Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,
Prof. Ajit Diwan,
Prof. D.B. Phatak

Department of Computer Science and Engineering
IIT Bombay

Session: Spanning Tree Algorithm
(Kruskal's Algorithm)
Content largely adapted from CLRS, Third Edition

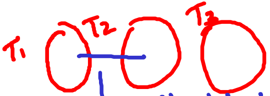
Kruskal's Algorithm: Introduction

Prims' algo:

Initialization: $T = \{\{v_1\}, \{v_2\}, \ldots \{v_n\}\}$ Greedy on vertices

$\{\{v_1, v_2\}, \ldots \{v_n\}\}$ → Continue until either
(a) Single tree
(b) No 2 trees to be connected using edge

1. Minimum-spanning-tree algorithm using greedy approach
2. Pick the smallest weight edge that does not cause a cycle in the minimum spanning tree
3. Finds an edge of the least possible weight that connects any two sub-trees in the forest
4. It finds a minimum spanning tree by adding increasing cost at each step



Should least possible wt

Kruskal's Algorithm

in order to facilitate merging of 2 trees in linear time

Set of edges in Tree i so far

SSets[i] = sequence (sorted in increasing id) of vertices in Tree i

Algorithm MST-Kruskal(G, w)
$\mathcal{T} = \phi$
$SSets[] =$ List of $G.numVertices()$ sequences
$SetID[] =$ List of $G.numVertices()$ integers
for $v \in G.getVertices()$ **do**
  $SSets[v].insert(v)$
  $SetID[v] = v$
end for
SE[] = Sorted edges of G.edges into non decreasing order by weight w
for $edge(u, v) \in SE[]$ **do**
  if $SetId[u] \neq SetId[v]$ **then**
    $\mathcal{T} = \mathcal{T} \cup (u, v)$
    $merge(SSets[u], SSets[v])$
    $SSets[v].empty()$
    $SetId[v] = u$
  end if
end for
return $\mathcal{T}$

SetId[i] = sequence SSet[j] that i is part of (to be able to find the tree containing i)

Assume $v \in \{0, 1 .. n-1\}$

To help greedily select smallest weighted edge

We have now merged SSets[v] into SSets[u]

Explicitly update the forest data structure!

$SSets[u] = SSets[u] \cup SSets[v]$

merge maintaining sorted order

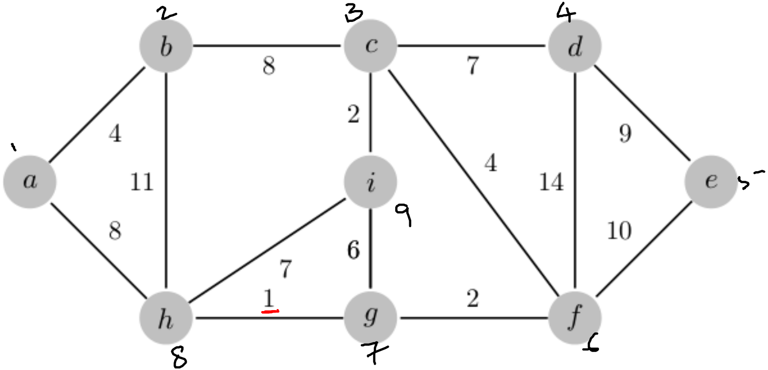
Figure: Kruskal's Algorithm

Loop invariant: $\mathcal{T}$ contains all edges of MST with wts < edges to be yet enumerated

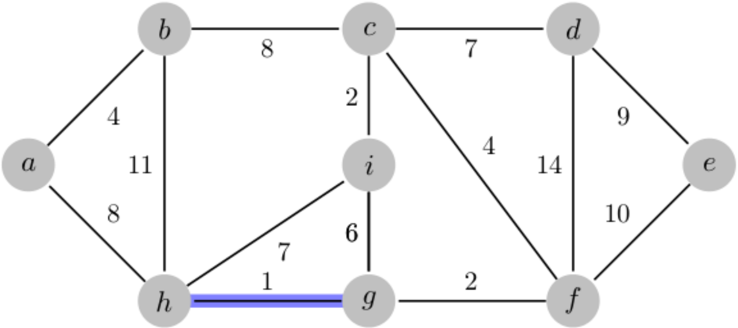
either connect 2 diff trees OR form cycle within a tree

