

Script generated by TTT

Title: TÄubig: GAD (11.07.2013)

Date: Thu Jul 11 12:01:30 CEST 2013

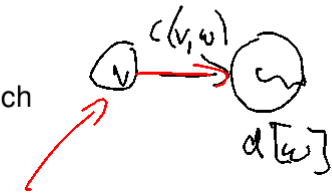
Duration: 44:20 min

Pages: 17

Bellman-Ford-Algorithmus

Zusammenfassung:

- **keine Distanzniedrigung** mehr möglich
($d[v] + c(v, w) \geq d[w]$ für alle w):
fertig, alle $d[w]$ korrekt für alle w
- **Distanzniedrigung möglich** selbst noch in n -ter Runde
($d[v] + c(v, w) < d[w]$ für ein w):
Es gibt einen negativen Kreis, also Knoten w mit Distanz $-\infty$.



Bellman-Ford-Algorithmus

```

BellmanFord(Node s) {
  foreach (v ∈ V) d[v] = ∞;
  d[s] = 0; parent[s] = s;
  for (int i = 0; i < n - 1; i++) { // n - 1 Runden
    foreach (e = (v, w) ∈ E)
      if (d[v] + c(e) < d[w]) { // kürzerer Weg?
        d[w] = d[v] + c(e);
        parent[w] = v;
      }
  }
  foreach (e = (v, w) ∈ E)
    if (d[v] + c(e) < d[w]) { // kürzerer Weg in n-ter Runde?
      infect(w);
    }
}

```

Bellman-Ford-Algorithmus

```

infect(Node v) { // -∞-Knoten
  if (d[v] > -∞) {
    d[v] = -∞;
    foreach (e = (v, w) ∈ E)
      infect(w);
  }
}

```

Gesamtlaufzeit: $O(m \cdot n)$

Bellman-Ford-Algorithmus

```

BellmanFord(Node s) {
  foreach (v ∈ V) d[v] = ∞;
  d[s] = 0; parent[s] = s;
  for (int i = 0; i < n - 1; i++) { // n - 1 Runden
    foreach (e = (v, w) ∈ E)
      if (d[v] + c(e) < d[w]) { // kürzerer Weg?
        d[w] = d[v] + c(e);
        parent[w] = v;
      }
  }
  foreach (e = (v, w) ∈ E)
    if (d[v] + c(e) < d[w]) { // kürzerer Weg in n-ter Runde?
      infect(w);
    }
}

```

Handwritten notes: $(n-1) < m$ and $\sum \frac{+1 \cdot m}{n \cdot m}$

Bellman-Ford-Algorithmus

```

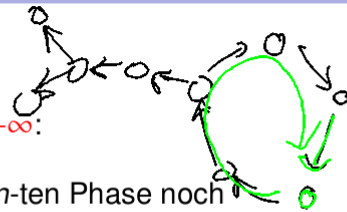
infect(Node v) { // -∞-Knoten
  if (d[v] > -∞) {
    d[v] = -∞;
    foreach (e = (v, w) ∈ E)
      infect(w);
  }
}

```

Gesamtlaufzeit: $O(m \cdot n)$

Bellman-Ford-Algorithmus

Bestimmung der **Knoten mit Distanz $-\infty$** :



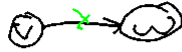
- betrachte alle Knoten, die in der n -ten Phase noch Distanzverbesserung erfahren
- aus jedem Kreis mit negativem Gesamtgewicht muss mindestens ein Knoten dabei sein
- jeder von diesen Knoten aus erreichbare Knoten muss Distanz $-\infty$ bekommen
- das erledigt hier die **infect**-Funktion
- wenn ein Knoten zweimal auftritt (d.h. der Wert ist schon $-\infty$), wird die Rekursion abgebrochen

Bellman-Ford-Algorithmus

Bestimmung eines **negativen Zyklus**:

- bei den oben genannten Knoten sind vielleicht auch Knoten, die nur an negativen Kreisen über ausgehende Kanten angeschlossen sind, die selbst aber nicht Teil eines negativen Kreises sind
- Rückwärtsverfolgung der **parent**-Werte, bis sich ein Knoten wiederholt
- Kanten vom ersten bis zum zweiten Auftreten bilden **einen** negativen Zyklus

Bellman-Ford-Algorithmus



Ursprüngliche Idee der Updates vorläufiger Distanzwerte stammt von Lester R. Ford Jr.

Verbesserung (Richard E. Bellman / Edward F. Moore):

- verwalte **FIFO-Queue** von Knoten, zu denen ein kürzerer Pfad gefunden wurde und deren Nachbarn am anderen Ende ausgehende Kanten noch auf kürzere Wege geprüft werden müssen
- wiederhole: nimm ersten Knoten aus der Queue und prüfe für jede ausgehende Kante die Distanz des Nachbarn
falls kürzerer Weg gefunden, aktualisiere Distanzwert des Nachbarn und hänge ihn an Queue an (falls nicht schon enthalten)
- Phase besteht immer aus Bearbeitung der Knoten, die **am Anfang** des Algorithmus (bzw. der Phase) in der Queue sind (dabei kommen während der Phase schon neue Knoten ans Ende der Queue)



Kürzeste einfache Pfade bei beliebigen Kantengewichten

Achtung!

Fakt

Die Suche nach kürzesten **einfachen** Pfaden (also ohne Knotenwiederholungen / Kreise) in Graphen mit beliebigen Kantengewichten (also möglichen negativen Kreisen) ist ein **NP-vollständiges Problem**.

(Man könnte Hamilton-Pfad-Suche damit lösen.)



