Borůvkas MST-Algorithmus

Wolfgang Mulzer

1 Der Algorithmus

- **Eingabe**: ein zusammenhängender gewichteter Graph $G = (V, E)$.
- **Annahme**: alle Kantengewichte in G sind paarweise verschieden.

```
// Initialisiere Kantenmenge des MST als leere Menge
A <- {}
while |A| < n-1 do
  B <- {}
  for each connected component C of (V,A) do
    find the lightest edge e with exactly one endpoint in C
    add e to B (if e is not in B yet)
  add all edges in B to A
```

2 Analyse

Jede Durchlauf der `while`-Schleife lässt sich in $O(|E|)$ Zeit implementieren:

- Benutze BFS in (V, A), um die Zsh-Komponenten zu bestimmen, speichere mit jedem Knoten die Nummer seiner Zsh-Komponente. $O(|V| + |A|) = O(|E|)$ Zeit (da $|V|, |A| = O(|E|)$ sind).
- Speichere mit jeder Zsh-Komponente die leichteste inzidente Kante. Gehe alle Kanten durch, betrachte die beiden Zsh-Komponenten inzident zu der aktuellen Kante, aktualisiere die leichteste Kante, falls nötig. $O(|E|)$ Zeit.

Lemma 2.1. *Nach dem i-ten Durchlauf der `while`-Schleife haben alle Zsh-Komponenten mindestens 2^i Knoten.*

Beweis. Induktion: vor dem ersten Durchlauf hat jede Komponente $1 = 2^0$ Knoten. In jedem Durchlauf wird jede Zsh-Komponente mit mindestens einer anderen Zsh-Komponente vereinigt, also verdoppelt sich die Gröβe mindestens. □

Folglich gibt es nach $O(\log |V|)$ Durchläufen nur noch eine Zsh-Komponente. Die Laufzeit ist $O(|E| \log |V|)$.