

Informatik II

Übung 12

FS 2019

Heutiges Programm

- 1 Feedback letzte Übung
- 2 Wiederholung Vorlesung
- 3 In-Class-Exercise (praktisch)

1. Feedback letzte Übung

2. Wiederholung Vorlesung

Union-Find Algorithmus MST-Kruskal(G)

Input: Gewichteter Graph $G = (V, E, c)$

Output: Minimaler Spannbaum mit Kanten A .

Sortiere Kanten nach Gewicht $c(e_1) \leq \dots \leq c(e_m)$

$A \leftarrow \emptyset$

for $k = 1$ **to** $|V|$ **do**

 MakeSet(k)

for $k = 1$ **to** m **do**

$(u, v) \leftarrow e_k$

if Find(u) \neq Find(v) **then**

 Union(Find(u), Find(v))

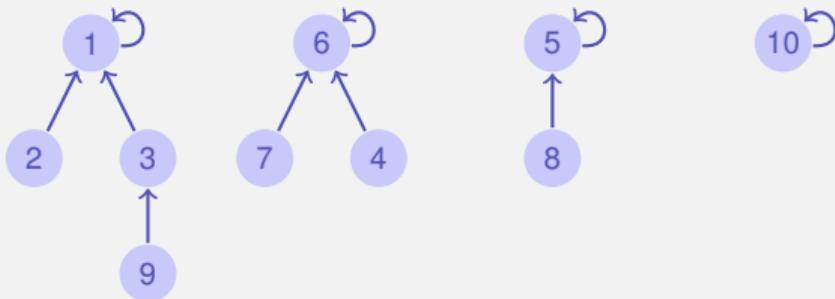
$A \leftarrow A \cup e_k$

else

// konzeptuell: $R \leftarrow R \cup e_k$

return (V, A, c)

Implementation Union-Find



Repräsentation als Array:

Index	1	2	3	4	5	6	7	8	9	10
Parent	1	1	1	6	5	6	5	5	3	10

Implementation Union-Find

Index	1	2	3	4	5	6	7	8	9	10
Parent	1	1	1	6	5	6	5	5	3	10

Make-Set(i) $p[i] \leftarrow i$; **return** i

Find(i) **while** ($p[i] \neq i$) **do** $i \leftarrow p[i]$
 return i

Union(i, j)¹ $p[j] \leftarrow i$;

¹ i und j müssen Namen (Wurzeln) der Mengen sein. Andernfalls verwende `Union(Find(i), Find(j))`

Optimierung der Laufzeit für Find

Baum kann entarten: Beispiel Union(8, 7), Union(7, 6), Union(6, 5), ...

Index	1	2	3	4	5	6	7	8	..
Parent	1	1	2	3	4	5	6	7	..

Laufzeit von Find im schlechtesten Fall in $\Theta(n)$.

Optimierung der Laufzeit für Find

Idee: Immer kleineren Baum unter grösseren Baum hängen.
Benötigt zusätzliche Grösseninformation (Array) g

Make-Set(i) $p[i] \leftarrow i; g[i] \leftarrow 1; \text{return } i$

Union(i, j) **if** $g[j] > g[i]$ **then** $\text{swap}(i, j)$
 $p[j] \leftarrow i$
 if $g[i] = g[j]$ **then** $g[i] \leftarrow g[i] + 1$

⇒ Baumtiefe (und schlechteste Laufzeit für Find) in $\Theta(\log n)$

Weitere Verbesserung

Bei jedem Find alle Knoten direkt an den Wurzelknoten hängen.

Find(i):

$j \leftarrow i$

while ($p[i] \neq i$) **do** $i \leftarrow p[i]$

while ($j \neq i$) **do**

$t \leftarrow j$
 $j \leftarrow p[j]$
 $p[t] \leftarrow i$

return i

Laufzeit: amortisiert *fast* konstant (Inverse der Ackermannfunktion).²

²Wird hier nicht vertieft.

Laufzeit des Kruskal Algorithmus

- Sortieren der Kanten: $\Theta(|E| \log |E|) = \Theta(|E| \log |V|)$.³
 - Initialisieren der Union-Find Datenstruktur $\Theta(|V|)$
 - $|E| \times \text{Union}(\text{Find}(x), \text{Find}(y))$: $\mathcal{O}(|E| \log |E|) = \mathcal{O}(|E| \log |V|)$.
- Insgesamt $\Theta(|E| \log |V|)$.

³da G zusammenhängend: $|V| \leq |E| \leq |V|^2$

3. In-Class-Exercise (praktisch)

Union-Find Datenstruktur Datenstruktur und ihre Optimierungen

Fragen oder Anregungen?