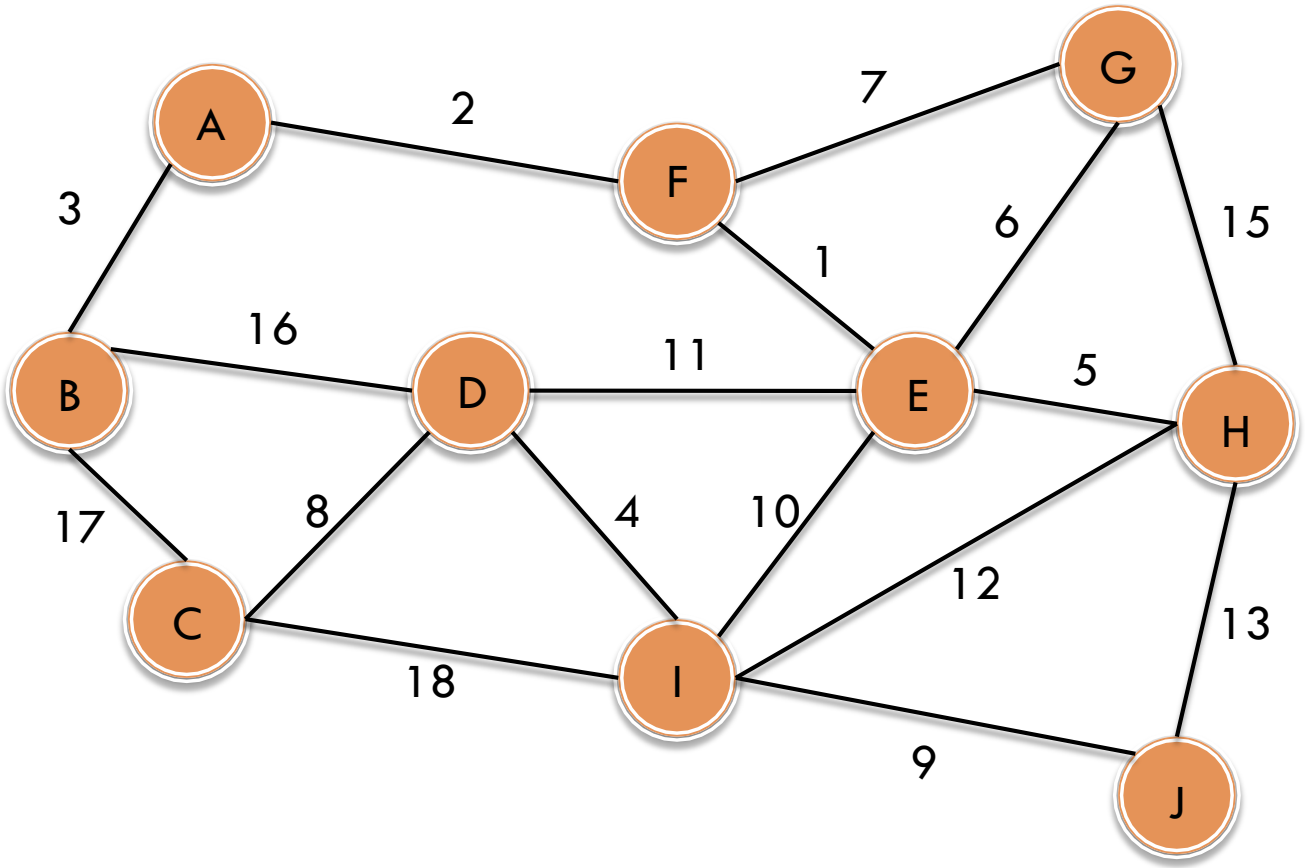
MINIMUM SPANNING TREE SEGMENTATION

David Esparza Alba
Ritsumeikan University

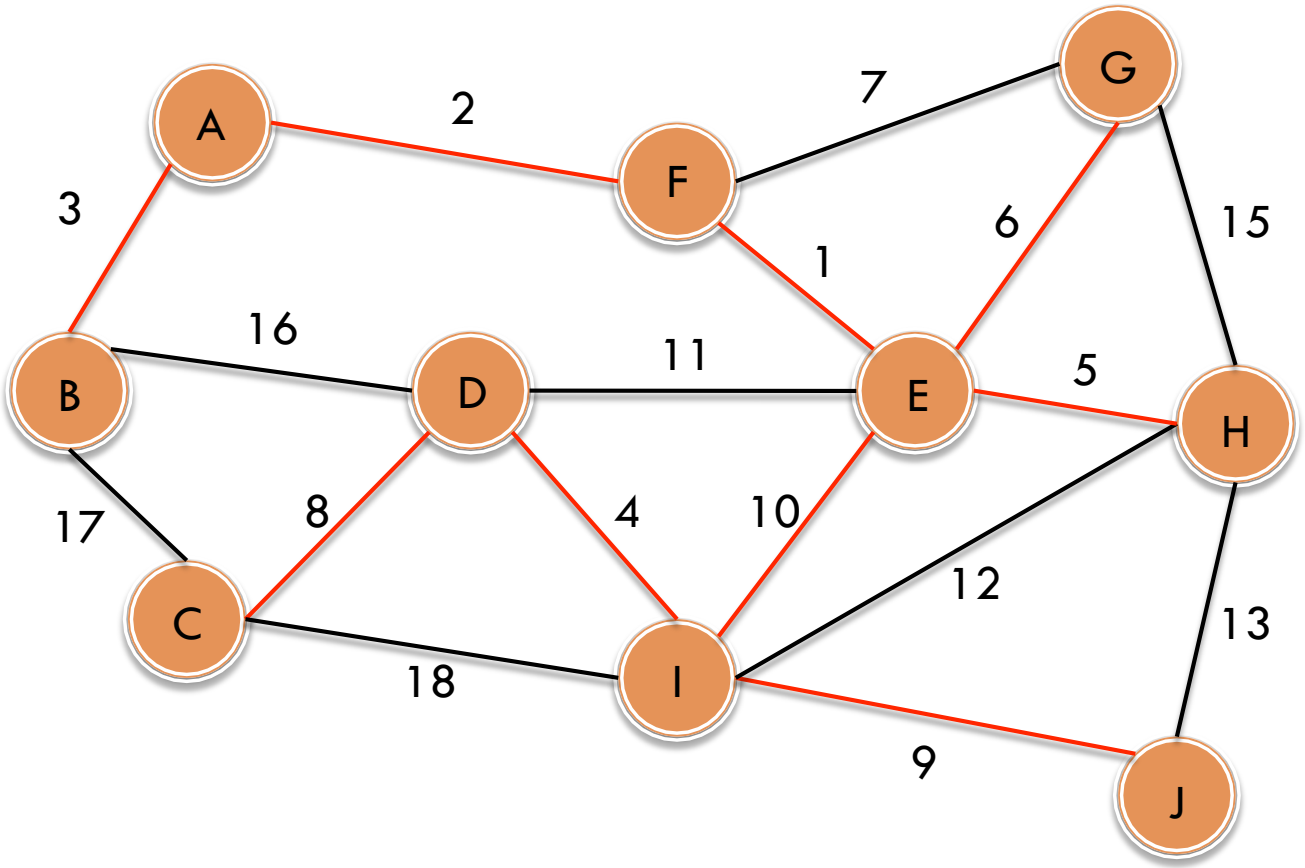
MST

- Tree: Is a undirected graph in which any two vertices are connected by a single path.
- Weighted Graph: A graph is a weighted graph if the edges have certain cost associated to them.
- Spanning Tree: Is a set of n-1 edges that form a tree that contains all n vertices in the graph. Acyclic graph in which any two pair of edges are connected.
- Minimum Spanning Tree: In a weighted graph, the MST is the spanning tree with minimum cost.

