Algorithmen und Datenstrukturen (für ESE) WS 2010 / 2011

Vorlesung 11, Montag, 17. Januar 2011

(Kürzeste Wege, Dijkstras Algorithmus, A*)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ab jetzt Übungstermine nur noch auf Anfrage
  - Ihre Erfahrungen mit dem Ü10 (Breitensuche etc.)
- Kürzeste Wege
  - Dijkstras Algorithmus zur Berechnung des kürzesten Weges zwischen zwei Knoten (in einem Graphen mit Kantenkosten)
  - Idee + Programm + Korrektheitsbeweis
  - **Übungsaufgabe (Ü11):** dasselbe für den A*-Algorithmus

Ihre Erfahrungen mit dem Ü10 (Graphen)

UNI FREIBURG

- Zusammenfassung von Ihrem Feedback [17.1 14:53]
  - Aufgabe 1 (Breitensuche implementieren) war wenig Code, aber einige fanden es "kniffliger als gedacht"
  - Aufgabe 3 (Zufallsgraphen erzeugen) fanden manche schwer verständlich
  - Ansonsten fanden es die meisten angemessen
  - Nächste Mal (heute) gerne etwas über den A*-Algorithmus

Kürzeste Wege (Wiederholung)

UNI FREIBURG

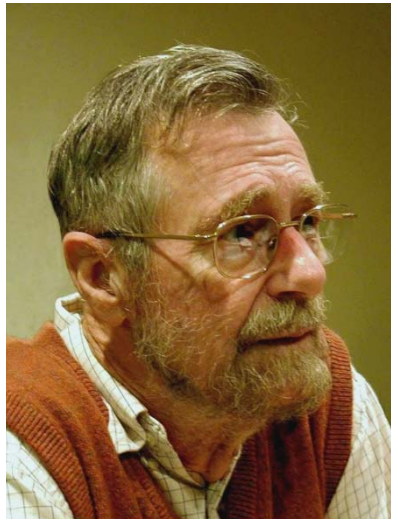
- Für einen Graphen $G = (V, E)$
  - Ein Pfad in G ist eine Folge $u_1, u_2, u_3, \ldots, u_l \in V$ mit
    - $(u_1,u_2), (u_2,u_3), \ldots, (u_{l-1},u_l) \in E$ [gerichteter Graph]
    - $\{u_1,u_2\}, \{u_2,u_3\}, \ldots, \{u_{l-1},u_l\} \in E$ [ungerichteter Graph]
  - Falls die Kanten Gewichte habe, ist die Summe der Gewichte dieser Kanten das (Gesamt)-Gewicht des Pfades
    - bzw. die Länge des Pfades oder Kosten des Pfades
  - Der kürzeste Pfad (engl. shortest path) zwischen zwei Knoten u und v ist der Pfad $u, \ldots, v$ mit der kürzesten Länge
  - Bei Kantenkosten sagt man auch billigster Pfad

Dijkstras Algorithmus

UNI FREIBURG

- Idee
  - Sei s der Startknoten und seien cost(s,u) die Kosten des billigsten Pfades von s zu u, für alle Knoten u
  - Besuche die Knoten in der Reihenfolge der cost(s,u)

    (Wir werden gleich sehen: wenn alle Kantenkosten = 1 sind, ist das genau Breitensuche)
- Ursprung
  - Edsger Dijkstra (1930 – 2002), niederländischer Informatiker, einer der wenigen Europäer, die den Turing-Award gewonnen haben (für seine Arbeiten zur strukturierten Programmierung)
  - Der Algorithmus ist aus dem Jahr 1959