Kruskal's Algorithm

$T = \{\{1\}, \{2\}\ . \ . \ \{9\}\}$

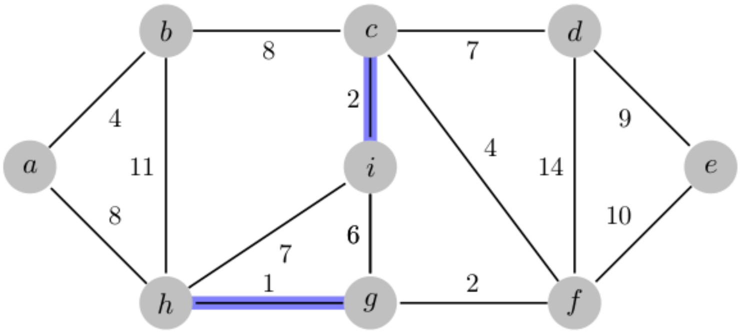


Kruskal's Algorithm

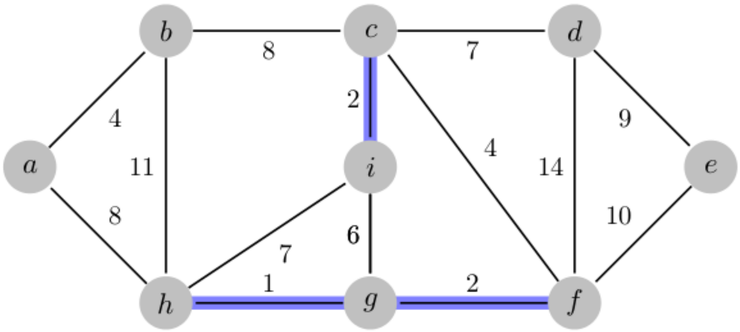


$\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g, h\}, \{i\}\}$

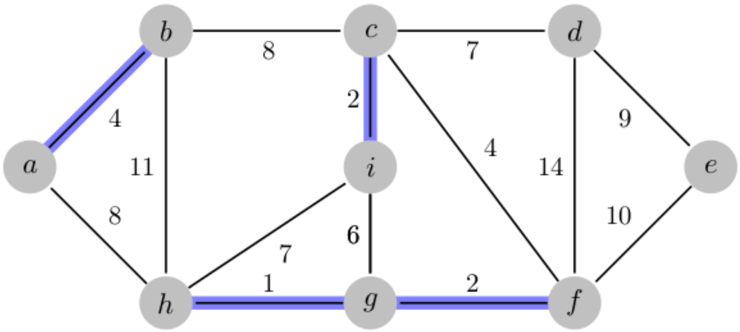
Kruskal's Algorithm



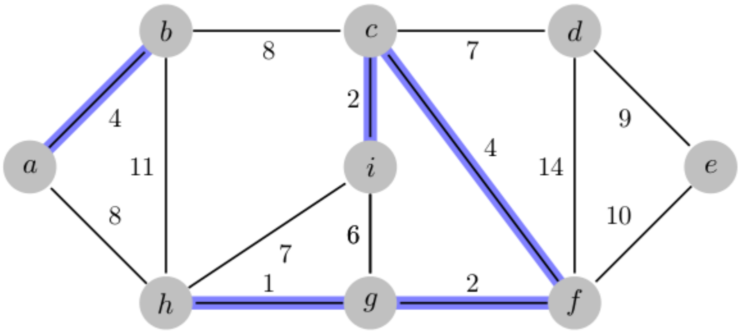
Kruskal's Algorithm



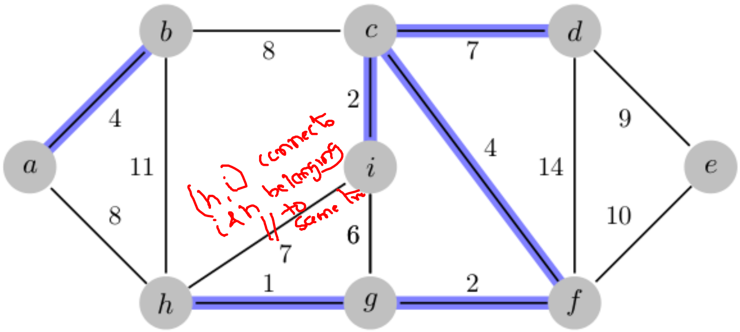
Kruskal's Algorithm



Kruskal's Algorithm



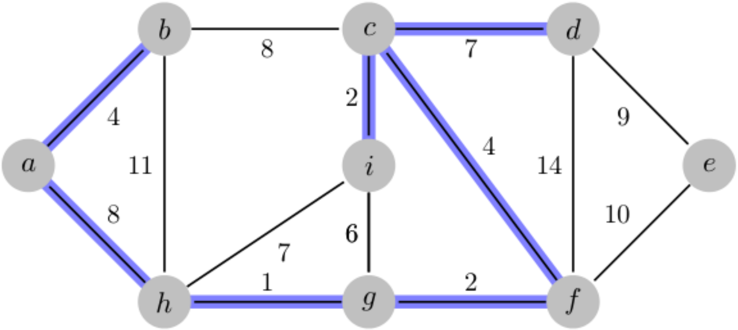
Kruskal's Algorithm



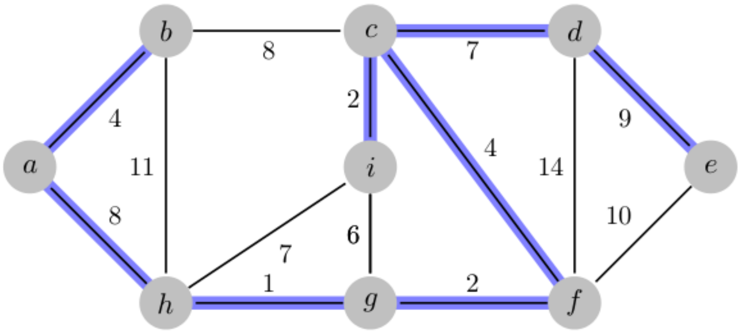
(h,i) connects i & h belonging to same tr

SetId[i] = SetId[h]

Kruskal's Algorithm



Kruskal's Algorithm



Analysis of Kruskal's Algorithm

Algorithm MST-Kruskal(G, w)
$\mathcal{T} = \phi$
$SSets[] =$ List of $G.numVertices()$ sequences
$SetID[] =$ List of $G.numVertices()$ integers
for $v \in G.getVertices()$ **do**
  $SSets[v].insert(v)$
  $SetID[v] = v$
end for $\Longrightarrow \quad c_1 \quad \times\ |V|$ times
SE[] = Sorted edges of G.edges into non decreasing order by weight w $\Longrightarrow \quad c_2 \quad \times\ O|E|log|E|$ times
for $edge(u, v) \in SE[]$ **do**
  if $SetId[u] \neq SetId[v]$ **then**
    $\mathcal{T} = \mathcal{T} \cup (u, v) \quad \Longrightarrow \quad c_3 \quad \times\ 2|E|$ times
    $merge(SSets[u], SSets[v]) \quad \Longrightarrow \quad c_4 \quad \times\ |E|$ times
    $SSets[v].empty() \quad \Longrightarrow \quad c_5 \quad \times\ |E|$ times
    $SetId[v] = u \quad \Longrightarrow \quad c_6 \quad \times\ |E|$ times
  end if
end for
return $\mathcal{T}$

Figure: Kruskal's Algorithm

$E = O(V^2)$ in worst case

$$T(n) = c_1|V| + c_2|E|log|E| + 2c_3(c_4 + c_5 + c_6)|E| = O(|E|log|E|) \text{ or } O(|E|log|V|)$$

Thank you