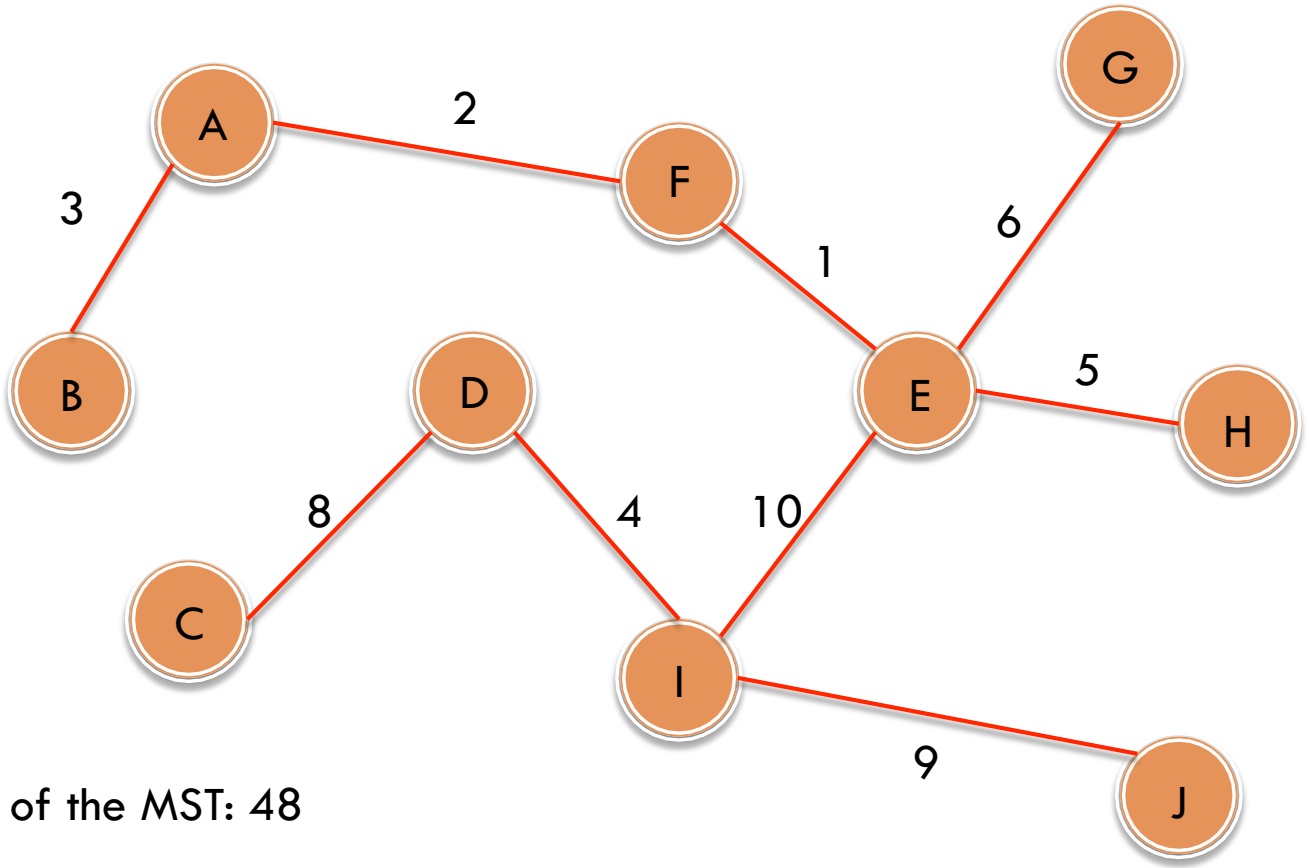
MST



MST



MST



Weight of the MST: 48

Kruskal Algorithm

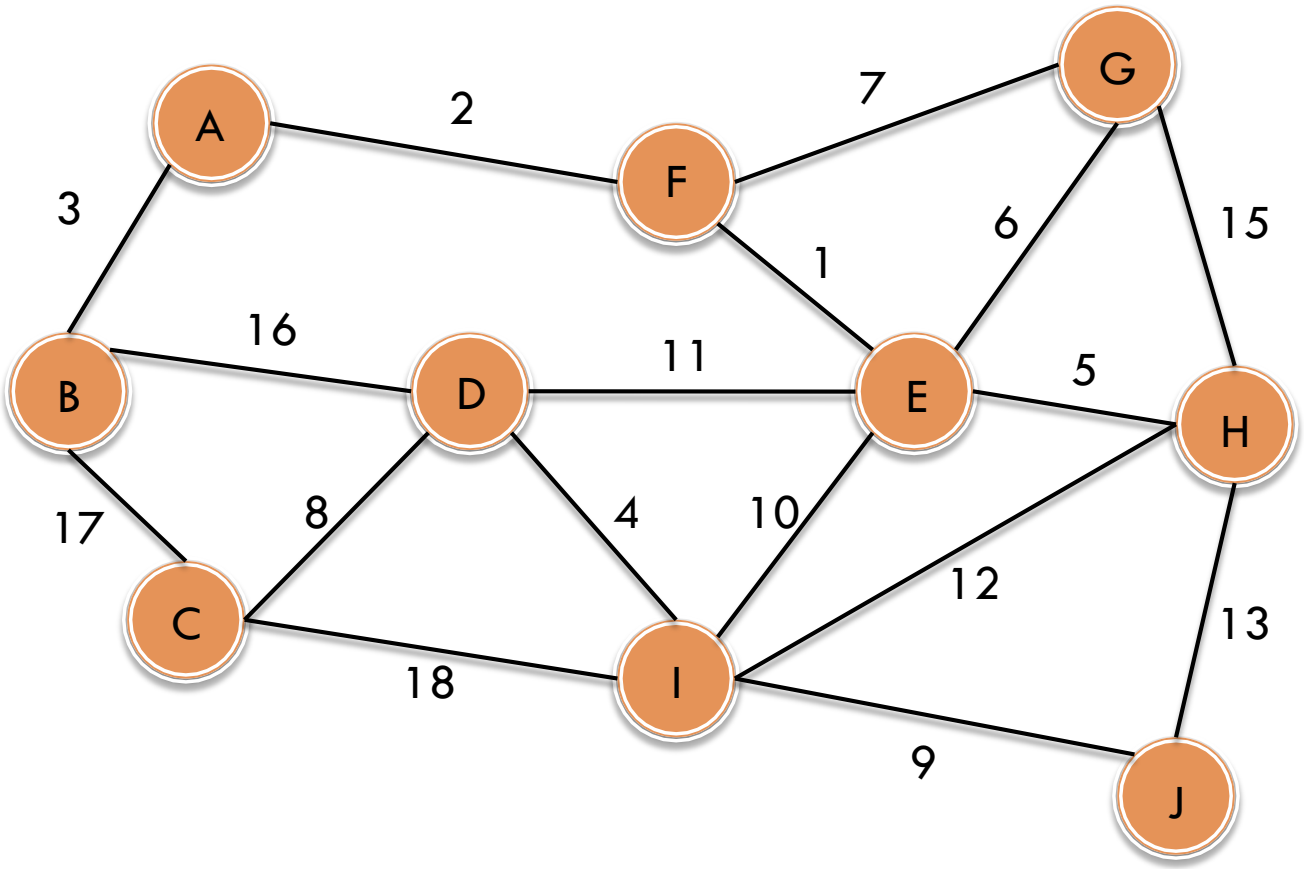
- Algorithm used to find the MST within a graph.
- Before to explain the algorithm let's make some definitions:
  - E: Set of all edges in the graph.
  - V: Set of all vertices in the graph.
  - MST: Set of edges that conforms the MST of the graph.
  - FindSet(v): Function that returns the set which vertex v belongs to.
  - UnionSet(u,v): Function that joins two different sets. In this case, the set of vertex u and the set of vertex v.

Kruskal Algorithm

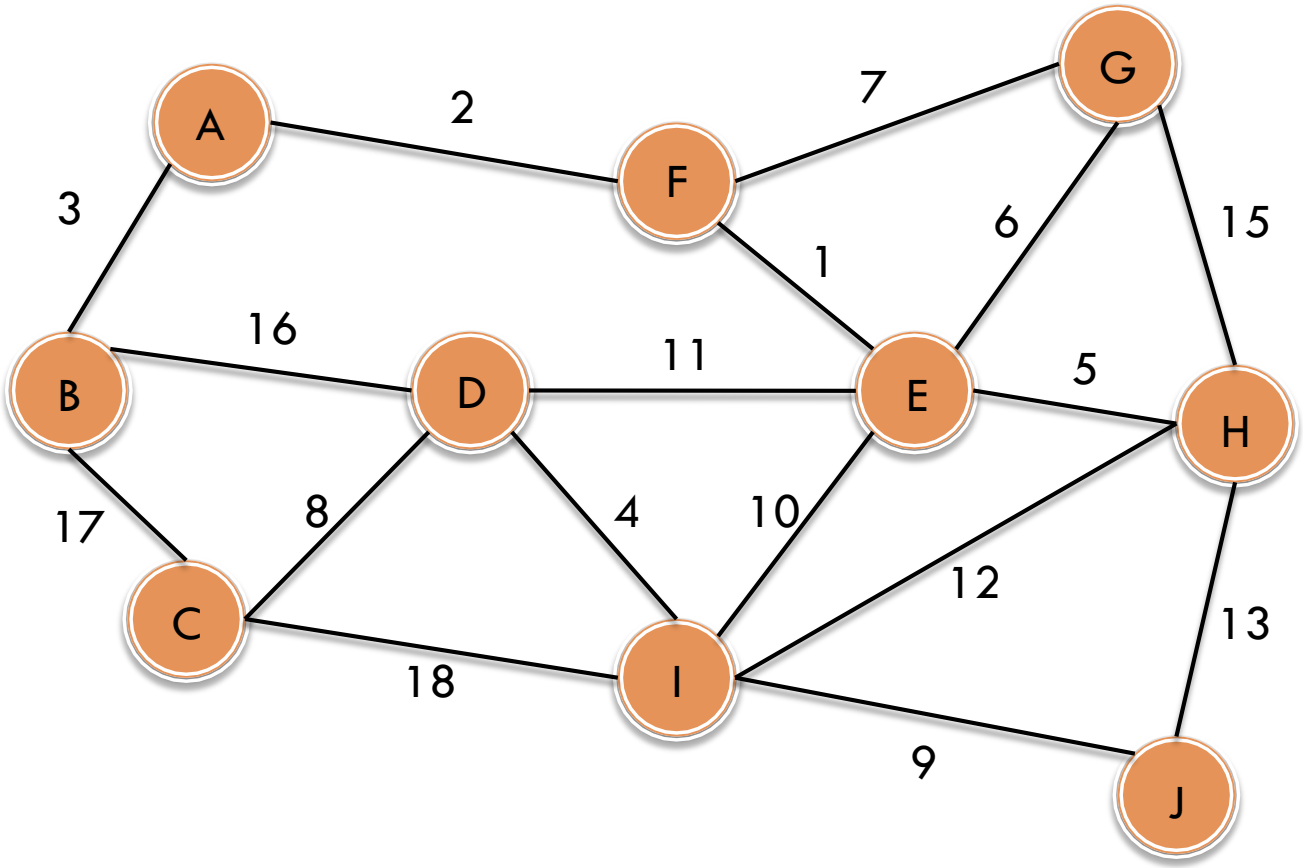
0) $MST \leftarrow$ empty
1) sort all the vertices $e_i \in E$ according to their weight
2) $i \leftarrow 0$
3) Be u_i and v_i the vertices joined by the edge e_i,
where $u_i, v_i \in V$ and $e_i \in E$
4) if($FindSet(u_i) \neq FindSet(v_i)$)
5) add e_i to MST
6) $UnionSet(u_i, v_i)$
7) $i \leftarrow i + 1$
8) if($i \leq |E|$)
9) return to 3
10) else
11) quit

Kruskal Algorithm

- Consider the following graph:



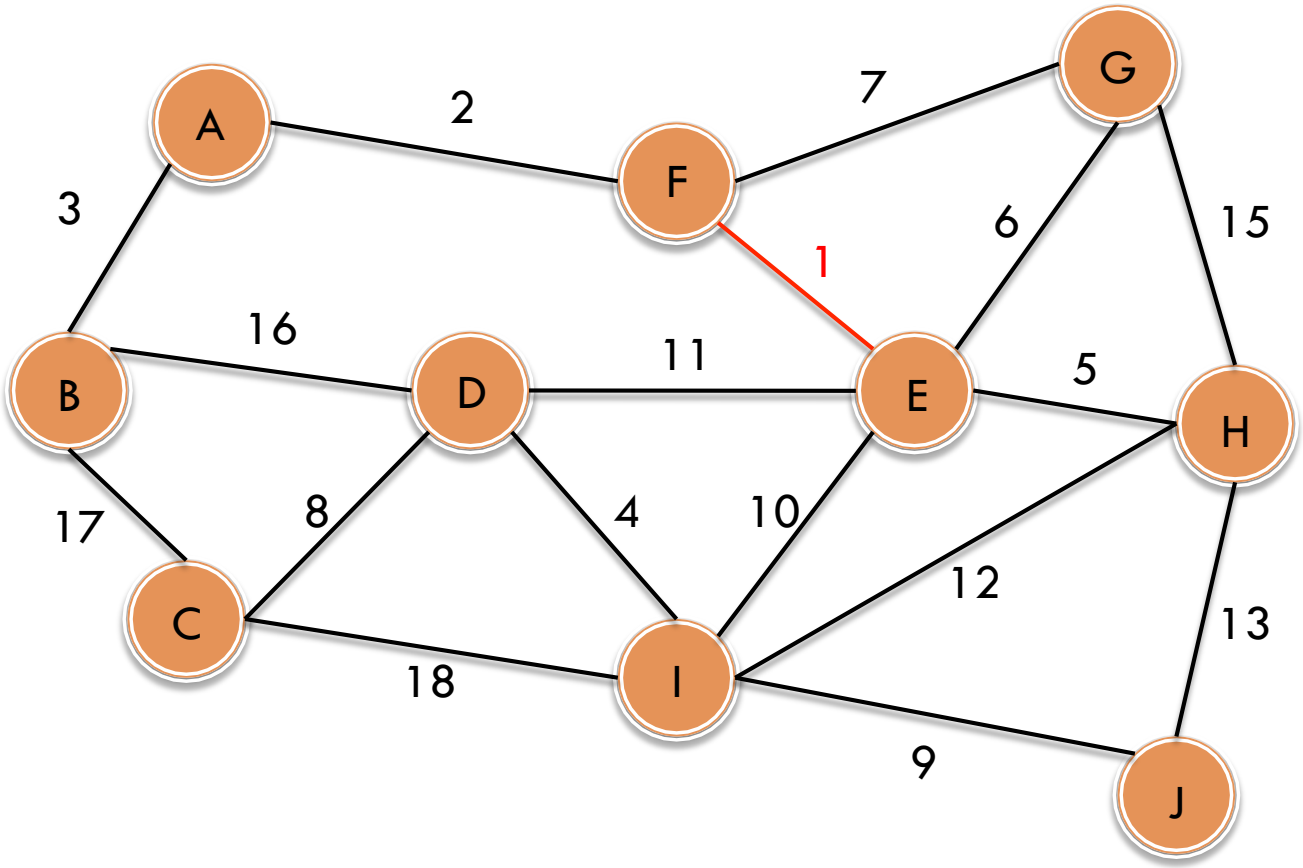
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