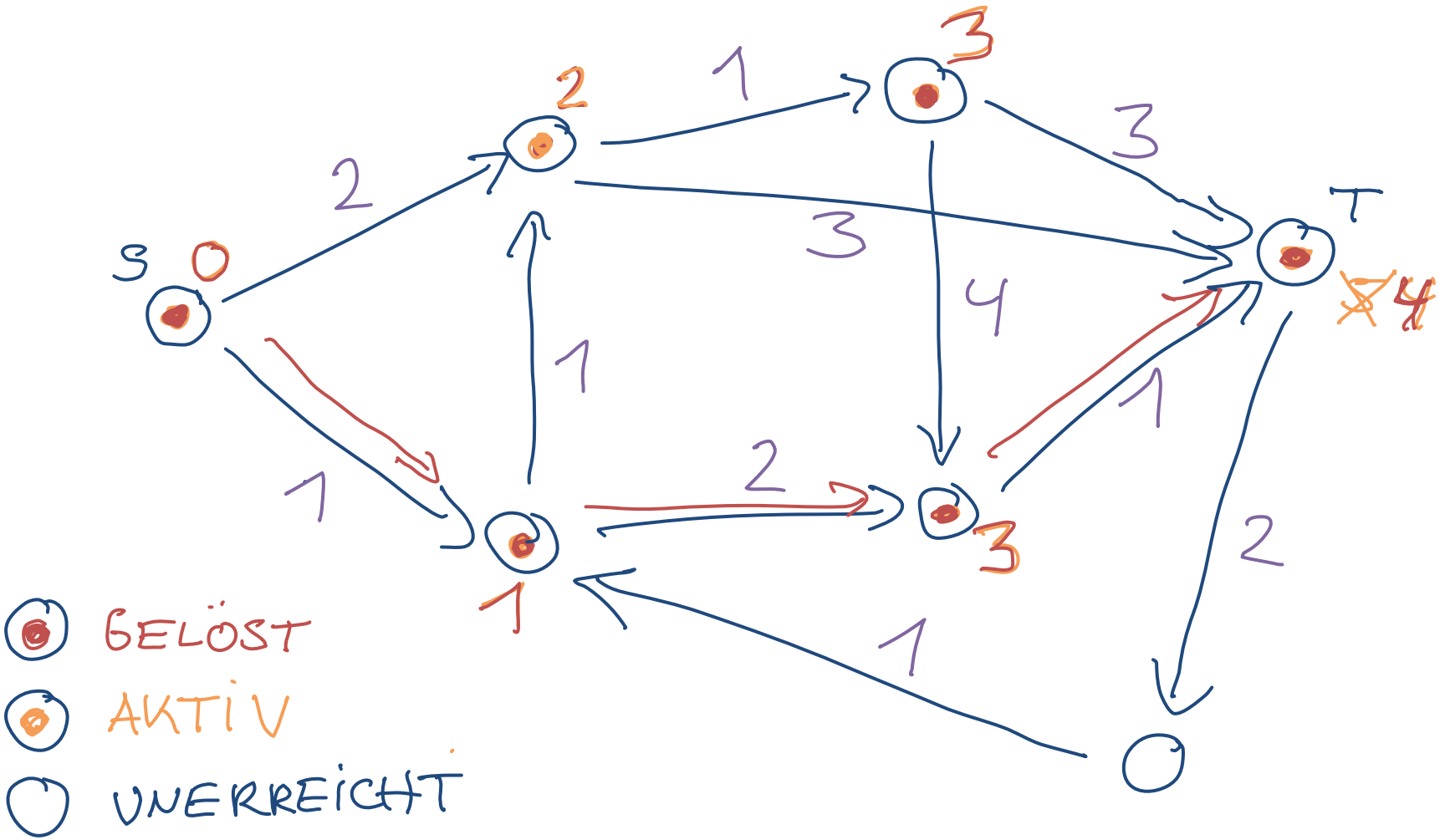


Dijkstras Algorithmus

UNI FREIBURG

- Algorithmus (high-level Beschreibung)
  - Drei Arten von Knoten: gelöste, aktive, und bisher unerreichte (englisch: settled, active, unreached)
    - für die gelösten Knoten u kennen wir cost(s, u)
    - für die aktiven Knoten haben wir einen Pfad mit vorläufigen Kosten cost(u) ≥ cost(s,u)
    - die unerreichten Knoten haben wir noch nicht erreicht
  - In jeder Runde holen wir uns den aktiven Knoten u mit den kleinsten vorläufigen Kosten cost(u) *
  - Für jeden Nachbarn v von u prüfen wir, ob wir v über u billiger erreichen können als bisher, das sog. Relaxieren von u
  - Den Knoten u betrachten wir dann als gelöst
  - Das machen wir solange bis wir bei * den Zielknoten erreichen

Dijkstras Algorithmus



Dijkstras Algorithmus

UNI FREIBURG

- Korrektheitsbeweis + Laufzeit
  - **Lemma:** In jeder Runde gilt für den aktiven Knoten u mit der kleinsten vorläufigen Distanz, dass cost(u) = cost(s, u)
    - Voraussetzung: alle Kantengewichte sind ≥ 0
  - Daraus folgt die Korrektheit:
    - Wir machen ja so lange, bis der Zielknoten der aktive Knoten mit der kleinsten vorläufigen Distanz ist
