

## Dynamic Shortest Path für SSSP

VO Algorithm Engineering

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

11. VO

29.11.2005

## Dynamisches SSSP

Geg.: Gerichteter Graph  $G=(V,E)$  mit **positiven**

Kantenkosten  $c(e) \in \mathbb{R}$ , eine Senke  $t \in V$ , sowie eine  
Sequenz der folgenden Operationen:

- **delete\_edge(v,w,c)**: entferne die Kante  $(v,w)$  aus  $G$
- **insert\_edge(v,w,c)**: füge die Kante  $(v,w)$  in  $G$  ein
- **dist(v)**: gib die Distanz zwischen Knoten  $v$  und  $t$  zurück
- **path(v)**: gib den kürzesten Weg von  $v$  nach  $t$  aus, falls  
einer existiert.

Damit sind auch die folgenden Operationen simulierbar:

- **increase\_weight(e,c')**: erhöhe die Kosten von Kante  $e$   
auf  $c'$  (durch delete\_edge und insert\_edge)
- **decrease\_weight(e,c')**: reduziere die Kosten der Kante  $e$   
auf  $c'$  (durch insert\_edge einer parallelen Kante)

4

## Literatur für diese VO

- G. Ramalingam und T. Reps: On the computational complexity of dynamic graph problems. Theoretical Computer Science 158, 1996, 233-277

2

## Dynamische Komplexitätsmaße

- Amortisiert: Worst Case in Größe des Inputs, wobei der Durchschnitt über eine Folge von Operationen genommen wird.
- Worst Case in Größe der Änderungen bzgl. des Inputs und des Outputs

5

## Überblick

Ein alternatives Komplexitätsmaß

Einführung SP-Teilgraph für SSSP

Algorithmus Dynamic SSSP

Analyse: Korrektheit + Laufzeit

3

## Alternatives Komplexitätsmaß

Worst Case bzgl. der Änderungen des Inputs und des Outputs:

- Ein Knoten heißt **modifiziert (modified)**, wenn er oder eine inzidente Kante einen neuen Input-Wert bekommen hat oder eingefügt oder entfernt wurde.
- Ein Knoten heißt **betroffen (affected)**, wenn er entweder neu eingefügt wurde oder er durch die Änderung einen neuen Output-Wert bekommen hat.
- CHANGED sei die Menge aller modifizierten oder betroffenen Knoten
- Sei  $|\delta| = |\text{CHANGED}|$
- Sei  $\|\delta\| = |\delta| + \text{Anzahl der zu CHANGED inzidenten Kanten}$
- So viele Änderungen sind mindestens notwendig
- $\rightarrow$  notwendige Laufzeit

6

## Alternatives Komplexitätsmaß

Worst Case bzgl. der Änderungen des Inputs und des Outputs:

- Ein dynamischer Algorithmus heißt **beschränkt (bounded)**, wenn die benötigte Zeit für einen Update Schritt durch eine Funktion in  $\|\delta\|$  beschränkt ist.
- Sonst heißt er **unbeschränkt (unbounded)**.
- Ein dynamisches Problem heißt **unbeschränkt (unbounded)**, wenn kein dynamischer beschränkter Algorithmus existiert.

- **HIER: ein beschränkter dynamischer SSSP Algorithmus**

7

## Delete\_edge(v,w): $G \rightarrow G^c$

- Ein Knoten in  $G^c$  heißt **betroffen**, wenn der Wert  $\text{dist}_{G^c}(v) \neq \text{dist}_G(v)$  ist.
- Eine SP-Kante  $(x,y)$  heißt **betroffen** durch die `delete_edge(v,w)` Operation, wenn kein Pfad in  $G^c$  von  $x$  nach  $t$  existiert, der die Kante  $(x,y)$  benutzt und Länge  $\text{dist}_{\text{alt}}(x)$  besitzt.

