Lösungen zu Übungsblatt 4

1 Algorithmus zur Kreisbestimmung im Netzwerksimplex

Beim Netzwerksimplexverfahren wird in jeder Iteration eine nicht-basische Kante $e = (u, v)$ zur aktuellen Basislösung $T = (V, A_T)$ hinzugefügt. Da T ein Spannbaum ist, entsteht dadurch ein eindeutiger Kreis $C \subseteq A_T \cup \{e\}$. Ziel ist es, diesen Kreis effizient zu bestimmen, um Flussanpassungen und die Wahl der austretenden Kante zu ermöglichen.

1. Pseudocode zur Kreisbestimmung

Funktion `find_predecessors_levels`

- **Eingabe:**
  - `EDGES` – Liste aller Kanten im Graphen (gerichtet oder ungerichtet)
  - `TREE` – Liste von Indizes in `EDGES`, die die aktuelle Baumlösung A_T beschreiben
- **Ausgabe:**
  - `PRED` – Vorgängerfunktion: `PRED[v]` ist der Elternknoten von v im Baum
  - `LEVEL` – Baumtiefe jedes Knotens im Baum

Algorithmus:

1. Initialisiere Queue Q mit Wurzelknoten 0.
2. Setze `PRED`$[0] = -1$, alle anderen auf ünbesucht".
3. Setze `LEVEL`$[0] = 1$, alle anderen auf 0.
4. TEMP = Kantenmenge aus `EDGES`, deren Index in `TREE` enthalten ist.
5. Solange $Q \neq \emptyset$:
   - Entferne vordersten Knoten q aus Q.
   - Suche benachbarte Knoten v in `TEMP`, die über q erreichbar sind und noch keinen Vorgänger haben.
   - Setze für jeden solchen v: `LEVEL[v]` = `LEVEL[q]` + 1 und `PRED[v]` = q.
   - Füge diese v in Q ein.

Funktion `find_circle`

- **Eingabe:**
  - `ENTERING_ARC` – Kante $e = (u, v)$, die eingefügt werden soll
  - `PRED` – Vorgängerfunktion
  - `LEVEL` – Tiefen der Knoten
- **Ausgabe:**
  - Liste der gerichteten Kanten, die den Kreis C bilden

Algorithmus:

1. Initialisiere Liste `CIRCLE` mit `ENTERING_ARC`.
2. Vergleiche die Tiefen von u und v:
   - Falls u tiefer liegt: gehe über `PRED` von u zur Höhe von v und füge Kanten (`PRED[u]`, u) zur

Liste hinzu.
- Falls v tiefer liegt: analog über v zur Höhe von u.

3. Nun sind beide Knoten auf gleicher Höhe:
   - Bewege beide Knoten gleichzeitig über `PRED` nach oben, bis sie sich treffen (LCA).
   - Füge auf beiden Seiten die Kanten (`PRED[temp1]`, $temp1$) bzw. ($temp2$, `PRED[temp2]`) zum Kreis hinzu.
4. Entferne Platzhalter und gib den vollständigen Kreis zurück.

2. Erklärung des Verfahrens

1. Die Funktion `find_predecessors_levels` durchläuft den Baum $T = (V, A_T)$ mittels Breitensuche, um die Eltern (Vorgänger) und die Tiefe jedes Knotens im Baum zu bestimmen.
2. Mit diesen Informationen kann der Kreis, der durch das Einfügen der neuen Kante $e = (u, v)$ entsteht, effizient bestimmt werden.
3. Falls u und v auf unterschiedlichen Tiefen liegen, wird zuerst der tiefer liegende Knoten durch seinen Vorgänger so lange zurückverfolgt, bis beide Knoten auf gleicher Ebene im Baum liegen.
4. Danach werden beide Knoten gleichzeitig über ihre Vorgänger zurückverfolgt, bis sie sich treffen. Diese Schnittstelle ist der Least Common Ancestor (LCA).
5. Der Kreis besteht somit aus:
   - dem Pfad von u zur LCA (rückwärts im Baum),
   - der Kante $e = (u, v)$,
   - dem Pfad von v zur LCA (rückwärts im Baum).
6. Die Richtung der Kanten ist für die spätere Flussberechnung im Netzwerksimplex entscheidend.

3. Bedeutung im Netzwerksimplex

Die Bestimmung dieses Kreises ist zentral für das Netzwerksimplexverfahren, da:
- entlang des Kreises eine mögliche Flussanpassung durchgeführt wird (Zirkulation),
- über die reduzierten Kosten entschieden wird, ob die neue Basislösung günstiger ist,
- und festgestellt wird, welche Kante aus dem Baum entfernt werden muss, um eine gültige Basislösung zu erhalten.