  - Daraus folgt für die Anzahl Operation auf der PW:
    - höchstens n Aufrufe von getMin / deleteMin n = #Knoten
    - höchstens m Aufrufe von insert m = #Kanten
  - Die Laufzeit ist somit $O((n + m) \cdot \log n)$
    - Mit einer besseren PW geht auch $O(n \cdot \log n + m)$

Dijkstras Algorithmus

$cost(u) \leq cost(v) + cost(v,u) = cost(s,v) + cost(v,u)$
$= cost(s,u)$

- Beweis des Lemmas:
  - Widerspruchsbeweis: wir nehmen an, dass es eine Runde im Algorithmus gibt, wo für den aktiven Knoten u mit der kleinsten vorläufigen Distanz cost(u) gilt, dass cost(u) ≠cost(s, u) und damit cost(u) > cost(s, u)
  - Wir betrachten die Runde und den Knoten u, wo das zum ersten Mal passiert:

$cost(u) > cost(s,u)$

$cost(u) = cost(v) + cost(v,u)$
$cost(v) < cost(u)$
> 0

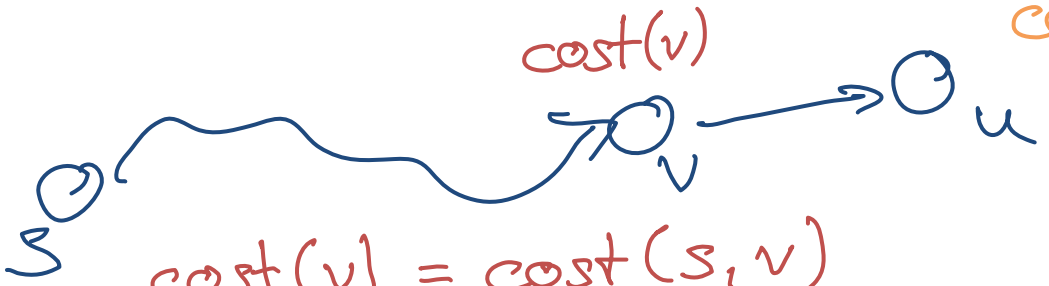
Als v dran war: $cost(v) = cost(s,v)$
(weil v der erste Knoten ist, bei dem es schief geht)

Versuch, das Lemma direkt zu beweisen:

wäre dann mit Induktion

Induktionsbehauptung: Für alle gelösten Knoten u gilt $cost(v) = cost(s,v)$.
für die Runden $1, 2, \dots, i$.

Runde $i+1$: Sei u der ~~kleinste~~ aktive Knoten mit dem kleinsten $cost(u)$.