10

## Definitionen

- Ein Teilgraph  $T$  von  $G$  heißt **kürzester-Wege-Baum** für  $G$  mit Senke  $t$ , falls
  - $T$  ist ein gerichteter Baum mit Wurzel  $t$
  - $V(T)$  ist die Menge aller Knoten, die  $t$  erreichen können, und
  - für jede Kante in  $T$  gilt:  $\text{dist}(u) = \text{dist}(v) + c(u,v)$

8

## Delete\_edge(v,w): $G \rightarrow G^c$

Beobachtungen in  $\text{SP}(G)$

- $(x,y)$  ist betroffen  $\Leftrightarrow y$  ist betroffen
- Knoten  $x \neq v$  ist betroffen  $\Leftrightarrow$  alle ausgehenden SP-Kanten von  $x$  sind betroffen
- Knoten  $v$  selbst ist betroffen  $\Leftrightarrow (v,w)$  die einzige ausgehende SP-Kante ist

11

## Definitionen

- Eine Kante heißt **SP-Kante**, wenn sie auf einem kürzesten Weg von  $v$  nach  $t$  liegt für ein  $v \in V$ .
- Eine Kante ist also genau dann SP-Kante, wenn gilt:  $\text{dist}(u) = \text{dist}(v) + c(u,v)$
- Sei  $\text{SP}(G)$  der durch die Menge aller SP-Kanten induzierte Teilgraph von  $G$  (der kürzeste-Wege Teilgraph).
- Jeder kürzeste Weg in  $G$  ist in  $\text{SP}(G)$  enthalten und umgekehrt:
- Jeder Weg in  $\text{SP}(G)$  ist ein kürzester Weg in  $G$ .
- Da die Kantengewichte alle positiv sind, ist  $\text{SP}(G)$  ein gerichteter, azyklischer Graph.

9

## Algorithmus Delete\_edge(v,w)

Phase 1:

- Bestimme die Menge aller betroffenen Knoten und Kanten
- Entferne die betroffenen Kanten von  $\text{SP}(G)$

Phase 2:

- Berechne neue  $\text{dist}$ -Werte aller betroffenen Knoten
- Aktualisiere  $\text{SP}(G)$

12

## Phase 1 von Delete\_edge(v,w)

- Menge der betroffenen Knoten = Menge der Knoten, die im Graphen  $SP(G)-(v,w)$  keine Verbindung zur Senke  $t$  besitzen.

### Algorithmus-Idee:

- Aufruf von Top-Sort, wenn  $outdegree$  von  $v$  in  $SP(G^*) = 0$ .
- Speichere  $SP(G)$  durch Adjazenzlisten:  $in(v)+out(v)$ , oder
- nicht explizite Speicherung, sondern durch Test:  $dist(x) == dist(y)+c(x,y)$

13

## Algorithmus Insert\_edge(v,w)

### Beobachtung:

- Wenn  $u$  betroffen ist, dann sehen alle kürzesten Wege in  $G^*$  von  $u$  nach  $t$  folgendermaßen aus:
- (kürzester  $(u,v)$ -Weg, Kante  $(v,w)$ , kürzester  $(w,s)$ -Weg)

$$u \text{ ist betroffen} \Leftrightarrow dist_G(u,v)+c(v,w)+dist_G(w) < dist_G(u)$$

Betrachte den kürzeste-Wege Baum  $T_v$  für Knoten  $v$  (Wurzel):

- Sei  $x$  beliebiger Knoten,  $u$ : Elter von  $x$  in  $T_v$
- Wenn  $x$  betroffen ist, dann muß auch  $u$  betroffen sein, denn sonst: ex. kürzester Weg  $P$  von  $u$  nach  $t$  ohne Kante  $(v,w)$ ; dann ist Weg  $((x,u),P)$  auch kürzester Weg;  $x$  wäre nicht betroffen.
- Menge der betroffenen Knoten bilden zusammenhängenden Teilbaum von  $T_v$  mit Wurzel  $v$

16

## Phase 2 von Delete\_edge(v,w)

- Sei  $A$  die Menge der nicht-betroffenen Knoten
- Ziel: Berechne  $dist(x)$  für alle Knoten in  $B=V \setminus A$

### Algorithmus-Idee:

- Kontrahiere  $A$  zu einem Knoten  $t'$  und ersetze die Kanten  $(x,y)$  durch  $(x,t')$  mit Kosten  $c(x,y)+dist(y)$
- Aufruf von Dijkstra für den neuen Graphen

14

## Algorithmus Insert\_edge(v,w)

Idee: Benutze Dijkstra's Algorithmus für Menge der betroffenen Knoten mit Key:  $dist(x)-dist(v)$  für alle  $x$

$$u \text{ ist betroffen} \Leftrightarrow dist_G(u,v)+c(v,w)+dist_G(w) < dist_G(u)$$

