1. Greedy
2. Dynamic Programming
3. Flow

GREEDY

1) Minimum Spanning Tree

Instanz: Graph G, $w: E \to \mathbb{R}$

Ges.: G-aufspannender Baum T mit $\sum_{e \in T} w(e)$ minimal.

> 2.3 Satz
>
> Sei G ungerichtet mit n Knoten. Folgende Ausagen sich äquivalent
>
> - (a) G ist ein Baum (d.h. kreisfrei und zh)
> - (b) G ist kreisfrei und hat n-1 Kanten
> - (c) G ist zh und hat n-1 Kanten
> - (d) G ist minimal zusammenhängend (G-e unzusammenhängend für alle Kanten e), d.h. jede Kante ist Brücke
> - (e) G ist minimal (bzgl. Kanten) mit $\delta(X) \neq \emptyset$ für alle $\emptyset \neq X \subset V$
> - (f) G ist maximal kreisfrei (G+e enthält Kreis für alle Kanten e)
> - (g) G enthält eindeutigen elementaren Weg zwischen je zwei Knoten

$\Rightarrow$ "aus jedem Schnitt eine Kante wählen" $\Rightarrow$ die billigste

"aus jedem Kreis eine Kante nicht wählen" $\Rightarrow$ die teuerste

Algo färbt Kante: grün $\leftrightarrow$ Kante ist in MST
rot $\leftrightarrow$ Kante ist nicht in MST

- while es gibt eine ungefärbte Kante in G do
  - wende eine der folgenden Färbungsregeln an
    - Grüne Regel
      - anwendbar wenn ein Schnitt $\bar{\delta}(X)$ von G existiert, für den gilt
        - $\bar{\delta}(X)$ enthält keine grüne Kante
        - $\bar{\delta}(X)$ enthält mindestens eine ungefärbte Kante
      - Aktion
        - wähle billigste ungefärbte Kante des Schnittes und färbe sie grün
    - Rote Regel
      - anwendbar wenn ein Kreis C von G existert, für den gilt
        - C enthält keine rote Kante
        - C enthält mindestens eine ungefärbte Kante
      - Aktion
        - wähle teuerste ungefärbte Kante des Kreises und färbe sie rot

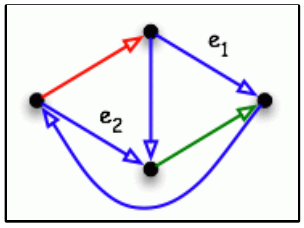
Z.: a) Algo ist wohldefiniert, d.h. eine der Regeln kann immer angewendet werden.

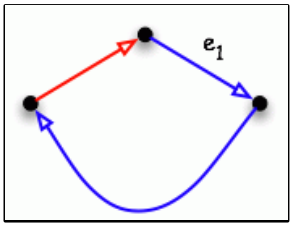
b) Grüne Kanten bilden am Ende einen MST

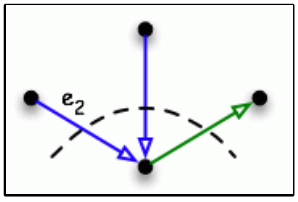
zu b): Invariante: $\exists$ ein MST, der alle grüne Kanten aber keine rote Kante enthält.

zu a):

- 2.5 Lemma (Färbungslemma von Minty)
  - Sei G ein Digraph, dessen Kanten rot, grün und blau gefärbt sind. Sei $e \in E(G)$ eine blaue Kante. Dann gilt genau eine der folgenden Ausagen:
    - (a) es gibt ungerichteten Kreis mit nur roten und blauen Kanten, der e enthält, so dass alle blauen Kanten im Kreis dieselbe Orientierung haben ("gleichgerichtet")
    - (b) es gibt ungerichteten Schnitt mit nur grünen und blauen Kanten, der e enthält, so dass alle blauen Kanten im Schnitt gleichgerichtet sind







Ansprüche der Algo:

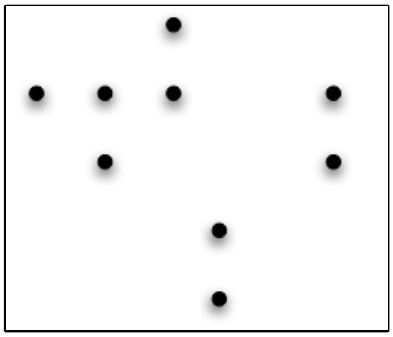
Kruskal:
- Sortiere Kanten nach Gewicht
- Füge Kante in dieser Reihenfolge ein, falls sie keinen Kreis schließt.