All Pairs Shortest Paths

gegeben:

- Graph mit beliebigen Kantengewichten, der aber keine negativen Kreise enthält

gesucht:

- Distanzen / kürzeste Pfade zwischen allen Knotenpaaren

Naive Strategie:

- n -mal Bellman-Ford-Algorithmus (jeder Knoten einmal als Startknoten)

$$\Rightarrow O(n^2 \cdot m)$$

$\in O(n^3)$



All Pairs Shortest Paths

Bessere Strategie:

- reduziere n Aufrufe des Bellman-Ford-Algorithmus auf n Aufrufe des Dijkstra-Algorithmus

Problem:

- Dijkstra-Algorithmus funktioniert nur für nichtnegative Kantengewichte

Lösung:

- Umwandlung in nichtnegative Kantenkosten ohne Verfälschung der kürzesten Wege

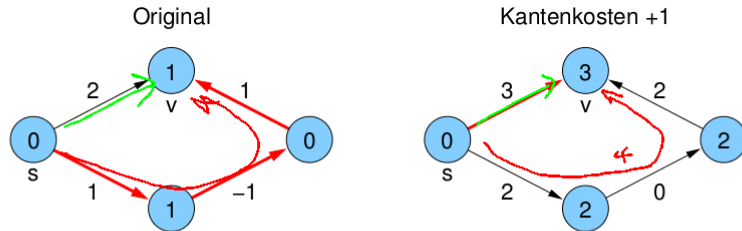


All Pairs Shortest Paths

Naive Idee:

- negative Kantengewichte eliminieren, indem auf jedes Kantengewicht der gleiche Wert c addiert wird

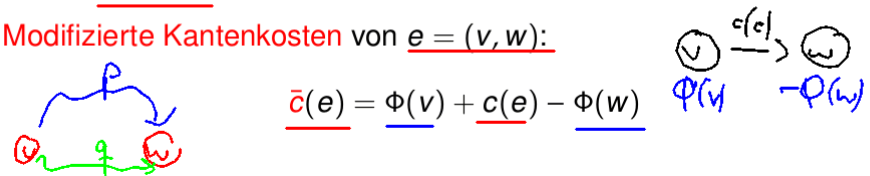
⇒ **verfälscht** kürzeste Pfade



All Pairs Shortest Paths

Sei $\Phi : V \mapsto \mathbb{R}$ eine Funktion, die jedem Knoten ein **Potential** zuordnet.

Modifizierte Kantenkosten von $e = (v, w)$:



$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$

Lemma

Seien p und q Wege von v nach w in G .

$c(p)$ und $c(q)$ bzw. $\bar{c}(p)$ und $\bar{c}(q)$ seien die aufsummierten Kosten bzw. modifizierten Kosten der Kanten des jeweiligen Pfads.

Dann gilt für jedes Potential Φ :

$$\bar{c}(p) < \bar{c}(q) \Leftrightarrow c(p) < c(q)$$

All Pairs Shortest Paths

Beweis.

Sei $p = (v_1, \dots, v_k)$ beliebiger Weg und $\forall i: e_i = (v_i, v_{i+1}) \in E$

Es gilt:

$$\begin{aligned} \bar{c}(p) &= \sum_{i=1}^{k-1} \bar{c}(e_i) \\ &= \sum_{i=1}^{k-1} (\Phi(v_i) + c(e_i) - \Phi(v_{i+1})) \\ &= \Phi(v_1) + c(p) - \Phi(v_k) \end{aligned}$$

*Handwritten note: $(v_1) \rightarrow (v_2) \rightarrow \dots \rightarrow (v_k)$
 $\Phi(v_1) + c(e_1) - \Phi(v_2)$
 $+ \Phi(v_2) + c(e_2) - \Phi(v_3)$*

d.h. modifizierte Kosten eines Pfads hängen nur von ursprünglichen Pfadkosten und vom Potential des Anfangs- und Endknotens ab. (Im Lemma ist $v_1 = v$ und $v_k = w$)

All Pairs Shortest Paths

Lemma

Annahme:

- Graph hat keine negativen Kreise
- alle Knoten von s aus erreichbar

Sei für alle Knoten v das Potential $\Phi(v) = d(s, v)$.

Dann gilt für alle Kanten e : $\bar{c}(e) \geq 0$

Beweis.

- für alle Knoten v gilt nach Annahme: $d(s, v) \in \mathbb{R}$ (also $\neq \pm\infty$)
- für jede Kante $e = (v, w)$ ist

$$\begin{aligned} d(s, v) + c(e) &\geq d(s, w) \\ d(s, v) + c(e) - d(s, w) &\geq 0 \\ \bar{c}(e) = \Phi(v) + c(e) - \Phi(w) &\geq 0 \end{aligned}$$

All Pairs Shortest Paths / Johnson-Algorithmus

- füge **neuen Knoten s** und Kanten (s, v) für alle v hinzu mit $c(s, v) = 0$
- ⇒ alle Knoten erreichbar
- berechne $d(s, v)$ mit Bellman-Ford-Algorithmus
- setze $\Phi(v) = d(s, v)$ für alle v
- berechne modifizierte Kosten $\bar{c}(e)$
- berechne für alle Knoten v die Distanzen $\bar{d}(v, w)$ mittels Dijkstra-Algorithmus mit modifizierten Kantenkosten auf dem Graph ohne Knoten s
- berechne korrekte Distanzen $d(v, w) = \bar{d}(v, w) + \Phi(w) - \Phi(v)$

All Pairs Shortest Paths / Johnson-Algorithmus

Beispiel:

