Datenstrukturen und Algorithmen

Übung 11

FS 2021

Heutiges Programm

1. Feedback letzte Übung
2. Wiederholung Vorlesung

1. Feedback letzte Übung

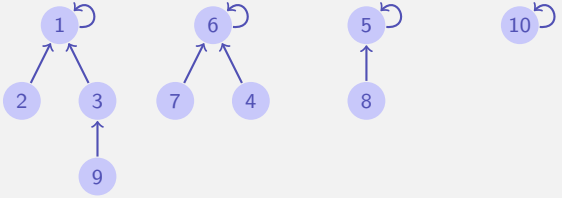
2. Wiederholung Vorlesung

Union-Find Algorithmus MST-Kruskal(G)

Input: Gewichteter Graph $G = (V, E, c)$
Output: Minimaler Spannbaum mit Kanten A.

Sortiere Kanten nach Gewicht $c(e_1) \leq ... \leq c(e_m)$
$A \leftarrow \emptyset$
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)	$p[i] \leftarrow i$; $g[i] \leftarrow 1$; **return** i
Union(i, j)	**if** $g[j] > g[i]$ **then** swap(i, j) $p[j] \leftarrow i$ **if** $g[i] = g[j]$ **then** $g[i] \leftarrow g[i] + 1$

$\Rightarrow$ 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]

[2] Wird hier nicht vertieft.

Laufzeit des Kruskal Algorithmus

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

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

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

Travelling Salesperson Problem

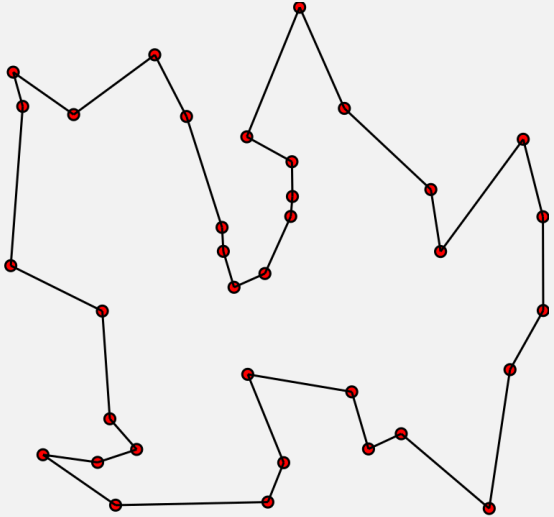
Problem

Gegeben ist eine Karte und eine Liste von Städten. Welches ist die kürzeste Route, die jede Stadt einmal besucht und in die ursprüngliche Stadt zurückkehrt?

Mathematical model

Auf einem ungerichteten, gewichteten Graph G ist nach dem Kreis gesucht, welcher jede der Knoten von G genau einmal enthält und die kleinste Kantensumme aufweist.

Travelling Salesperson Problem



Travelling Salesperson Problem

- Es ist kein Polynomialzeitalgorithmus zum Lösen des Problems bekannt.
- Es gibt verschiedene heuristische Algorithmen. Diese liefern oft nicht die optimale Lösung.

Travelling Salesperson Problem

- Der heuristische Algorithmus, den Sie auf CodeExpert implementieren sollen (*The Travelling Student*) benutzt einen Minimalen Spannbaum:
  1. Berechne den Minimalen Spannbaum M
  2. Mache eine Tiefensuche auf M
- Der Algorithmus ist eine 2-Approximation. Das bedeutet, dass er eine Lösung liefert, die maximal 2 mal die Kosten einer optimalen Lösung aufweist.
- Der Algorithmus geht von einem vollständigen Graphen $G = (V, E, c)$ aus, auf dem die Dreiechsungleichung gilt: $c(v, w) \leq c(v, x) + c(x, w) \; \forall \; v, w, x \in V$