$cost(v) = cost(s,v)$

$cost(u) \leq cost(v) + cost(v,u)$

$= cost(s,v) + cost(v,u) = cost(s,u)$

Dijkstras Algorithmus

UNI FREIBURG

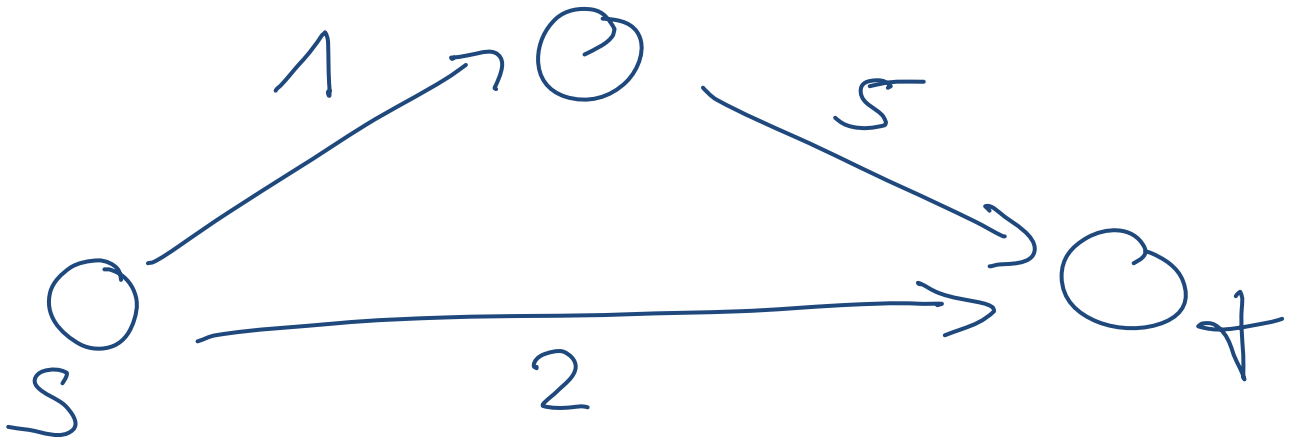
- Kommentare
  - Bevor Dijkstras Algorithmus den Zielknoten t erreicht, mit Kosten $c(s, t)$, hat er die kürzesten Wege zu allen Knoten u mit $c(s, u) < c(s, t)$ berechnet
  - Dijkstras Algorithmus löst damit nicht nur das sogenannte single source single target shortest path Problem, sondern gleich das sogenannte single source all targets Problem
  - Das hört sich verschwenderisch an, es gibt aber für allgemeine Graphen keine (viel) bessere Methode
  - Bei negativen Kantenkosten kann es negative Zyklen geben; um mit denen umzugehen braucht man andere Algorithmen (z.B. den Bellman-Ford Algorithmus)

A*-Algorithmus

- Idee
  - Zusatzinformation: für jeden Knoten u eine Abschätzung h(u) für die Kosten cost(u, t) von u zum Zielknoten t
  - Der A*-Algorithmus macht dann im Prinzip dasselbe wie Dijkstras Algorithmus
    - nur dass die Schlüssel mit dem die Knoten in die PW gesteckt werden, nicht die vorläufigen Distanzen cost(u) sind, sondern die Summe cost(u) + h(u)
  - Das funktioniert unter folgenden Voraussetzungen
    - $h(u) \leq cost(u, t)$ für alle u (h ist zulässig)
    - $h(u) \leq cost(u, v) + h(v)$ für alle u und v (h ist monoton)
  - Implementierung + Korrektheitsbeweis analog zu Dijkstra
    - das ist Ihre Übungsaufgabe für diese Woche

A*-Algorithmus

- Beispiel



Literatur / Links

- Kürzeste Wege und Dijkstras Algorithmus
  - In Mehlhorn/Sanders:

    10 Shortest Paths
  - In Wikipedia

    http://en.wikipedia.org/wiki/Shortest_path_problem

    http://en.wikipedia.org/wiki/Dijkstra's_algorithm