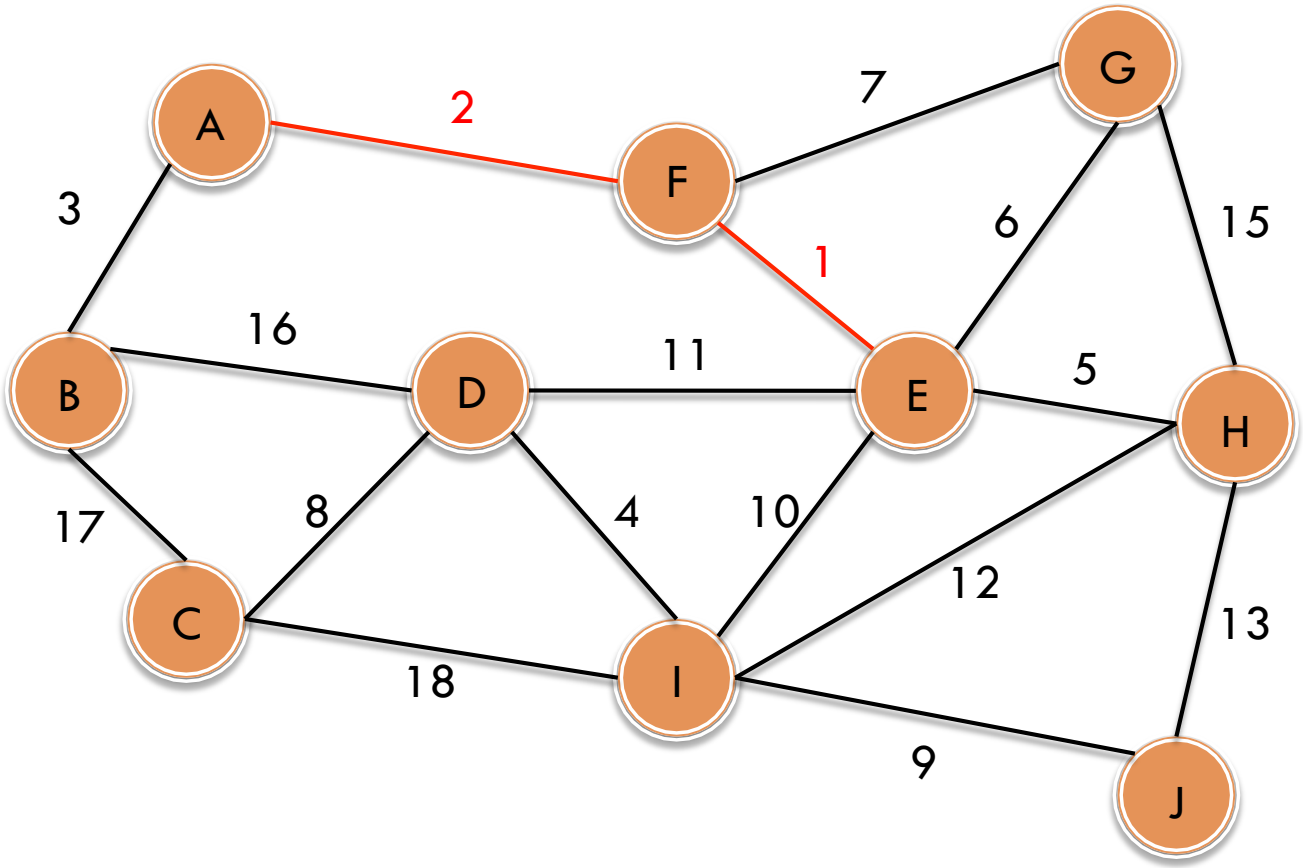
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

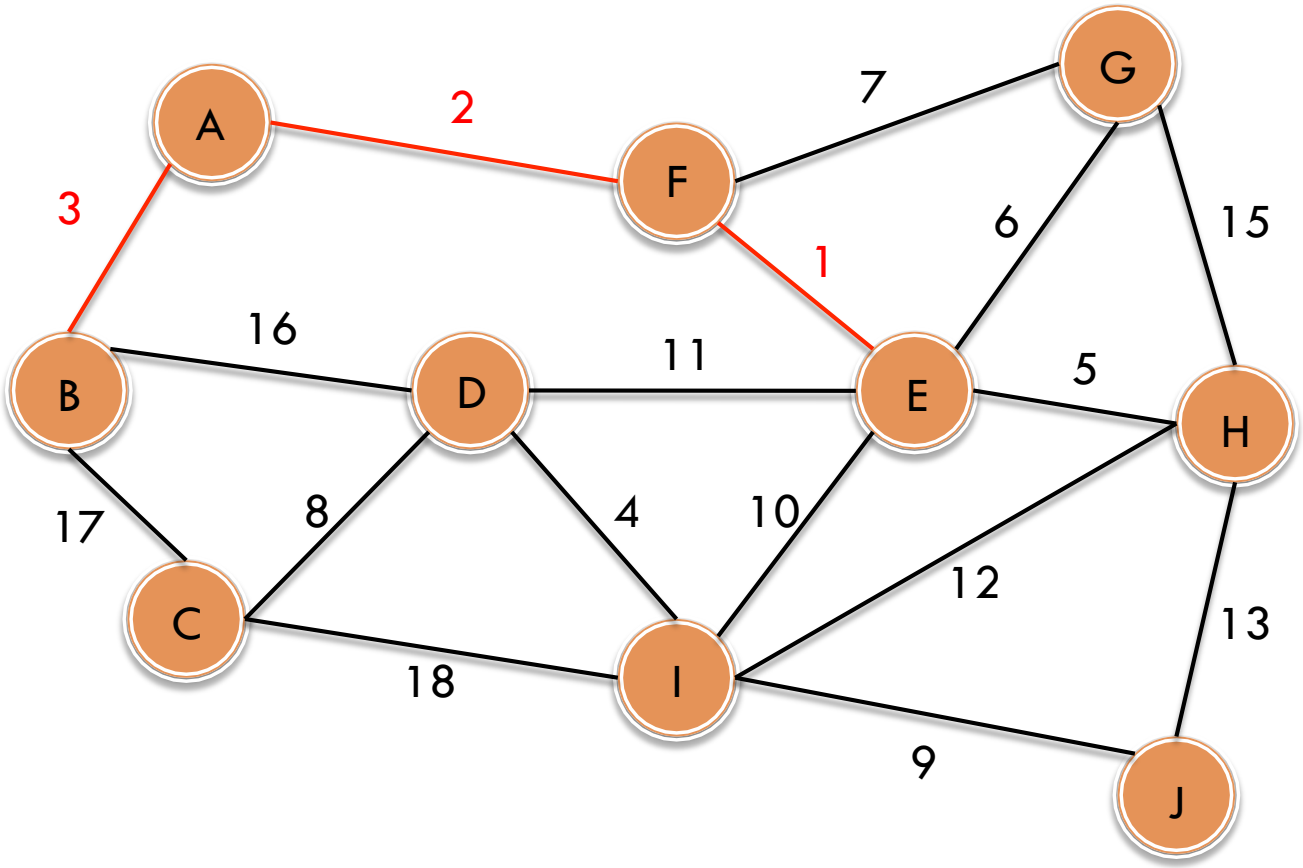
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

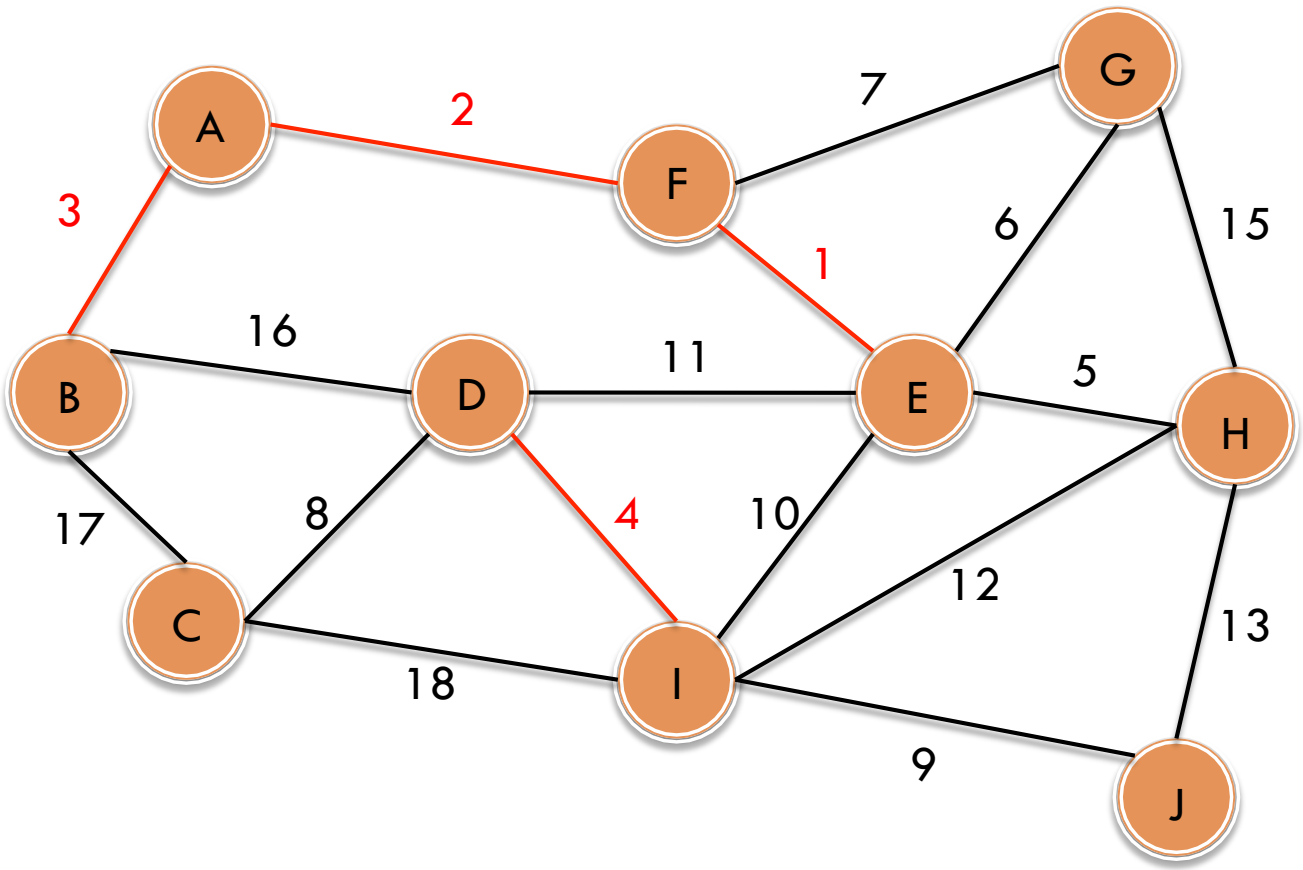
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