"nur wenig Regel"

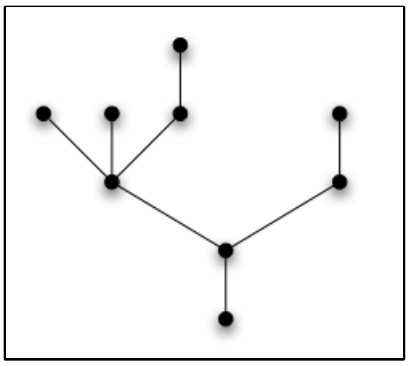
Prim:
- Starte mit bel. Knoten, füge ihn in R ein
- Finde den Knoten mit "geringstem Abstand" zu R und füge ihn in R ein.

"nur grüne Regel"

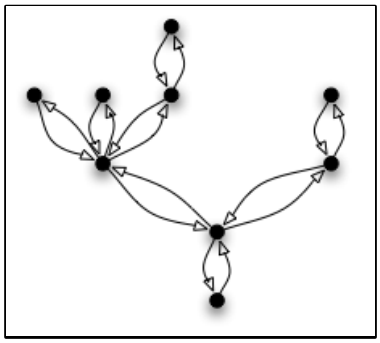
Anwendung von MST: z.B. 2-Approximation für metrisches TSP.

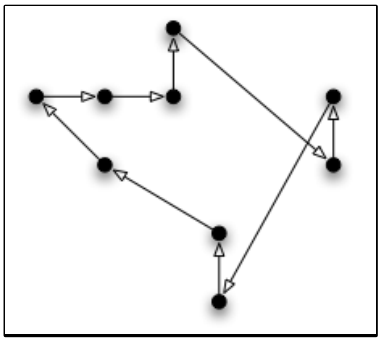


MST T



Eulerg!





"Euler-Tour mit Abkürzungen"

$ALG(I) \leq 2 \cdot \text{Länge}(T) \leq 2 \cdot OPT(I)$

Δ-Ungl. $\quad$ $\text{Länge}(MST) \leq \text{Länge}(Tour)$

2) Dijkstra-Algorithmus

- kürzester Weg in Graphen
- im Allg. NP-vollständig:
  Reduktion von Hamilton Path:
  ↓
  (∃ elem. s-t-Weg, der über alle Knoten in G geht?)

Instanz: Digraph G, $s,t \in V$, $c: E \to \mathbb{R}$

gesucht: elem. s-t-Weg P mit $\sum_{e \in P} c(e)$ minimal

$\Rightarrow \quad c(e) \equiv -1$

$\Rightarrow$ $\exists$ s-t-Hamilton-Path in G $\iff$ kürzester s-t-Weg in G ist $(n-1)$.

$\Rightarrow$ Forderung für kürzeste-Wege-Probl.: c **konservativ**, d.h. $\nexists$ Kreis C mit $\sum_{e \in C} c(e) < 0$.

Vorr. für Dijkstra noch stärker: $c(e) \geq 0 \quad \forall e \in E$.

Alg':

```
d(v) = ∞   ∀ v ∈ V
d(s) = 0
Q := {s}
while (Q ≠ ∅) {
    v ∈ Q: d(v) minimal        // Greedy!
    Q := Q ∪ {v}
    ∀ (v,w) ∈ E {
        if (d(w) > d(v) + c((v,w)) {
            d(w) = d(v) + c((v,w))
```

$Q := Q \cup \{v\}$

}

}

Beweis der Korrektheit: ① Beim Entfernen von v aus Q ist $d[v]$ die kürzeste s-v-Weglänge.

