5.5 Preflow-Push

Idee:
Sende von s aus möglichst viel Fluss ins Netzwerk k und versuche ihn schrittweise bis zum Knoten t zu leiten. Lasse dabei Überschuss wieder zurück nach s fließen. Die Richtung findet man mit Hilfe von Distanz-Labels.

5.5.1 Definition

i) Ein Preflow x ist eine Funktion $x : E \to \mathbb{N}_0$ mit

   a) $0 \leq x_{ij} \leq u_{ij}$ Kapazitätsbedingung

   b) $\sum_{\substack{\text{für ein-}\\ \text{gehende}\\ \text{Kanten}}} x_{ji} - \sum_{\substack{\text{für aus-}\\ \text{gehende}\\ \text{Kanten}}} x_{ik} \geq 0 \; \forall \; i \in V \setminus \{s,t\}$

   (dh. eventuell mehr einfließender als abfließender Fluss)

ii) $e(i) := \sum x_{ji} - \sum x_{ik}$ heißt Überschuss (oder Excess) von i.
Ein Preflow mit $e(i) = 0 \; \forall \; i \in V \setminus \{s,t\}$ ist ein Flow.

iii) Ein Knoten $i \in V \setminus \{s,t\}$ mit $e(i) > 0$ heißt aktiv.

Idee für Algorithmus:

- Schiebe (push) Fluss von s nach t
  (von Knoten zu Knoten $\leftrightarrow$ erhöhender Pfad)
- Während des Ablaufs entsteht dadurch ein Preflow (mit Excess-Knoten)
- Am Ende soll der Preflow ein echter Flow sein.
  (dh. es existieren keine aktiven Knoten mehr)

Für die Richtung ($s \to t$) verwenden wir eine Distanzfunktion.

5.5.2 Distanz-Funktion

Die Distanz ist die Länge (= Anzahl) von Pfaden nach t.

Definition

Für einen Preflow x definieren wir in $G(x)$ eine Distanzfunktion $d = V \to \mathcal{N}_0$ mit

i) $d(t) = 0$

ii) $d(i) \leq d(j) + 1 \; \forall \; (i,j) \in G(x)$

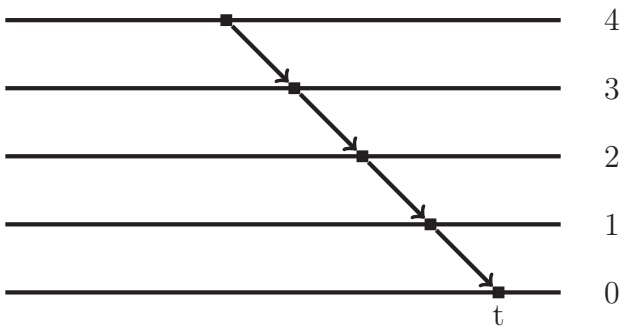
Beobachtung:

i) d ist untere Schranke für exakte Distanzwerte.

ii) $d = 0$ (Nullfunktion) ist gültige Distanzfunktion.

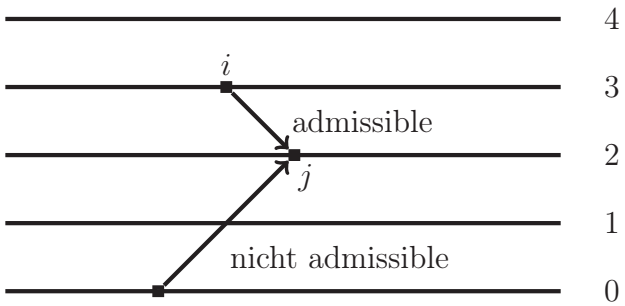
Intention:

$d(i)$ entspricht der Höhe oder dem Level des Knoten i.



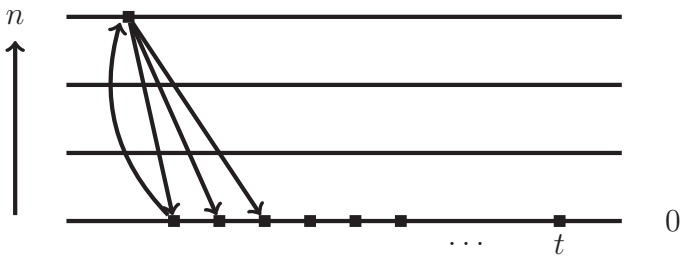
Definition admissible

Eine Kante (i,j) in $G(x)$ heißt admissible, wenn $d(i) = d(j) + 1$ dh. der Höhenunterschied genau 1 beträgt.



5.5.3 Preflow-Push-Algorithmus

1. Saturiere alle von s ausgehenden Kanten.
2. Setze $d(i) = \begin{cases} 0, & \text{für } i \neq s \\ n, & \text{für } i = s \end{cases}$



$G(x)$

Überschussknoten = alle Nachbarn von s

3. Betrachte einen aktiven Knoten i und schiebe Fluss über eine zulässige Kante
   $\rightarrow$ PUSH
4. Falls ein aktiver Knoten i keine zulässige, ausgehende Kante hat
   $\rightarrow$ RELABEL
   $d(i) \leftarrow min\{d(j)|j \text{ Nachbar in } G(x)\} + 1$

Der generische Preflow-Push-Algorithmus

Der generische Preflow-Push-Algorithmus nutzt keine besondere Strategie (wie z.B. highest-label, FIFO, ...) zur Auswahl aktiver Knoten.

Algorithmus 4 : Generischer Preflow-Push-Algorithmus

foreach $v \in V$ **do**
  $d(v) \to 0$
  $e_v \to 0$
end
foreach $(u, v) \in E$ **do**
  $x_{uv} \to 0$
end
foreach $j \in V$ *mit* $(s, j) \in E$ **do**
  $x_{sj} \leftarrow u_{sj}$ // volle Kapazität
  $e_j \leftarrow e_j + x_{sj}$
end
$d(s) \leftarrow n$
while $\exists$ *aktive Knoten (dh.* $e(i) > 0$*)* **do**
  Wähle einen aktiven Knoten i
  PUSH/RELABEL(i)
end

Algorithmus 5 : Push/Relabel(i)

if $\exists$ *admissible Edge* (i, j) *in* $G(x)$ **then**
  Wähle eine solche Kante (i, j)
  $\delta \leftarrow min\{e(i), r_{ij}\}$ Überschuss, Restkapazität
  $x_{ij} \leftarrow x_{ij} + \delta$
  $e(i) \leftarrow e(i) - \delta$
  $e(j) \leftarrow e(j) + \delta$
end
else
  $d(i) \leftarrow min\{d(j)|(i, j)$ in $G(x)\} + 1$
end

Laufzeit:
$\mathcal{O}(\#\text{ Pushes} + \#\text{ Relabels})$

Lemma 7
Falls $e(i) > 0$ (i aktiv), dann existiert ein Pfad von s nach i in $G(x)$.

Lemma 8
Für alle aktiven Knoten $i \in V$ gilt $d(i) < 2n$.
(Folgt aus Lemma 7)

Lemma 9
Es werden maximal $2n^2$ Relabels durchgeführt.

Beweis
Für jeden einzelnen Knoten werden maximal $2n$ Relabels durchgeführt.
(Folgt aus Lemma 8)

□

Lemma 10
Es werden maximal nm saturierende Pushes durchgeführt.

Lemma 11
Es werden maximal n^2m nicht-saturierende Pushes durchgeführt.

Theorem 4

Der generische Preflow-Push-Algorithmus hat eine Laufzeit von $\mathcal{O}(n^2 \cdot m)$