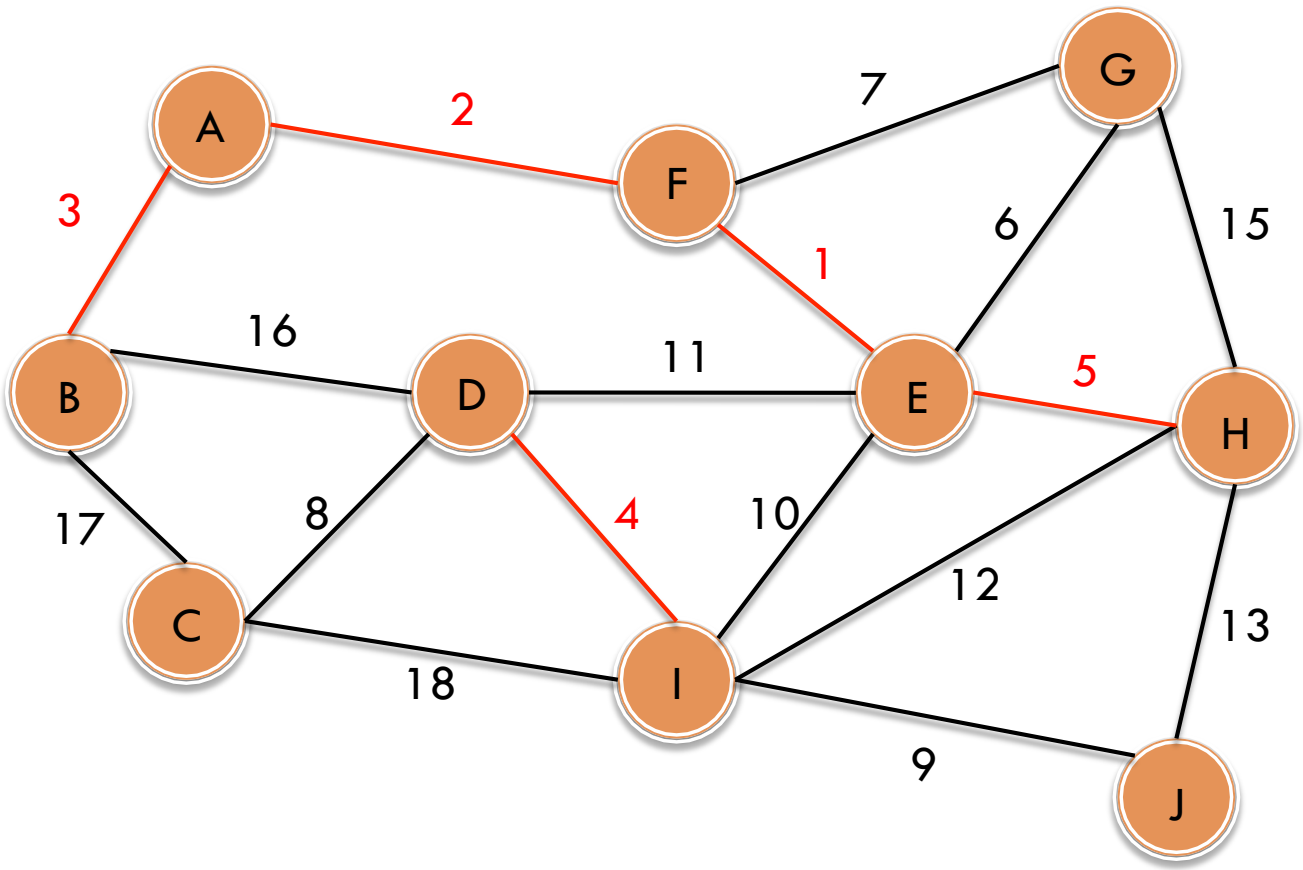
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

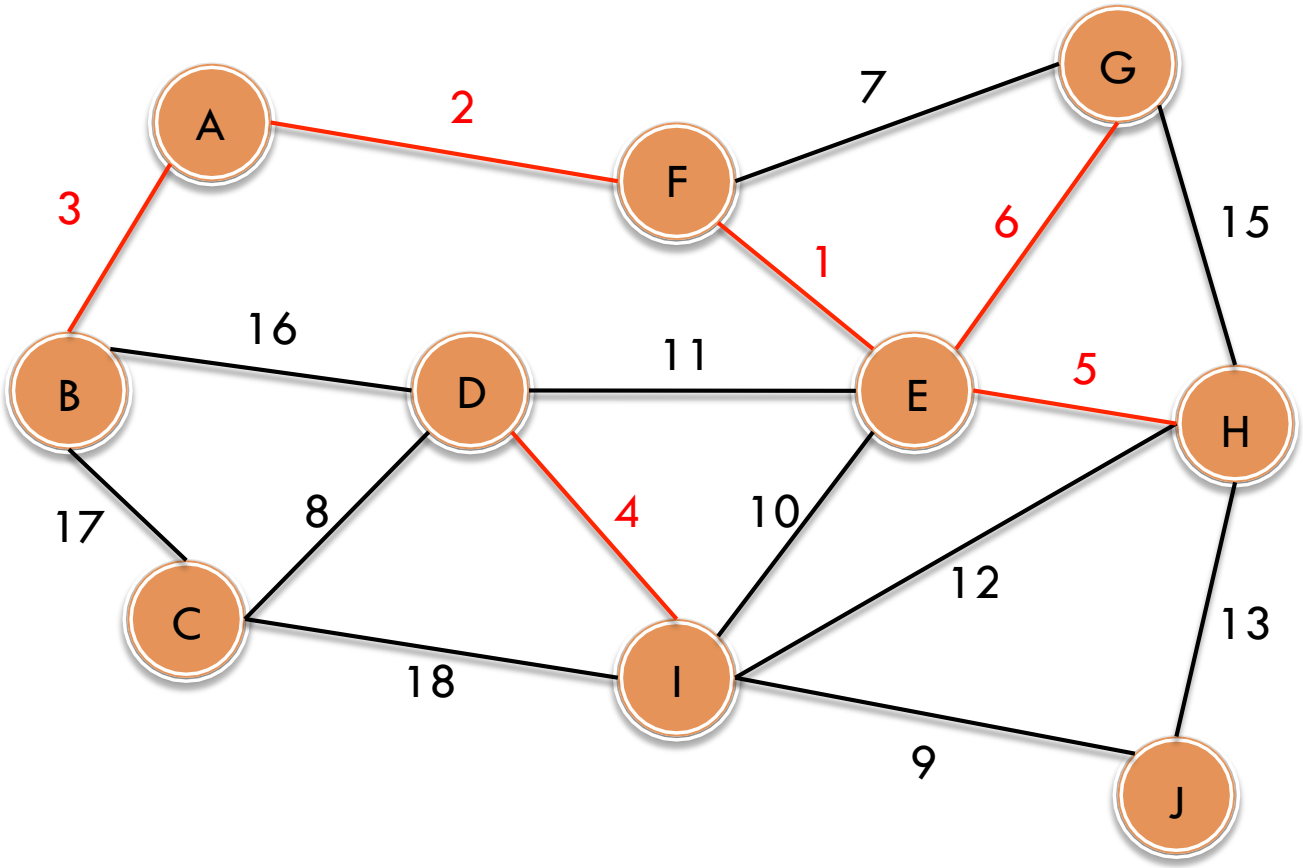
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

Kruskal Algorithm

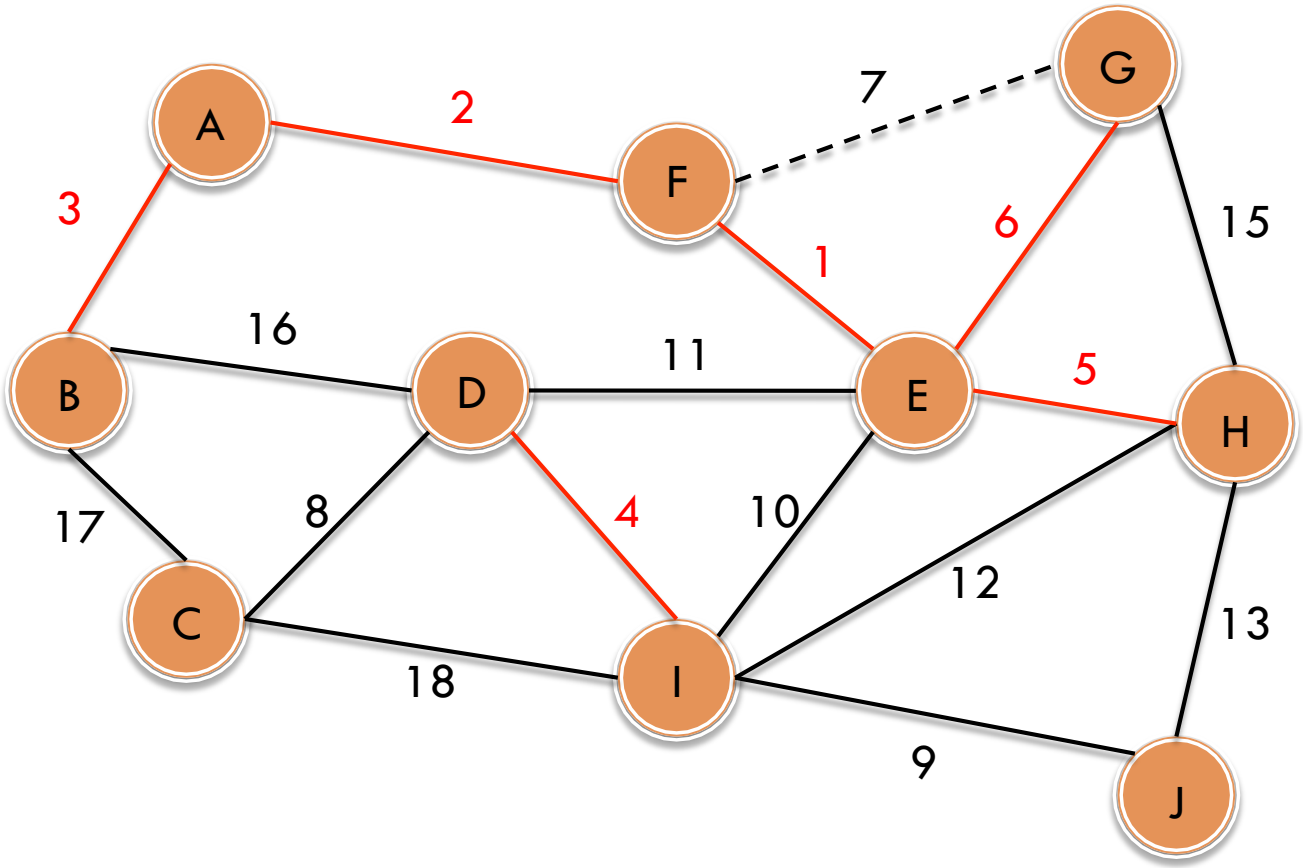


Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
F – G	7
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

