Minimum Spanning Tree

- The Cycle Property (Red Rule)
  - Reverse Delete Algorithm for MST
- Kruskal's Algorithm for MST
- Runtime and Implementation
  - Disjoint Sets Data Structure

Imdad ullah Khan

Kruskal's Algorithm

Input: An undirected weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}$
Output: A spanning tree of G with minimum total weight

1. Makes a forest by making each vertex (an empty) tree
2. In every iteration merge two trees
3. Repeat until only one tree remains

Which trees to merge?

1. Process edges in increasing order
2. If (u, v) creates a cycle (u and v are in one tree), ignore it
3. If u and v are in two different trees, merge the corresponding trees

Kruskal's Algorithm

Algorithm Kruskal's Algorithm, $G = (V, E, w)$

Sort edges in increasing order of weights ▷ let $e_1, e_2, \ldots, e_m$ be the sorted order

$F \leftarrow \emptyset$ ▷ Begin with a forest with no edges

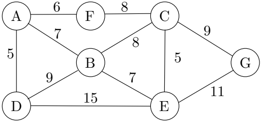
for $i = 1$ to m **do**

  if $F \cup e_i$ does not contain a cycle **then**

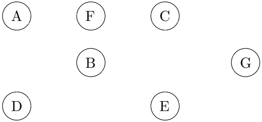
    $F \leftarrow F \cup \{e_i\}$

return F

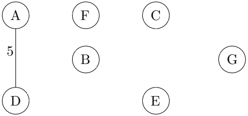
Kruskal's Algorithm: Example



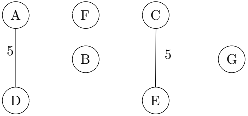
$G = (V, E, w)$



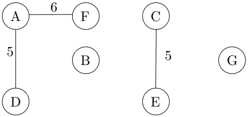
Initially each vertex is a tree



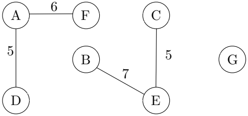
(A, D) is picked for merging



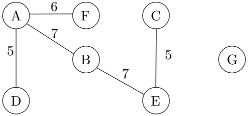
(C,E) is picked for merging



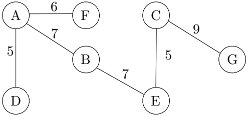
(A, F) is picked for merging



(B, E) is picked for merging



(A, B) is picked for merging



(C, G) is merged skipping $(F, C), (B, C), \dots$

Kruskal's Algorithm: Correctness

Algorithm Kruskal's Algorithm, $G = (V, E, w)$

Sort edges in increasing order of weights ▷ $e_1, e_2, \ldots, e_m$ is the sorted order
$F \leftarrow \emptyset$ ▷ Begin with a forest with no edges
for $i = 1$ to m **do**
  if $F \cup e_i$ does not contain a cycle **then**
    $F \leftarrow F \cup \{e_i\}$
return F

Correctness: F is a spanning tree of G

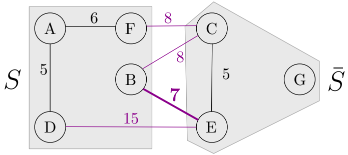
- F is a subgraph of G
- F is connected
- F has no cycle
- F is spanning

Optimality: F is the minimum spanning tree of G

Kruskal's Algorithm: Correctness

Correctness: F is a spanning tree of G

- F is a subgraph of G ▷ only used edges in G
- F has no cycles ▷ only add edges not making cycles
- F is connected and spans V
- Consider any cut $[S, \overline{S}]$ in G
- We will show that F crosses the cut $[S, \overline{S}]$
- So F has no empty cut
- Since $[S, \overline{S}]$ is not empty in G
- Edges $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ $(k \geq 1)$ cross $[S, \overline{S}]$
- We will pick e_{i_1}, as it cannot create cycle ▷ lonely cut lemma



Kruskal's Algorithm: Correctness

Optimality: F is the minimum spanning tree of G

Proof follows from the cut property

If an edge $e \in E$ is the lightest edge crossing some cut $[S, \overline{S}]$, then e belongs to the MST of G

- When edge $e = (u, v)$ was added, $F \cup \{(u, v)\}$ had no cycle
- Let S to be the tree containing u
- Let $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ be edges crossing $[S, \overline{S}]$ $\triangleright$ $k \geq 1$, $\because$ $[S, \overline{S}]$ is not empty
- e must be the lightest edge among $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$
- Otherwise some other edge must have been processed and S would be different
- The cut property guarantees inclusion of e in the MST