

Kap. 6.5: Minimale Spannbäume ff



Professor Dr. Petra Mutzel
 Karsten Klein
 Lehrstuhl für Algorithm Engineering, LS11
 Fakultät für Informatik, TU Dortmund

20. VO 2. TEIL DAP2 SS 2009 2. Juli 2009

Überblick

- 6.5: Minimale Spannbäume (MST) ff:
 – Algorithmus von **Prim**

Grundideen

- MST: Algorithmus von **Prim**
- Kürzeste Wege in Graphen (s. nächste Vorlesung): Algorithmus von **Dijkstra**

Beide nutzen ähnliches Konzept:

- **Greedy** (*gierig, gefräßig*) später mehr dazu
- Priority Queue
- Lassen einzelnen Baum wachsen

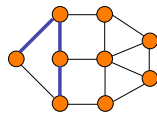
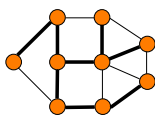
Algorithmus von Prim

basiert auf MST Eigenschaft:

MST Eigenschaft

basiert auf:

Aussichtsreiche Menge T von Kanten:
 Es existiert ein MST, der alle Kanten von T enthält.



MST Eigenschaft

Lemma 1 (Wiederholung):

Sei $G=(V,E)$ zshgd. mit Gewichtsfunktion $w : E \rightarrow \mathbb{R}$.
 Seien:

- $S \subset V$
 - $T \subset E$ aussichtsreich und *keine* Kante aus T verlässt S
 - $e \in E$ Kante mit minimalem Gewicht, die S verlässt
- Dann ist $T \cup \{e\}$ aussichtsreich.

PRIMs Algorithmus: lasse Baum von s aus wachsen mit MST Eigenschaft

Kantenmenge T, aussichtsreich
Kante e verläßt S mit min. Gewicht

Knotenmenge S
Dann: T ∪ e aussichtsreich

MST sind nicht eindeutig!!

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 15

Algorithmus von Prim

Vgl. mit Kruskal:

- Kruskal läßt Wald wachsen mit Union-Find bis Spannbaum erzeugt
- Prim läßt von Startknoten s einen einzelnen Baum wachsen

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 16

Algorithmus von Prim

Idee:

- Verwalte Menge **S** von Knoten in Unterbaum, ausgehend von **s**, **T** leer
- Wähle billigste **S** verlassende Kante $e=(u,v), v \in S, e \in T$
- ⇒ (Lemma 1) **T** aussichtsreich
- Stop, falls **S=V** ⇒ **T** bildet MST

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 17

Schema: Algorithmus von Prim

Wähle Startknoten $s \in V$
 $S := \{s\}; T := \emptyset$ // **S** Unterbaumknoten, **T** Kanten

while $S \neq V$ **do** {
 Wähle $e = (u, v)$, e verläßt **S** mit *min. Gewicht*
 $T := T \cup \{e\}; S := S \cup \{v\}$
}

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 18

Korrektheit Prim

Induktion über Kanten e_1, \dots, e_k , die zu **T** hinzugefügt werden:
 Zeige: $T_i = \{e_1, \dots, e_i\}$ aussichtsreich für $0 \leq i \leq k$

$i = 0: T_0 = \emptyset \Rightarrow$ aussichtsreich

$1 \leq i \leq k: T_{i-1}$ aussichtsreich, **S** Knotenmenge vor Hinzunahme von e_i , keine Kante aus T_{i-1} verläßt **S**
 ⇒ e_i leichteste **S** verlassende Kante
 ⇒ **Mit Lemma 1: $T_i = T_{i-1} \cup \{e_i\}$ aussichtsreich**
 Und: Keine Kante aus T_i verläßt neues **S**

Am Ende hat **T** genau $|V|-1$ Kanten ⇒ MST

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 19

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
 In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 20

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 21

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 22

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 23

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 24

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 25

Algorithmus von Prim

Knotenmenge S, Kantenmenge T:
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 26

Algorithmus von Prim

Knotenmenge S , Kantenmenge T :
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 27

Algorithmus von Prim

Knotenmenge S , Kantenmenge T :
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 28

Algorithmus von Prim

Knotenmenge S , Kantenmenge T :
In Teilbaum T_i

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 29

Schema: Algorithmus von Prim

Wähle Startknoten $s \in V$
 $S := \{s\}; T := \emptyset$ // S Unterbaumknoten, T Kanten
while $S \neq V$ **do** {
 Wähle $e = (u, v)$, e verläßt S mit *min. Gewicht*
 $T := T \cup \{e\}; S := S \cup \{v\}$
}

Effiziente Realisierung?

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 30

Realisierung von Prim

- Testen auf Kreise (Union-Find Partition) wie bei Kruskal nicht nötig
- Kein Sortieren nötig
- Auswahl leichtester Kante nötig
⇒ Priority Queue
- Wir speichern nicht die Kanten selbst, sondern die Knoten $v \in V \setminus S$. Priorität ist das Gewicht der Kante von S nach v

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 31

Dazu: ADT Priority Queue

- Dyn. Verwaltung von Elementmenge mit *Prioritäten*
- Operationen:
 - INSERT(p, v) : Position // Knoten v , Priorität p
 - DELETE(pos)
 - MINIMUM() : Position
 - EXTRACTMIN() : $P \times V$ // Rückg: min-Knoten und prio
 - DECREASEPRIORITY(pos, p) // neue Priorität p
- Bekannt: Binary (Min-)Heap in Array

tu technische universität dortmund AE Petra Mutzel DAP2 SS09 32

Pseudo-Code: Initialisierung

```

G=(V, E), float w[E]           // Graph und Gewichte
(1) var  $\pi[V]$ , PriorityQueue Q, pos[V] //  $\pi$ : Vorgänger in MST
                                // pos: Position in Q
(2) for each  $u \in V \setminus \{s\}$  do {
(3)   pos[u] := Q.INSERT( $\infty$ , u)
(4) }
(5) pos[s] := Q.INSERT(0, s)
(6)  $\pi[s] := nil$ 

```

Pseudo-Code Prim

```

(7) while not Q.ISEMPTY() do // über alle Knoten
(8)   ( $p, u$ ) := Q.EXTRACTMIN()
(9)   pos[u] := nil // gewählten Knoten entfernen
(10)  for all  $e = (u, v) \in E(u)$  do { // ausgehende Kanten
(11)   if pos[v]  $\neq nil$  and // e verlässt S?
(12)     $w(e) < Q.PRIORITY(pos[v])$  then { // günstiger?
(13)     Q.DECREASEPRIORITY(pos[v],  $w(e)$ )
(14)      $\pi[v] := u$ 
(15)  } } }

```

Analyse

- Aufbau des Heaps für PQ: in $\Theta(|V|)$
- $|V|$ Durchläufe der while-Schleife mit EXTRACTMIN $\Rightarrow O(|V| \log |V|)$
- 2 Durchläufe der forall-Schleife für jede Kante
- max. $|E|$ Aufrufe von DECREASEPRIORITY $\Rightarrow O(|E| \log |V|)$

Gesamtlaufzeit $O(|E| \log |V|)$, da $|E| \geq |V| - 1$

Warum kann Greedy-Verfahren funktionieren?

- MST besteht aus Teilbäumen, die ebenfalls MSTs sind
- Lokal beste Entscheidung (leichteste Kante) führt zu MST

nächste VO: kürzeste Wege Algorithmen