Kruskal Algorithm

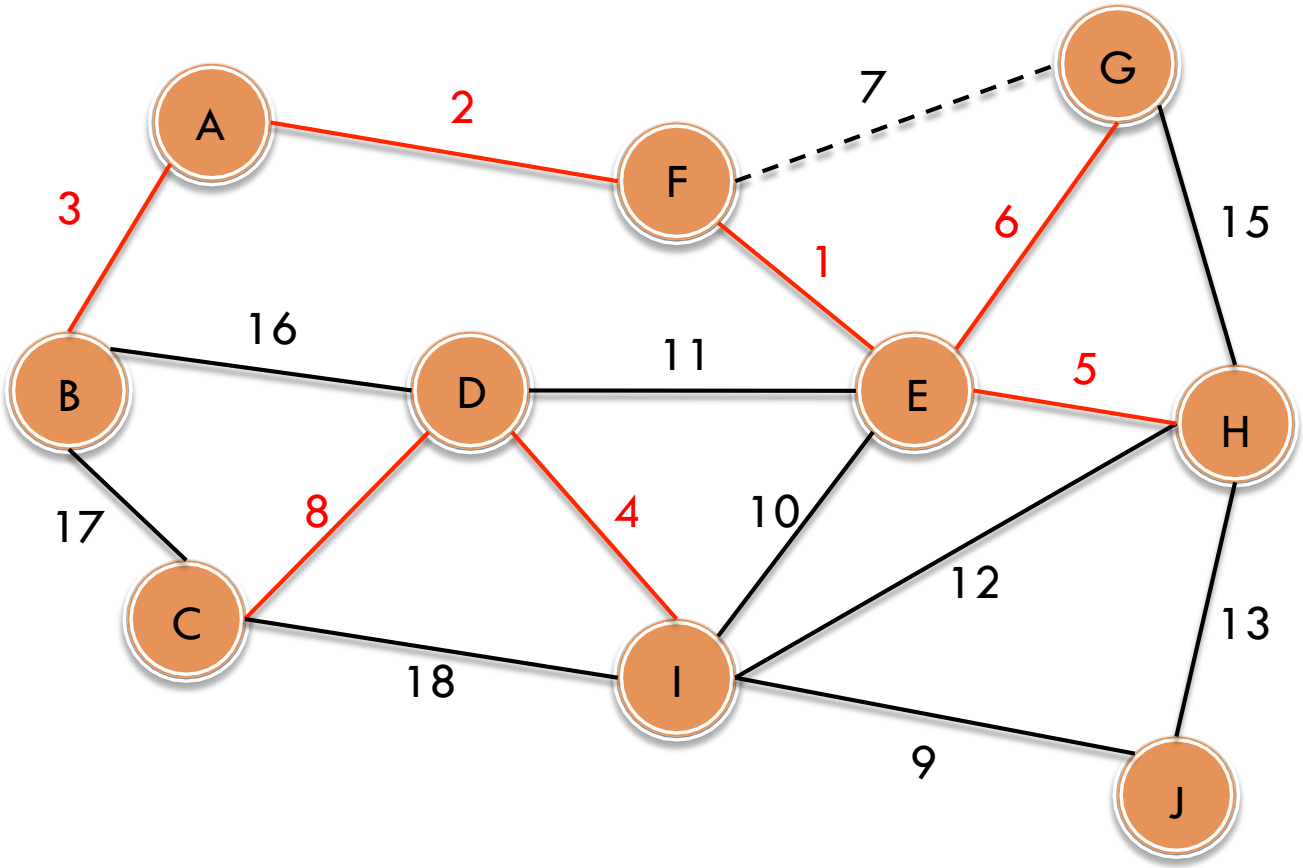
Edge F – G forms a cycle



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

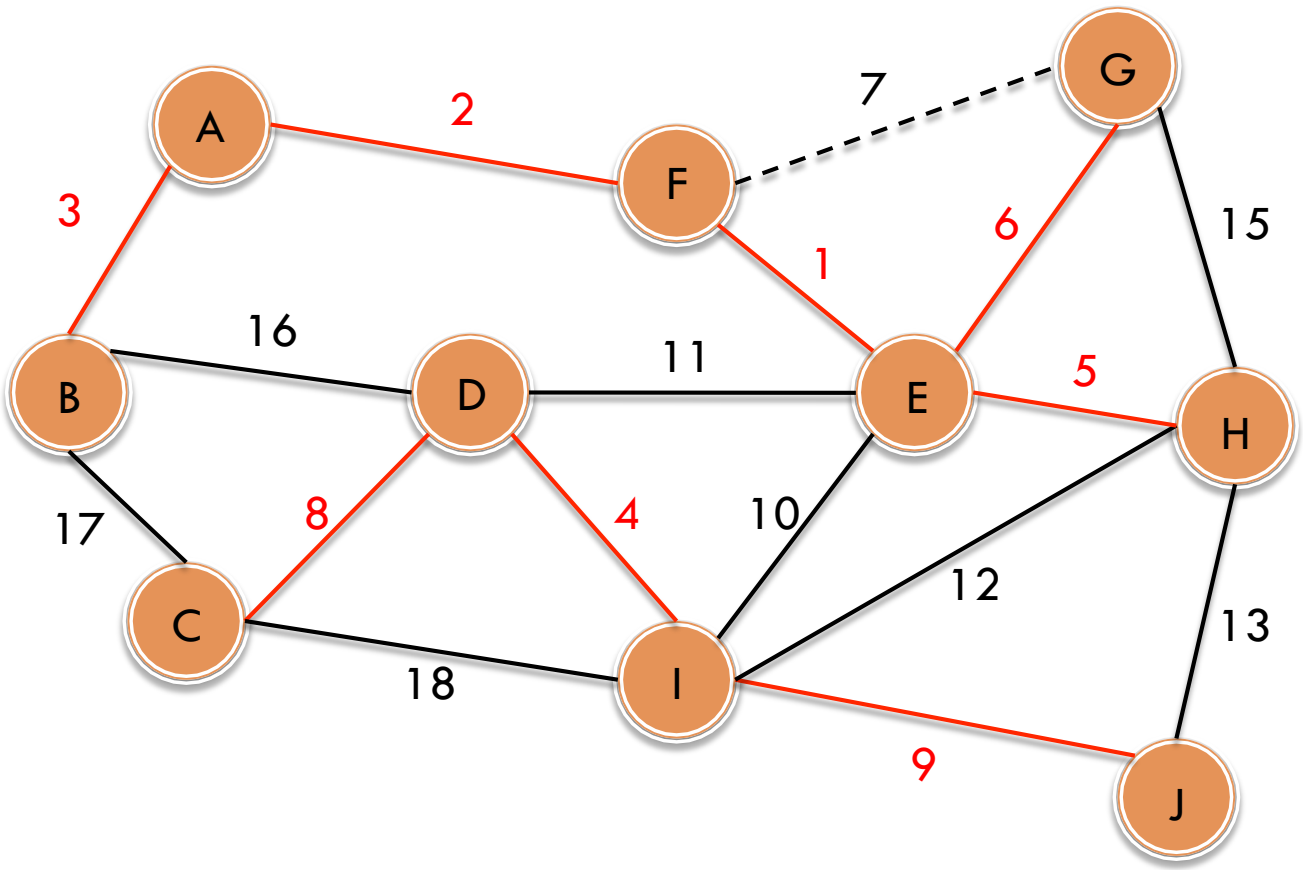
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

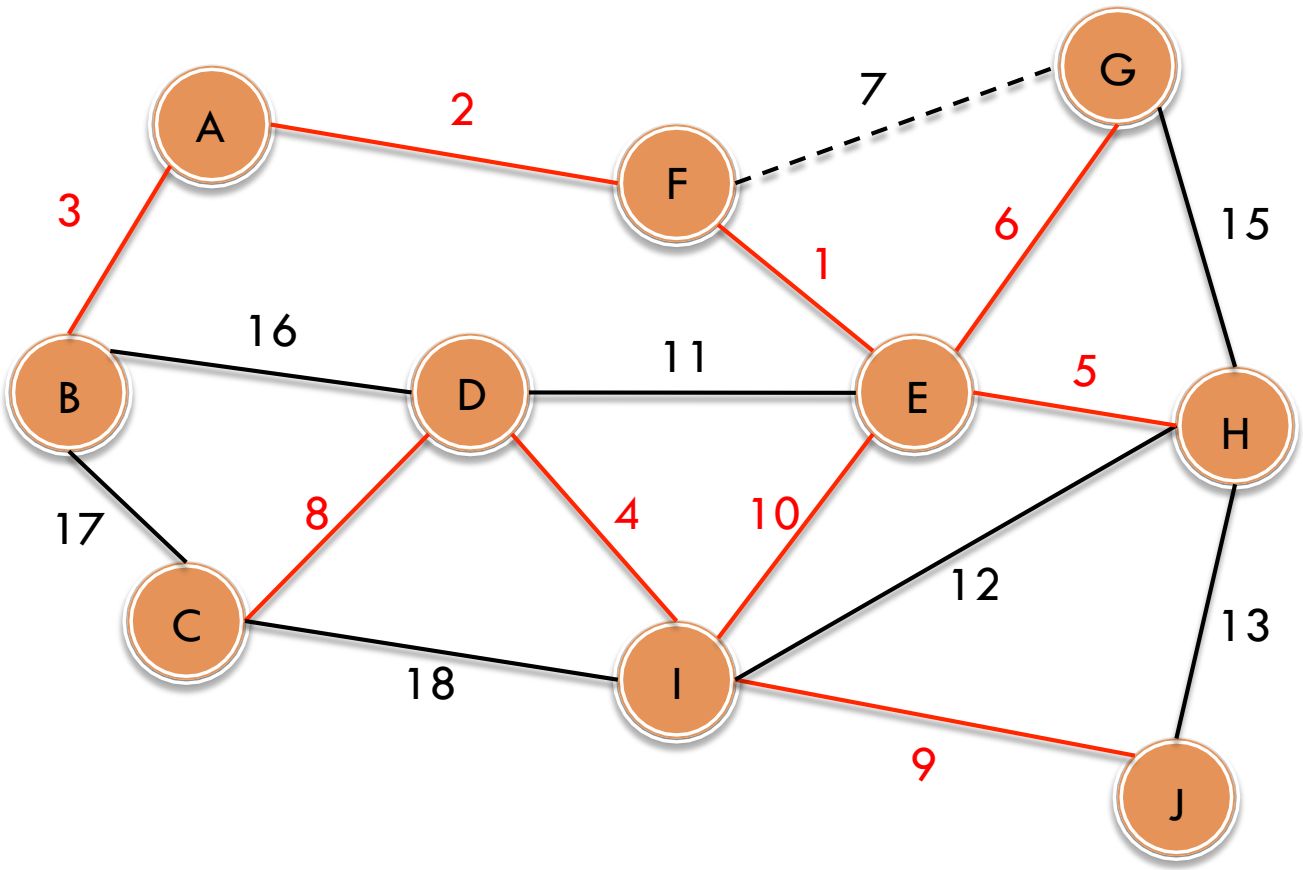
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

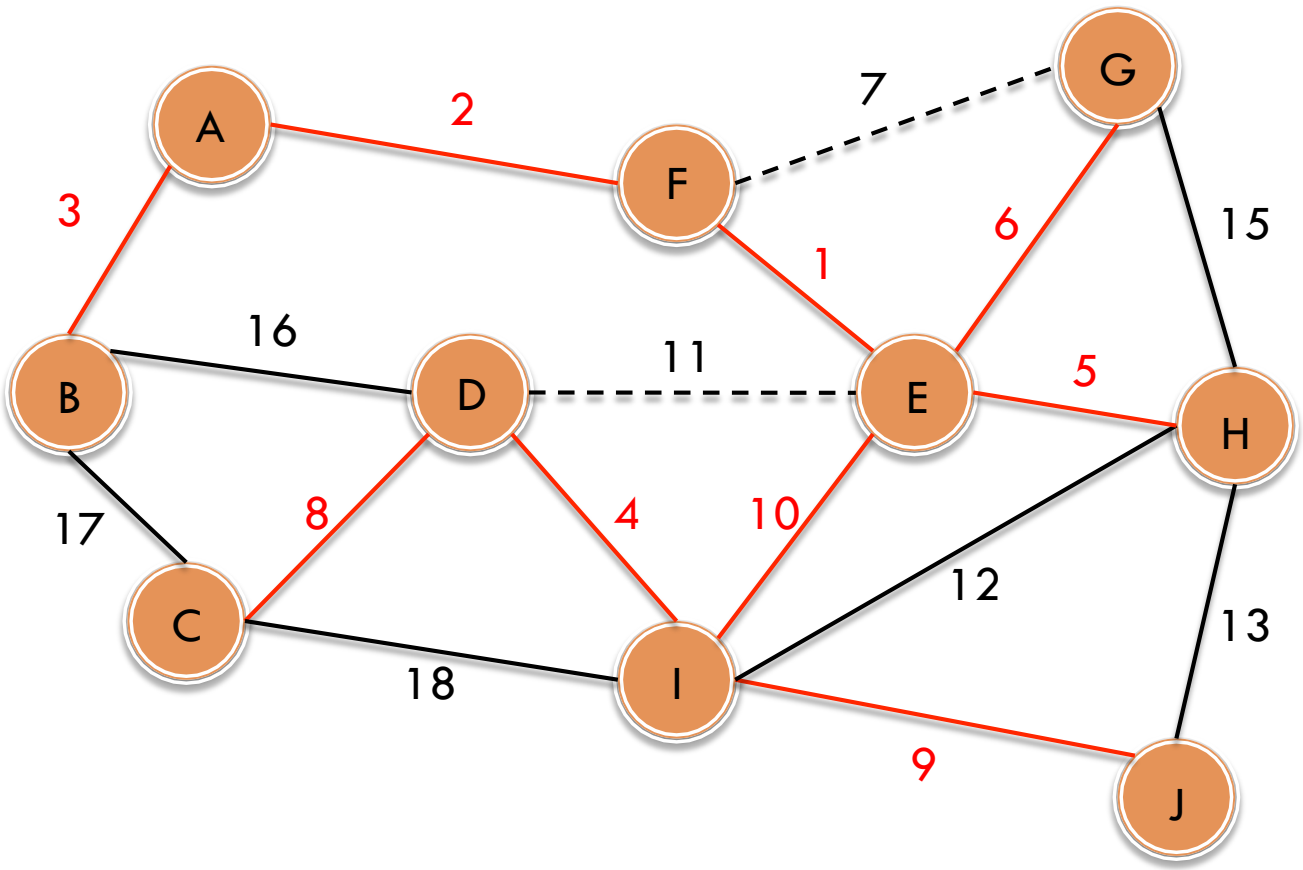
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
D – E	11
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

