Datenstrukturen und Algorithmen

Exercise 11

FS 2021

Program Today

1. Feedback of last exercise
2. Repetition of Lecture

1. Feedback of last exercise

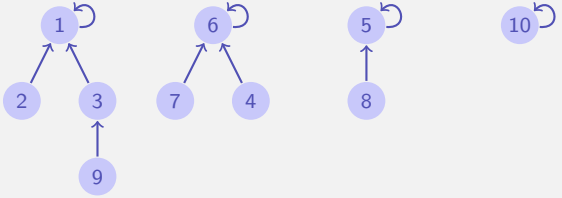
2. Repetition of Lecture

Union-Find Algorithm MST-Kruskal(G)

Input: Weighted Graph $G = (V, E, c)$
Output: Minimum spanning tree with edges A.

Sort edges by weight $c(e_1) \leq ... \leq c(e_m)$
$A \leftarrow \emptyset$
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
        // 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	$\mathbf{1}$	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)	$p[i] \leftarrow i$; $g[i] \leftarrow 1$; **return** i
Union(i, j)	**if** $g[j] > g[i]$ **then** 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]

[2] We 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 $\Theta(|V|)$
- $|E|\times$ Union(Find(x),Find(y)): $\mathcal{O}(|E| \log |E|) = \mathcal{O}(|E| \log |V|)$.

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

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

Travelling Salesperson Problem

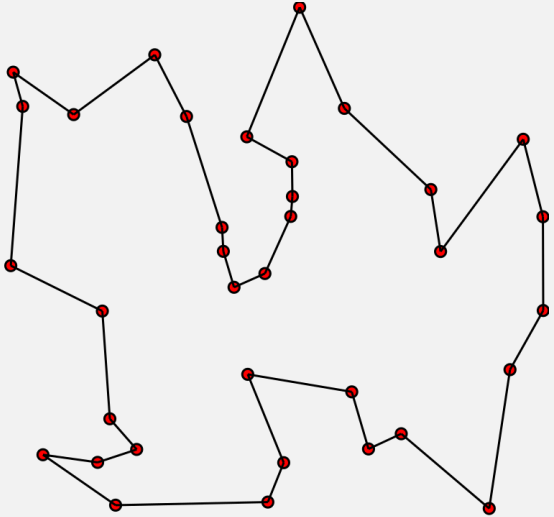
Problem

Given a map and list of cities, what is the shortest possible route that visits each city once and returns at the original city?

Mathematical model

On an undirected, weighted graph G, which cycle containing all of G's vertices has the lowest weight sum?

Travelling Salesperson Problem



Travelling Salesperson Problem

- The problem has no known polynomial-time solution.
- Many heuristic algorithms exists. They do not always return the optimal solution.

Travelling Salesperson Problem

- The heuristic algorithm that you are asked to implement on CodeExpert (*The Travelling Student*) on CodeExpert uses an MST:
  1. Compute the minimum spanning tree M
  2. Make a depth first search on M
- The algorithm is 2-approximate, meaning that the solution it generates has at most twice the cost of the optimal solution.
- The algorithm assumes a complete graph $G = (V, E, c)$that satisfies the triangle inequality: $c(v, w) \leq c(v, x) + c(x, w) \ \forall \ v, w, x \in V$