen Flusses ist dieser Pfad ein augmentierender Pfad der
15 wieder die Ausgangsfluss ergeben würde. Da dies für jede Iteration gilt und der Algorithmus mit
16 einem Fluss von 0 terminiert, kann ein Fluss von 0 entgegengesetzter Reihenfolge durch hinzuneh-
17 men der entfernten Pfade wieder zum maximalen Fluss schrittweise augmentiert werden.

18 Anzahl der Pfade

19 Da in jeder Iteration mindestens eine Kante aus dem Fluss entfernt wird, durchläuft der Algorithmus
20 maximale so viele Iterationen wie der Fluss, also auch der Netzwerk, Kanten hat. Pro Iteration wird
21 nur ein Pfad ausgegeben, also wird eine Folge von Pfaden gefunden, deren Länge durch die Anzahl
22 der Kanten im Netzwerk begrenzt ist.

23 Aufgabe 20

24 Wir betrachten möglichen Operationen des Push-Relabel Algorithmus gerennt voneinander:

25 **Relabel** Die Anzahl der Operationen ist, wie in der Vorlesung gezeigt, durch $2 \cdot |V|^2$ beschränkt.
26 $2 \cdot |V|^2 < |V| \cdot |E|$, weil wir einen zusammenhängenden Graphen betrachten.

27 **Sättigender Push** Ebenfalls wie in der Vorlesung gezeigt, liegt die Anzahl dieser Operationen in
28 $2 \cdot |V| \cdot |E|$

29 **Nicht-Sättigender Push** Jeder Push erhöht den Fluss über eine Kante um mindestens 1. Jede
30 Kante kann aber höchstens einen Fluss von $k-1$ bekommen bevor sie gesättigt ist. Eine Kante
31 kann also höchstens $k-1$ mal nicht-sättigend gepusht werden. Das Residuale Netzwerk hat
32 höchstens $2|E|$ Kanten, also können höchstens $2 \cdot (k-1) \cdot |E|$ nicht sättigende Pushes durch-
33 geführt werden.

Da Push-Relabel nur diese drei Operationen durchführt, werden insgesamt höchstens

$$2 \cdot |V|^2 + 2 \cdot |V| \cdot |E| + 2 \cdot (k-1) \cdot |E|$$

34 Operationen durchgeführt, die jeweils nur konstante Zeit benötigen.

Daraus ergibt sich eine obere Lauzeitschranke von

$$\mathcal{O}(k \cdot |V| \cdot |E|)$$

35 .