Nicht schief, falls $\exists e: c(e) < 0$

② Am Ende der Iteration, in der v aus Q entfernt wurde sind $d(w)$ $\forall w \in Q$ die kürzeste Länge von s-v-Wege, die nur über Knoten gehen, die schon aus Q entfernt wurden.

→ dann Induktion.

Verallgemeinerung von Dijkstra: • kürzeste Wege in Graph mit Kosten für Linksabbiegen. (Aufgabe 16)

⇒ bis zu 4 Label je Knoten: k. Wege von N, S, O, W.

⇒ allgemeine Idee wird oft verwendet!

Spezialisierungen: • zielgerichtete Dijkstra für euklidische Graphen (Aufgabe 16)

• bidirektionale Dijkstra (Aufgabe 17)

---

3) Approximationsalgorithmen

• oft Greedy — dann Approximationsgüte zeigen

z.B. • log-n-Approx. für Weighted Vertex Cover (greedy nach $\frac{w(v)}{|\text{neu überdeckte Kanten}|}$)

- 2-Approximation für (unweighted) Vertex Cover
  - greedy inklusionsmaximales Matching
  - alle Endknoten von Matchingkanten auswählen.

DYNAMIC PROGRAMMING

- "Prinzip der optimalen Substruktur"

1) Bellman-Ford-Alg für kürzeste Wege bei konservativen Kantengewichten

DP-Gleichung: $c_{ij}^{(m+1)} = \min_{k \in V} \{ c_{ik}^{(m)} + c_{kj} \}$

- $c_{ij}^{(m+1)}$: [Länge eines k.W. von i nach j mit $\leq m+1$ Kanten]
- $c_{ik}^{(m)}$: optimale Substruktur
- c_{kj}: "Teilwege von kürzesten Wegen sind kürzeste Teilwege"

Anwendung:
- Successive Shortest Path Alg für Min Cost Flow
- Min Cost Perfect Matching in bipartiten Graphen (VE)

2) Minimum Mean Cycle Algorithm

Instanz: Digraph G, $c: E \to \mathbb{R}$, ($s \in V(G)$ mit alle Knoten sind von s aus erreichbar ↑ o.B.d.A.)

Ges: ger. Kreis C mit $\frac{\sum_{e \in C} c(e)}{|C|}$ minimal.

(Bem.: Min Cycle ist NP-vollständig (Reduktion von Hamilton-Kreis))

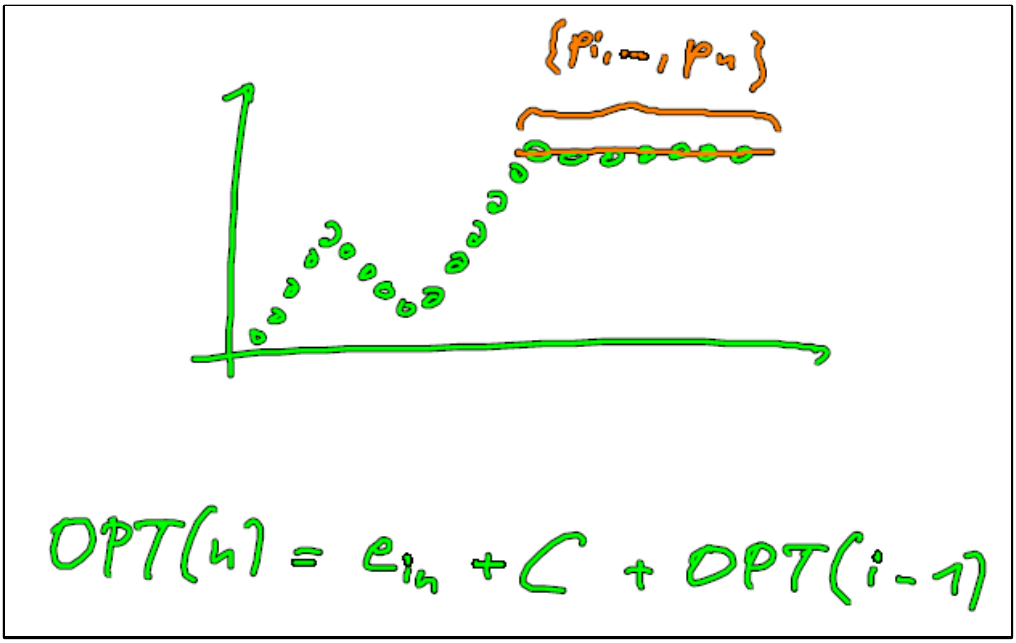
Satz (Karp '78): Sei $F_k(v)$ die minimale Länge eines s-v-Weges mit genau k Kanten. Dann ist die Länge eines Min Mean Cycles in G