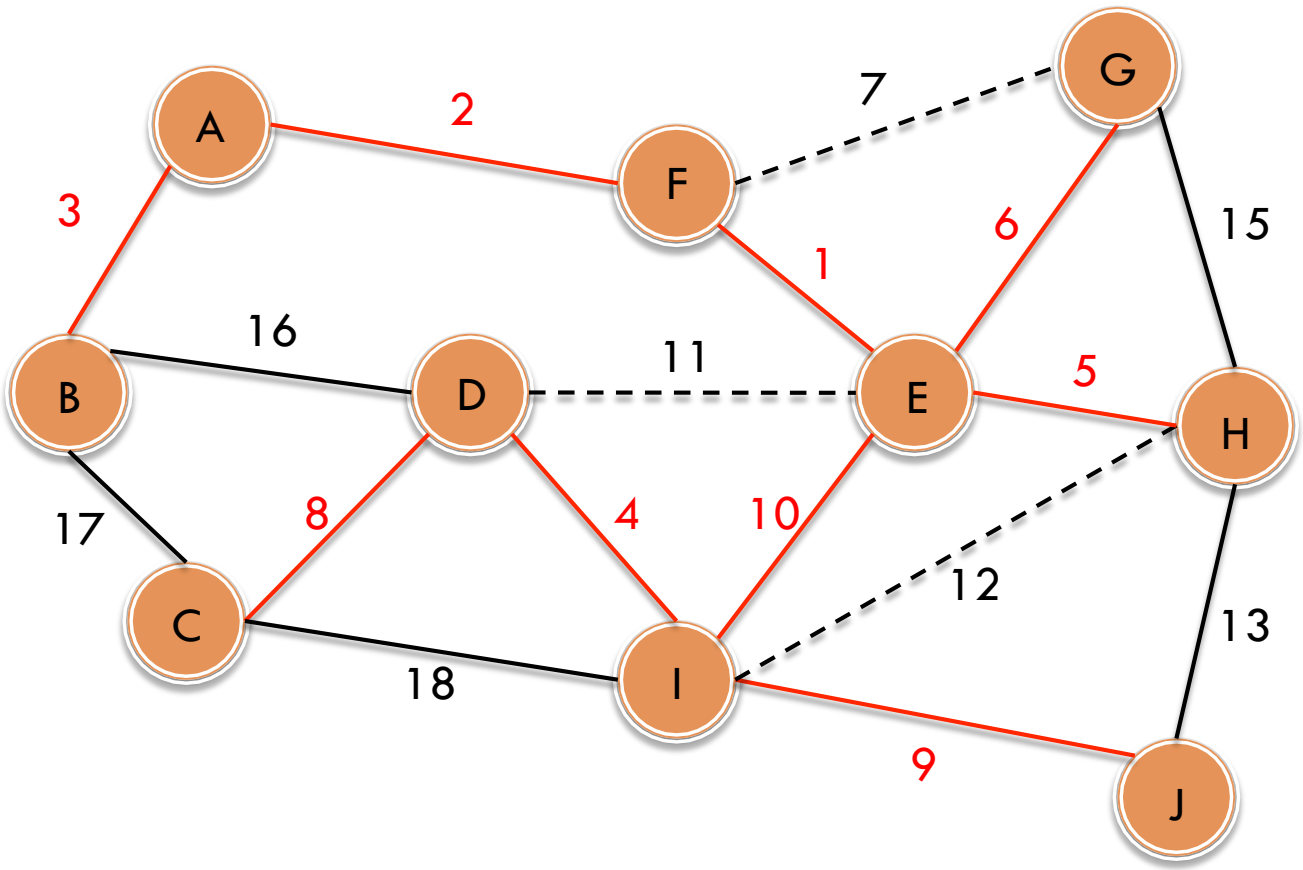
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
H – I	12
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

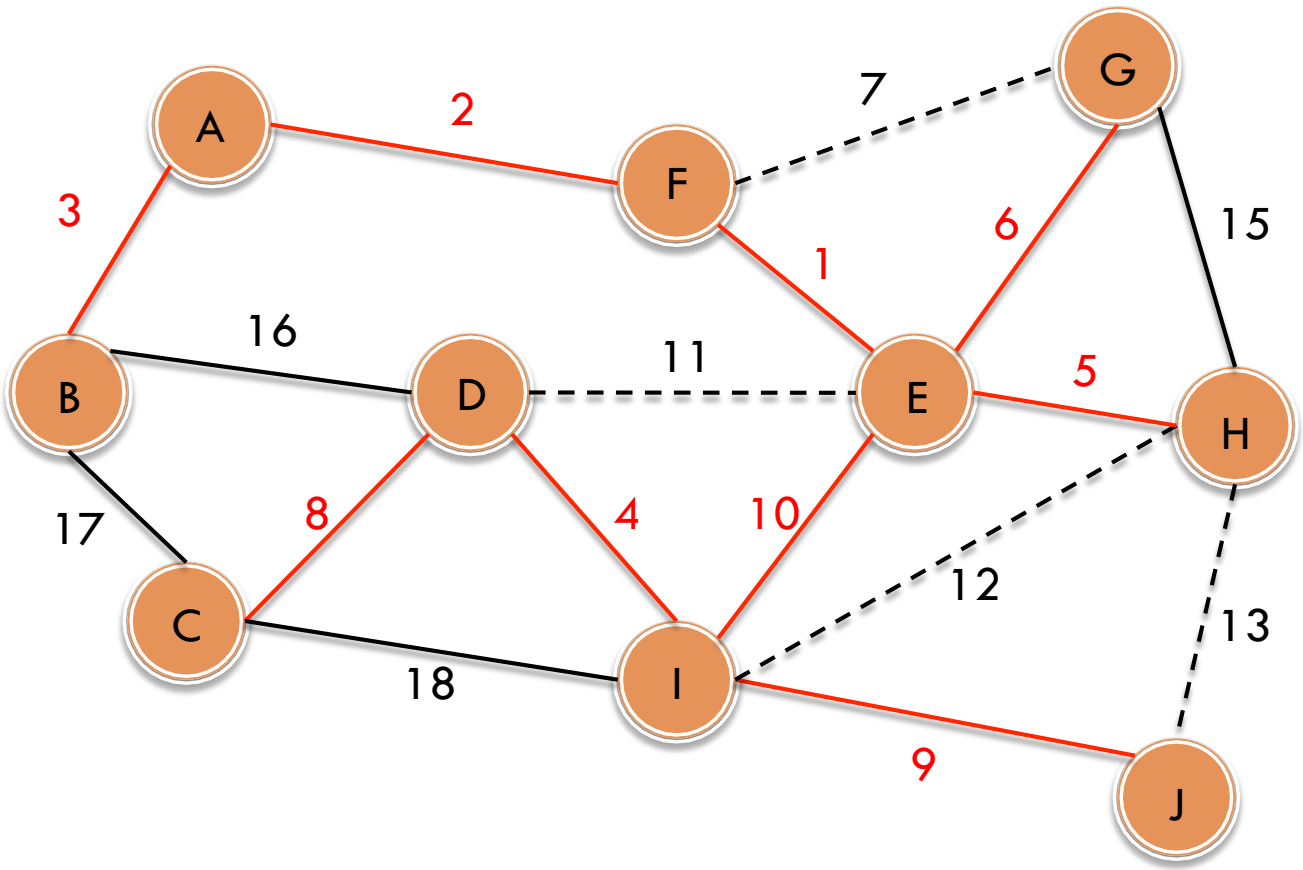
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
H – J	13
G – H	15
B – D	16
B – C	17
C – I	18

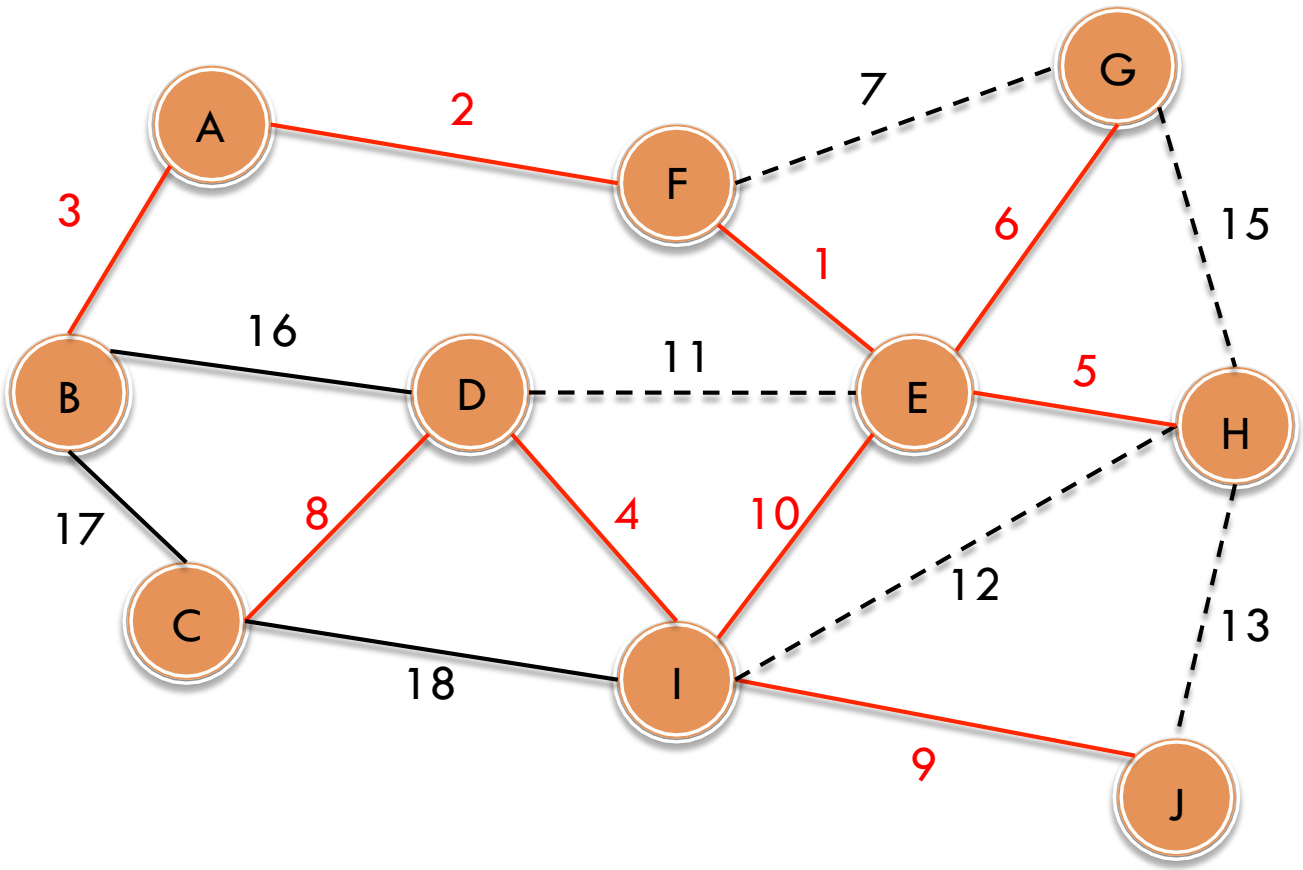
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
~~H – J~~	~~13~~
G – H	15
B – D	16
B – C	17
C – I	18

