

Informatik II

Übung 10

FS 2020

Program Today

- 1 Feedback of last exercise
- 2 Repetition of Lecture
- 3 In-Class-Exercise (practical)

1. Feedback of last exercise

2. Repetition of Lecture

Union-Find Algorithm MST-Kruskal(G)

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

Output: Minimum spanning tree with edges A .

Sort edges by weight $c(e_1) \leq \dots \leq c(e_m)$

$A \leftarrow \emptyset$

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

\lfloor 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))

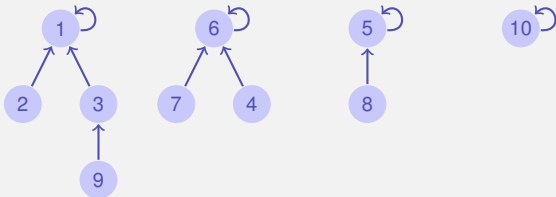
$A \leftarrow A \cup e_k$

else

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

return (V, A, c)

Implementation Union-Find



Representation as 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 and j need to be names (roots) of the sets. Otherwise use `Union(Find(i),Find(j))`

Optimisation of the runtime for Find

Tree may degenerate. Example: 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	..

Worst-case running time of Find in $\Theta(n)$.

Optimisation of the runtime for Find

Idea: always append smaller tree to larger tree. Requires additional size information (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** $g[i] \leftarrow g[i] + 1$

\Rightarrow Tree depth (and worst-case running time for Find) in $\Theta(\log n)$

Further improvement

Link all nodes to the root when Find is called.

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

Cost: amortised *nearly* constant (inverse of the Ackermann-function).²

²We do not go into details here.

Running time of Kruskal's Algorithm

- Sorting of the edges: $\Theta(|E| \log |E|) = \Theta(|E| \log |V|)$.³
- Initialisation of the Union-Find data structure $\Theta(|V|)$
- $|E| \times \text{Union}(\text{Find}(x), \text{Find}(y))$: $\mathcal{O}(|E| \log |E|) = \mathcal{O}(|E| \log |V|)$.

Overall $\Theta(|E| \log |V|)$.

³because G is connected: $|V| \leq |E| \leq |V|^2$

3. In-Class-Exercise (practical)

Union-Find datastructure and its optimisations

Questions / Suggestions?