$$\mu(G) := \min_{v \in V} \max_{0 \le k \le n-1} \frac{F_n(v) - F_k(v)}{n-k}$$

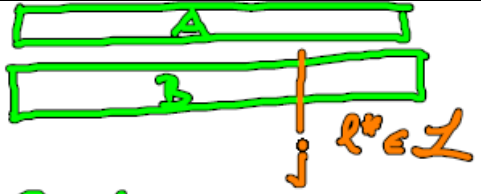
- Algorithmus ähnlich zu Bellman-Ford.
- $O(n(n+m))$

Anwendung: • Min Mean Cycle Canceling für Min Cost Flow

weiter DP-Algo: • "Segmented Least Squares" (UE)



- "Babelfish"-Aufgabe (Aufgabe 20)



j ist die Position des ersten Zeichens von l^* in B^*

$$OPT(n) = OPT(j-1) + c^*(l^*, j)$$

$\Rightarrow$ DP machbar auch "von links"

Flow

1) Max Flow: Instanz: Digraph G, $u: E \to \mathbb{R}^+$, $s,t \in V(G)$

Ges.: s-t-Fluss F mit $\nu(F)$ maximal.

- Fluss: $f: E \to \mathbb{R}^+$ mit $f(e) \leq u(e)$ $\forall e \in E$
- s-t-Fluss: Fluss F mit $\sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e)$ $\forall v \in V \setminus \{s,t\}$
- Wert eines s-t-Flusses: $\nu(f) := \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) = \sum_{e \in \delta^-(t)} f(e) - \sum_{e \in \delta^+(t)} f(e)$

- Residualgraph G_f zu F:

$V(G_f) = V(G)$ (VK / RK)

$E(G_f) = \{(v,w), (w,v) : (v,w) \in E(G)\}$

$$u(e) := \begin{cases} u(e) - f(e) & e \in VK \\ f(e) & e \in RK \end{cases}$$

- ein f-augmentierender Weg in G_f ist s-t-Weg P in G_f mit $\min_{e \in P} \{c_f(e)\} > 0$

Satz (Optimalitätskrit. für max Fluss):

f maximal $\Leftrightarrow$ $\nexists$ f-augmentierender Weg in G_f

als Folgerung: In einem Flussnetzwerk (G, c, s, t) ist die minimale Kapazität eines s-t-Schnitts gleich dem maximalen Wert eines s-t-Flusses.

$\Rightarrow$ Alg für max Fluss (Ford-Fulkerson)

- $f \equiv 0$
- while ($\exists$ f-aug. Weg P in G_f)
  augmentiere f entlang P.

$\Rightarrow$ $O(v_f \cdot m)$ bei ganzz. Kapazitäten

Verbesserung: Augmentiere entlang kürzestmöglichem Weg. (Edmonds-Karp-Algo)

FF, EK heißen **primale Algos**

- verwalten eine zul. Lösung
- stelle Schritt für Schritt Optimalität her

Dagegen: **duale Algos**

- erhalte Optimalitätskriterium

. stelle Schritt für Schritt Zulässigkeit her

z.B. Path-Relabel-Algo.

. mache Präfluss zu echtem Fluss

. dabei gibt es nur einen s-t-Weg in G_f

Spezialisierung: Goldberg-Tarjan

2) Min Cost Flow

. primal: Min Mean Cycle Canceling Algo

. dual: Successive Shortest Path Algo