Betrachte den kürzeste-Wege Baum  $T_v$  für Knoten  $v$  (Wurzel):

- Sei  $x$  beliebiger Knoten,  $u$ : Elter von  $x$  in  $T_v$
- Wenn  $x$  betroffen ist, dann muß auch  $u$  betroffen sein, denn sonst: ex. kürzester Weg  $P$  von  $u$  nach  $t$  ohne Kante  $(v,w)$ ; dann ist Weg  $((x,u),P)$  auch kürzester Weg;  $x$  wäre nicht betroffen.
- Menge der betroffenen Knoten bilden zusammenhängenden Teilbaum von  $T_v$  mit Wurzel  $v$

17

## Laufzeitanalyse

- Phase 1:
    - Anzahl der Iterationen:  $|AFFECTED|$
    - Eine Iteration benötigt  $O(|Pred(u)|)$
    - Gesamt:  $O(|AFFECTED|)$
  - Phase 2: mit Fibonacci Heaps
    - Insert + Decrease\_key:  $O(1)$  amortisiert
    - ExtractMin  $O(\log p)$  mit  $p$ =Anzahl der Elemente im Heap
    - Anzahl der Iterationen: höchstens  $|AFFECTED|$
    - Gesamt:  $O(|AFFECTED|+|AFFECTED| \log |AFFECTED|)$
- Laufzeit:  $O(|\delta|+|\delta| \log |\delta|)$

15

## Algorithmus Insert\_edge((v,w),c)

- Insert edge  $(v,w)$  in  $G$ , setze Kosten  $c(v,w)$ ,  $Q=\emptyset$
- Falls  $c(v,w)+dist(w)<dist(v)$  dann:
  - $dist(v)=c(v,w)+dist(w)$
  - InsertHeapQ( $v,0$ )
- Sonst Falls  $c(v,w)+dist(w) == dist(v)$ 
  - Füge  $(v,w)$  in  $SP(G)$  ein;  $outdeg(v)++$ ;

18

## Algorithmus Insert\_edge((v,w),c) ff

- Solange  $Q \neq \emptyset$ :
  - $u \leftarrow \text{ExtractMin}(Q)$
  - Entferne alle ausgehenden Kanten von  $u$  in  $SP(G)$  und update  $\text{outdeg}(u)$
  - Für jeden Knoten  $x \in \text{Succ}(u)$ :
    - Falls  $c(u,x) + \text{dist}(x) == \text{dist}(u)$  dann:
      - Füge  $(u,x)$  in  $SP(G)$  ein und aktualisiere  $\text{outdeg}(u)$
  - Für jeden Knoten  $x \in \text{Pred}(u)$ :
    - Falls  $c(x,u) + \text{dist}(u) < \text{dist}(x)$  dann:
      - $\text{dist}(x) = c(x,u) + \text{dist}(u)$
      - Aktualisiere  $\text{HeapQ}(x, \text{dist}(x) - \text{dist}(v))$
    - Sonst Falls  $c(x,u) + \text{dist}(u) == \text{dist}(x)$ 
      - Füge  $(x,u)$  in  $SP(G)$  ein und aktualisiere  $\text{outdeg}(x)$

19

## Laufzeitanalyse

- mit Fibonacci Heaps
  - Insert + Decrease\_key:  $O(1)$  amortisiert
  - ExtractMin  $O(\log p)$  mit  $p = \text{Anzahl der Elemente im Heap}$
  - Anzahl der Iterationen: höchstens  $|\text{AFFECTED}|$
  - Gesamt:  $O(|\text{AFFECTED}| + |\text{AFFECTED}| \log |\text{AFFECTED}|)$

– Laufzeit:  $O(|\delta| + |\delta| \log |\delta|)$

20

## Zusammenfassung

- Theorem: Es gibt einen beschränkten Dynamischen Algorithmus für das SSSP-Problem mit Laufzeit  $O(|\delta| + |\delta| \log |\delta|)$  für jede Update-Operation  $\text{Delete\_edge}(v,w)$  und  $\text{Insert\_edge}(v,w)$ .

ENDE

21