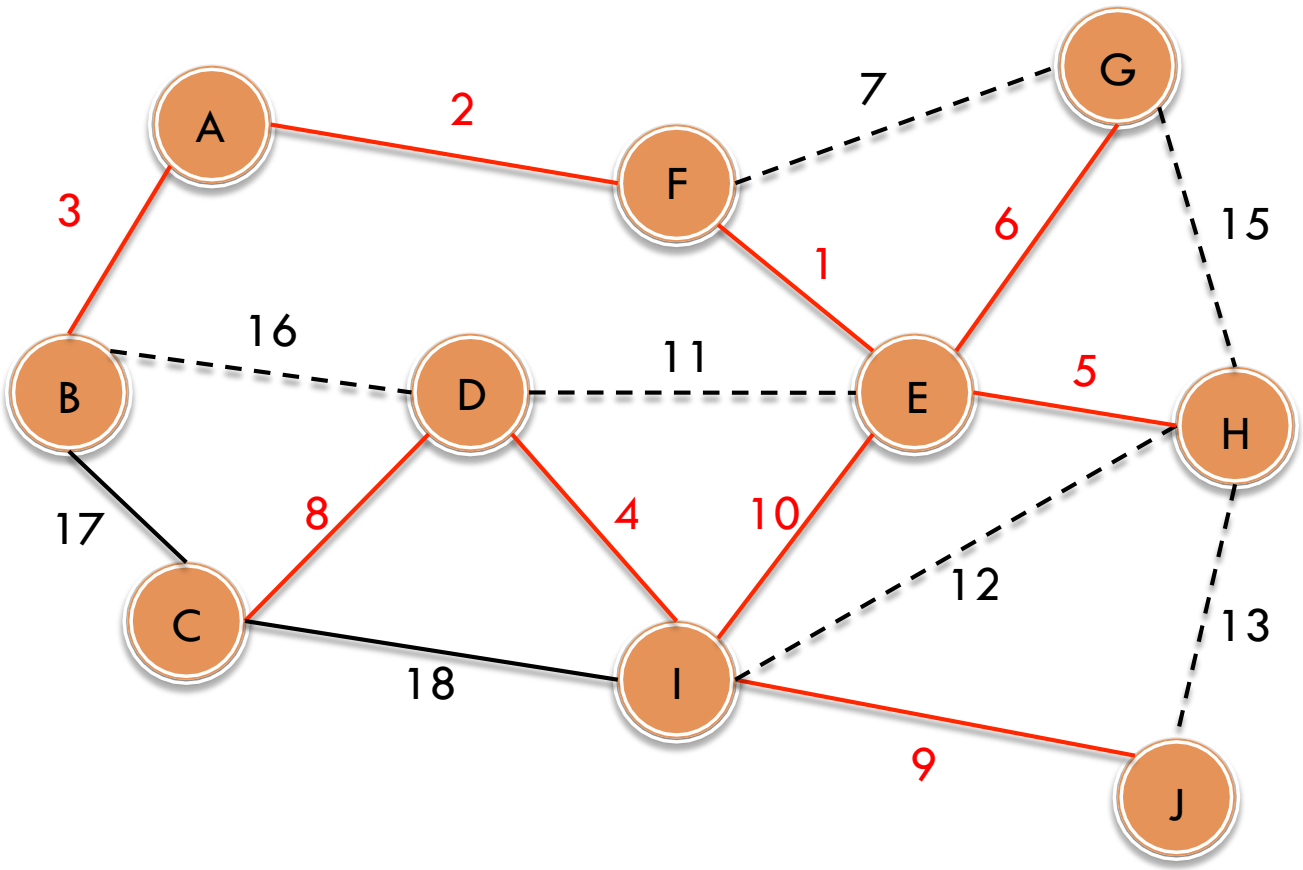
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
~~H – J~~	~~13~~
~~G – H~~	~~15~~
B – D	16
B – C	17
C – I	18

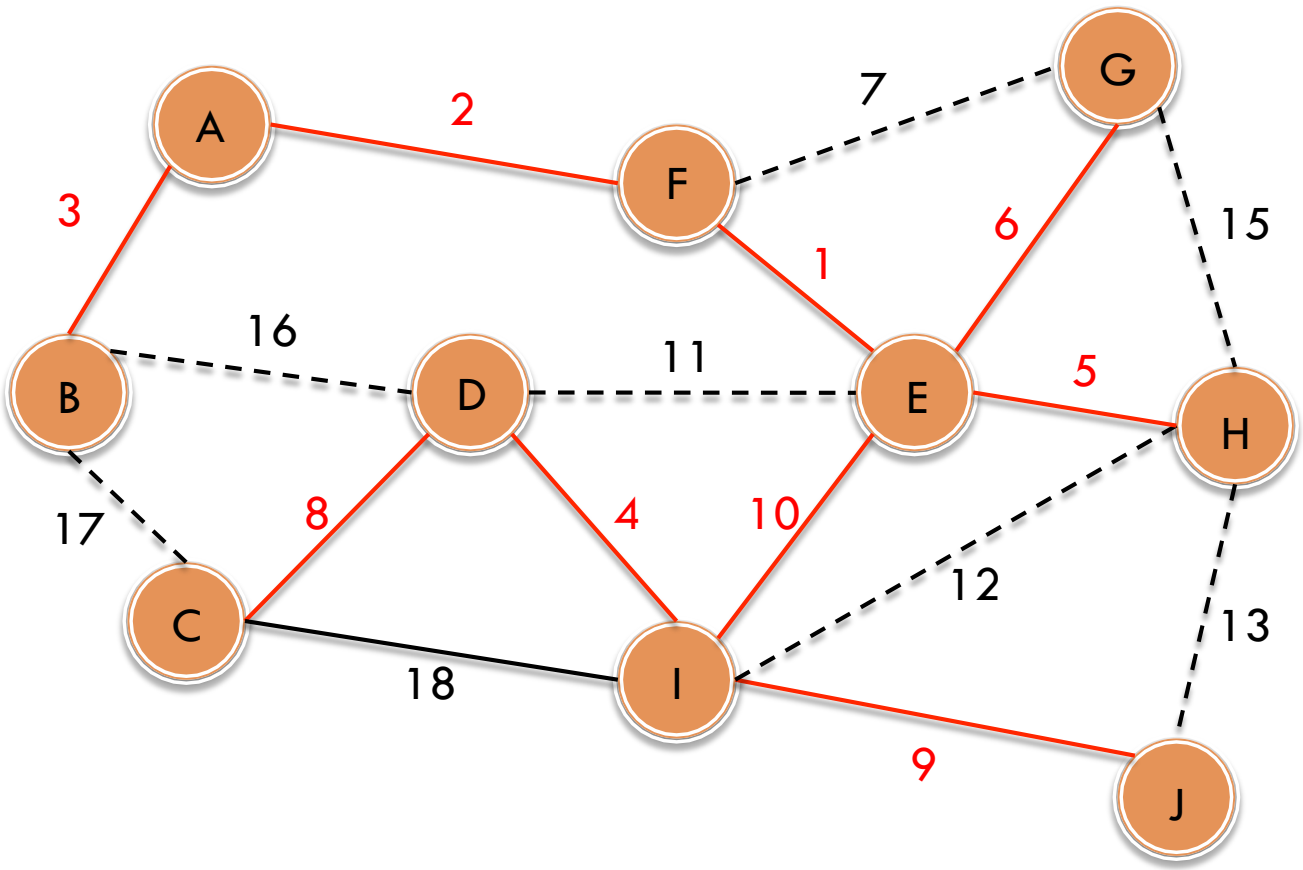
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
~~H – J~~	~~13~~
~~G – H~~	~~15~~
~~B – D~~	~~16~~
B – C	17
C – I	18

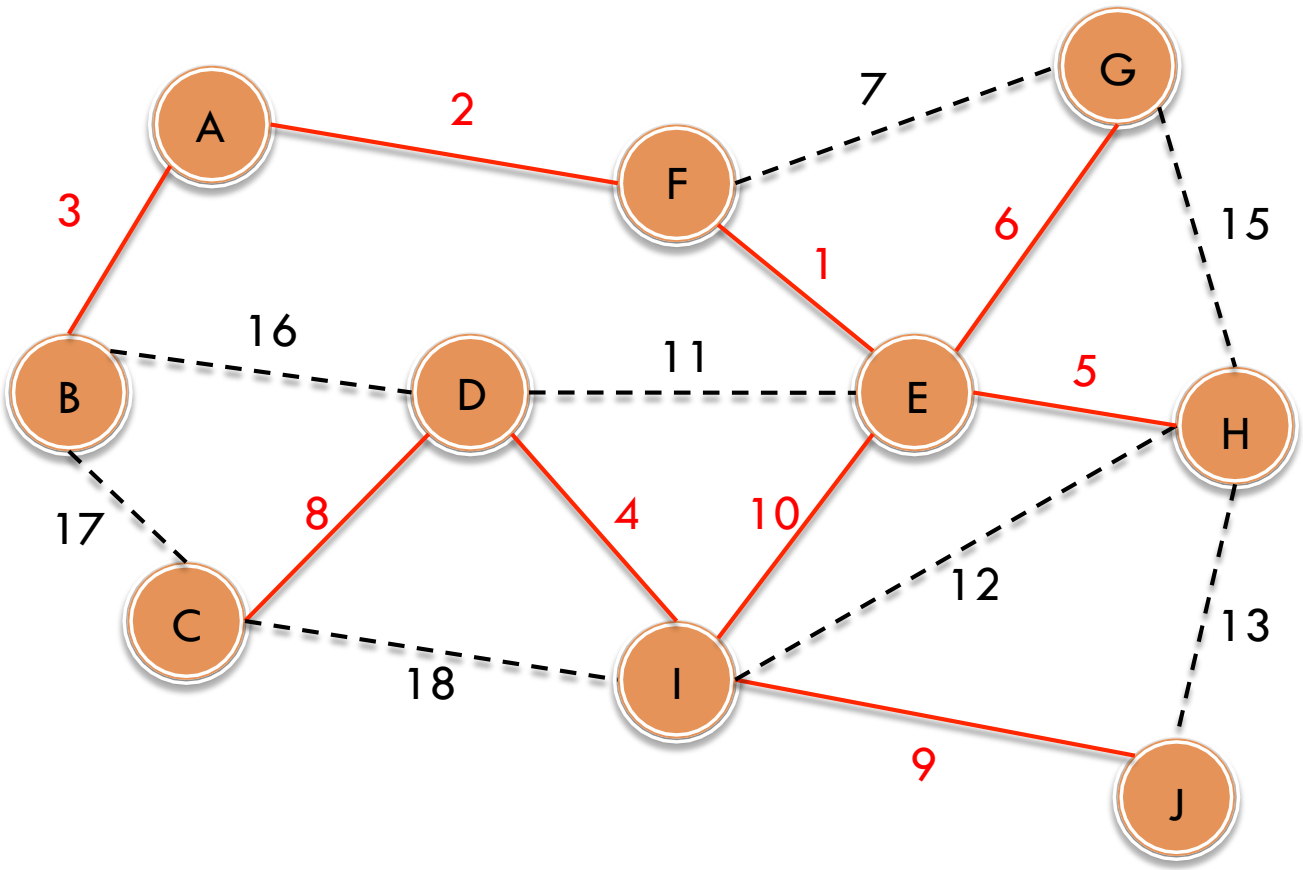
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
~~H – J~~	~~13~~
~~G – H~~	~~15~~
~~B – D~~	~~16~~
~~B – C~~	~~17~~
C – I	18

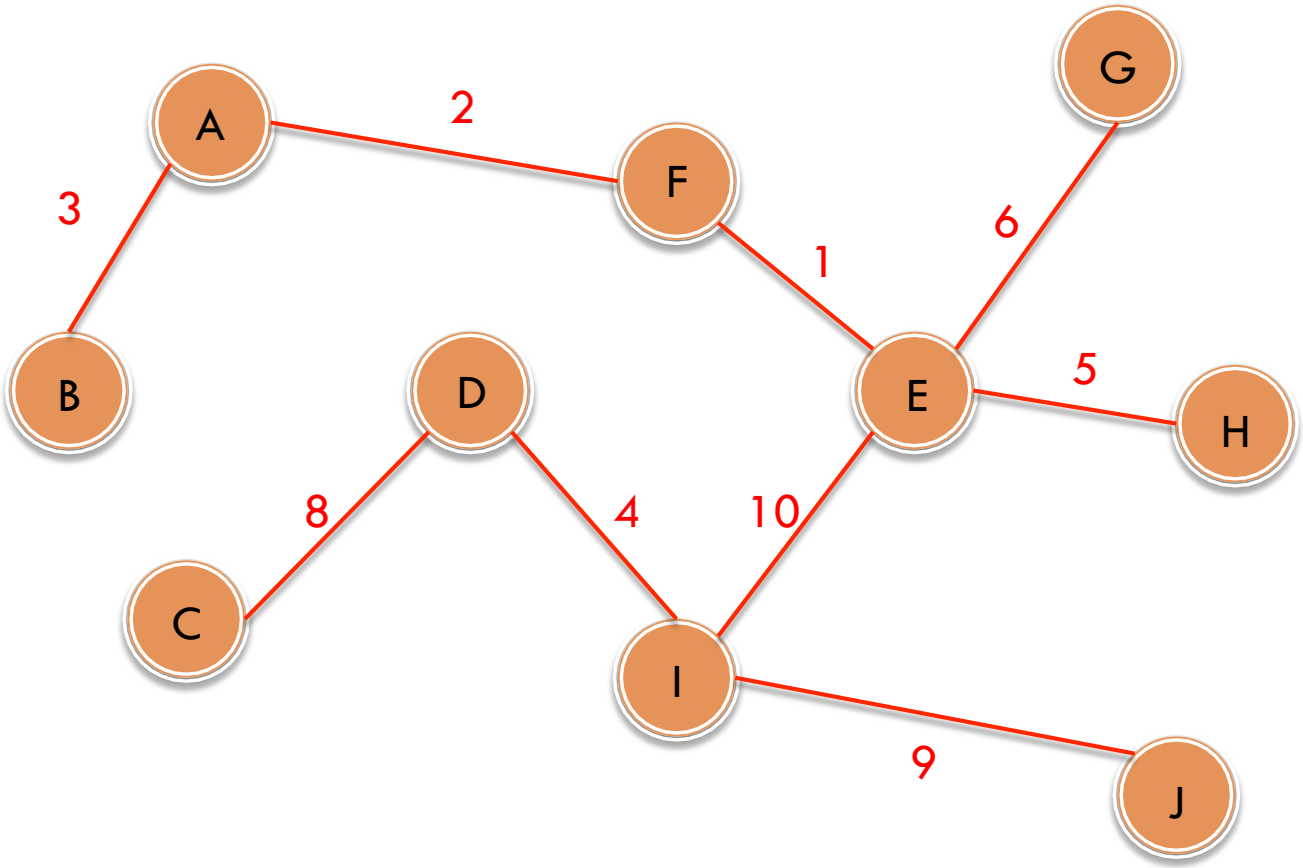
Kruskal Algorithm



Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
~~F – G~~	~~7~~
C – D	8
I – J	9
E – I	10
~~D – E~~	~~11~~
~~H – I~~	~~12~~
~~H – J~~	~~13~~
~~G – H~~	~~15~~
~~B – D~~	~~16~~
~~B – C~~	~~17~~
~~C – I~~	~~18~~

Kruskal Algorithm



Cost of the MST: 48

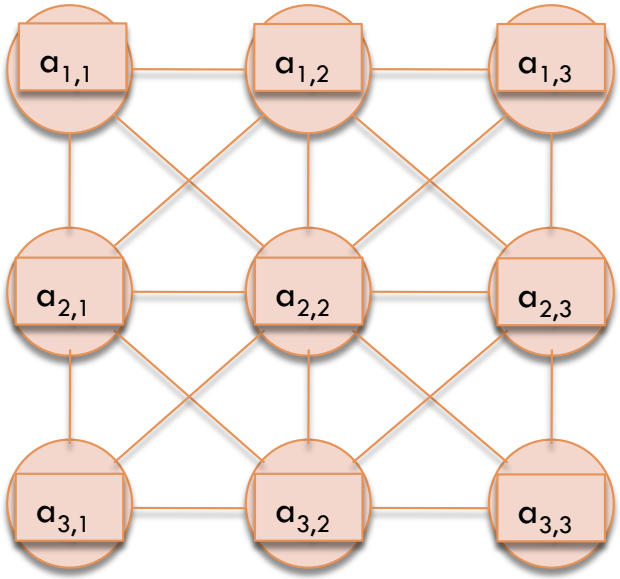
Sorted Edges

Edge	Weight
E – F	1
A – F	2
A – B	3
D – I	4
E – H	5
E – G	6
C – D	8
I – J	9
E – I	10

MST Segmentation

- If we see an image as a graph, with te pixels as vertices, we can do a search in the image in the same way that we do in a graph.

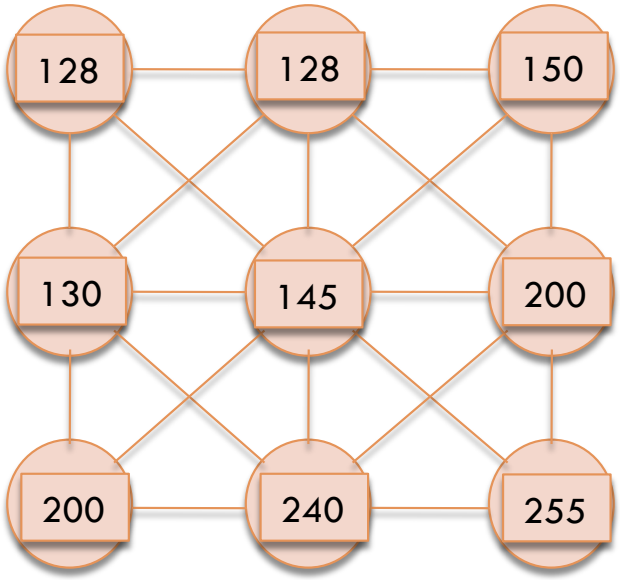
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$



MST Segmentation

- Each node (pixel) has a value associated to it, representing the intensity of that pixel, then each node can have a value in the rank of 0 to 255. Let's see an example:

128	128	150
130	145	200
200	240	255

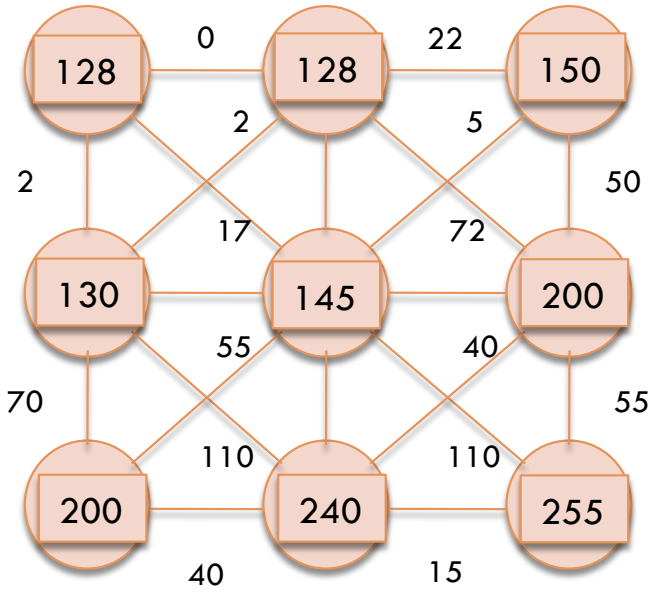


MST Segmentation

- If we want to find the MST of a graph, this must be weighted, then, What would be the value of the edges in an image?
- That's easy, Consider the edge that joins two vertices u and v. We will define the weight of that edge as the absolute value of the difference between the two vertices, then $w(e_{uv}) = |I_u - I_v|$, where I_k represents the intensity of the k^{th} vertex.

MST Segmentation

- For the graph shown before, we have:



MST Segmentation

- The main idea of the MST Segmentation algorithm is to divide an image into multiple MST's. Let's define $C = (C_1,\ldots,C_m)$ as the set of MST's in the image.

- Be *w(e)* the weight of edge e, let's define the Internal Difference of a MST, to be the edge with the largest weight in that component.

$$Int(C_k) = \max_{e \in C_k} w(e)$$

- The Minimum Internal Difference between two components is defined by:

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$$

MST Segmentation

- Be $|C_i|$ the number of elements in component C_i and K a constant parameter, define τ (C_i) as a threshold function based on the size of the i^{th} component:

$$\tau(C_i) = \frac{K}{|C_i|}$$

- Using the Kruskal algorithm, *"two vertices u and v of different components C_u and C_v respectively are considered of the same component if $w(e_{uv})$ is less than $Mint(C_1,C_2)$, if that happens, merge both components into a single one, otherwise nothing happens"*.

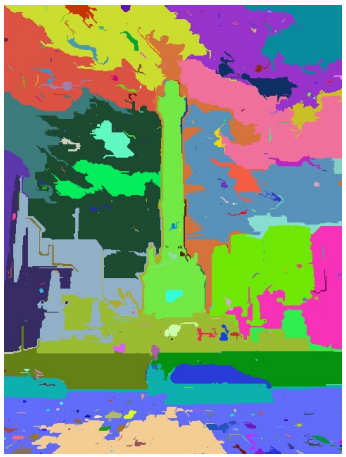
MST Segmentation

0) $MST \leftarrow$ empty
1) sort all the vertices $e_i \in E$ according to their weight
2) $i \leftarrow 0$
3) Be u_i and v_i the vertices joined by the edge e_i,
where $u_i, v_i \in V$ and $e_i \in E$
4) if($FindSet(u_i) \neq FindSet(v_i)$)
5) add e_i to MST
6) $UnionSet(u_i, v_i)$
7) $i \leftarrow i + 1$
8) if($i \leq |E|$)
9) return to 3
10) else
11) quit

Add this expression:
And $w(e_i) < Mint(FindSet(u_i), FindSet(v_i))$

MST Segmentation

K = 10000



K = 15000

K = 20000

MST Segmentation

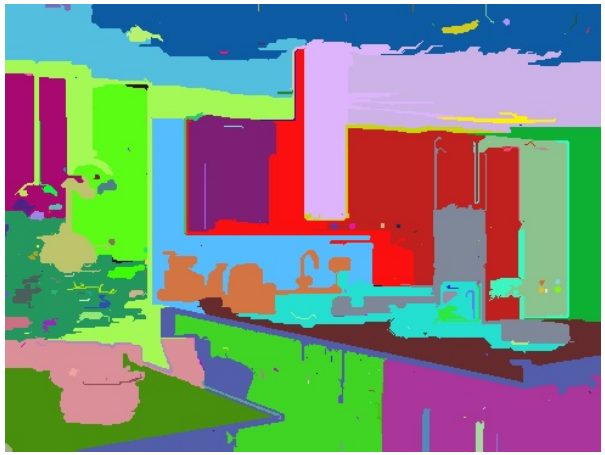
K = 10000

K = 15000

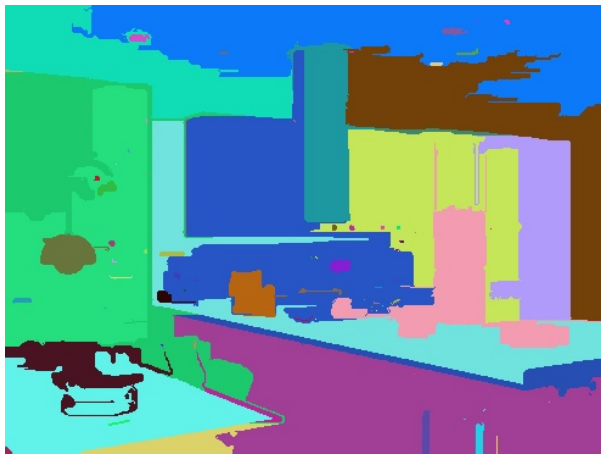
K = 20000

MST Segmentation

K = 10000



K = 15000



K = 20000

Conclusions

Ritsumeikan University

Conclusions

- Avoid components formed by few components.
- Wide areas of the same color can be separated in different components.
- The value of K must be defined empirically.
- Apply the same concept for each channel (red, green and blue), instead for the intensity (to do).
- Apply more rules to avoid that wide areas of the same color be separated (to do).