Kruskal & Union-find

YAO ZHAO

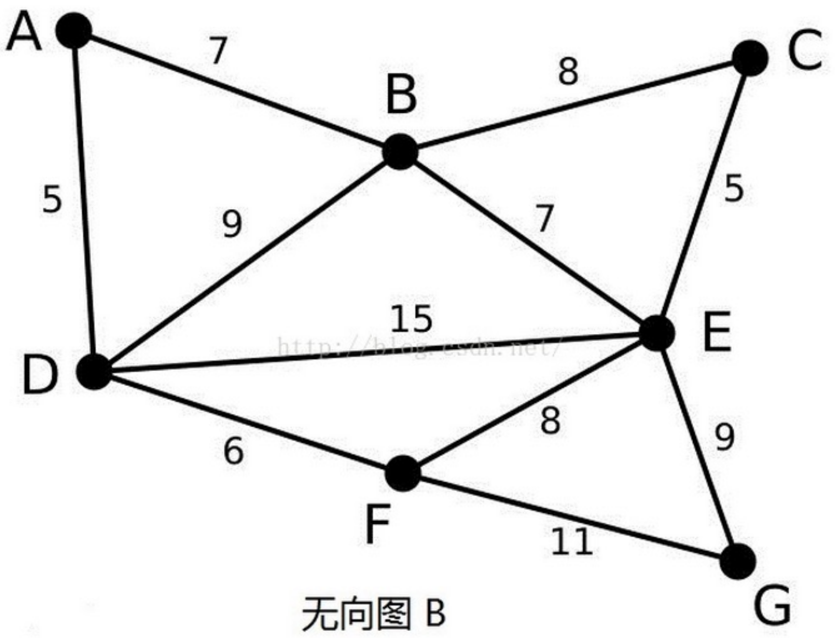
Implementation: Kruskal's Algorithm

Implementation. Use the union-find data structure.

- Build set T of edges in the MST.
- Maintain set for each connected component.
- $O(m \log n)$ for sorting and $O(m\, \underbrace{\alpha\,(m, n)}_{\text{essentially a constant}})$ for union-find.

$m \leq n^2 \Rightarrow \log m$ is $O(\log n)$

Kruskal



无向图 B

Kruskal:

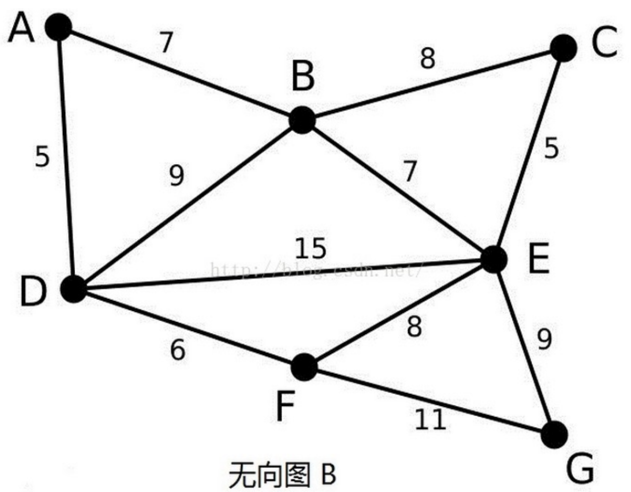
1. Sorting all the sides
2. Finding smallest bridge (n, m)
3. Whether node n and node m are in a same tree?
   If yes, skip
   If no, merge two trees
4. If the number of node is N, we should merge N-1 times.
5. When merge two trees, add the w value

How to merge two trees (n, m)? Union-find

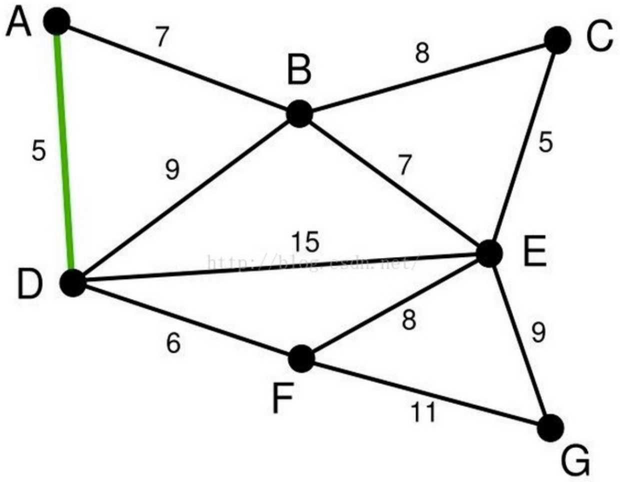
1. Find root of n and m respectively
2. If root of n equals to root of m, n and m is in a same tree. Skip
3. Get the height of root n and root m

```
 if(rootN.height > rootM.height) rootM.parent =rootN
else if(rootN.height<rootM.height) rootN.parent=rootM
else  rootM.parent=rootN   rootN.height++;
```

Sample Kruskal-1



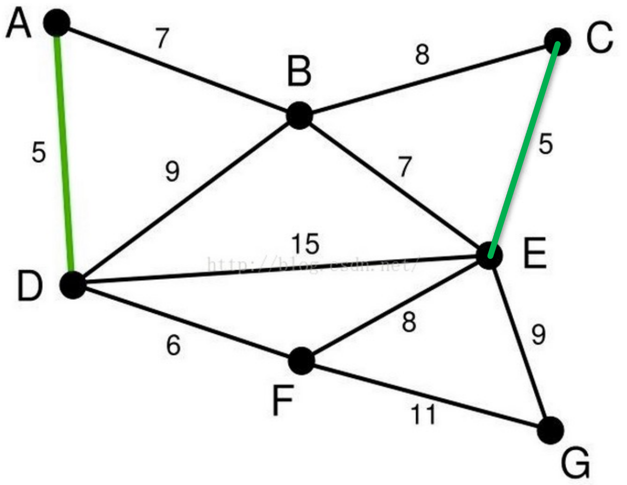
无向图 B



index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	0	0	0	0	0	0
weight	0	0	0	0	0	0	0

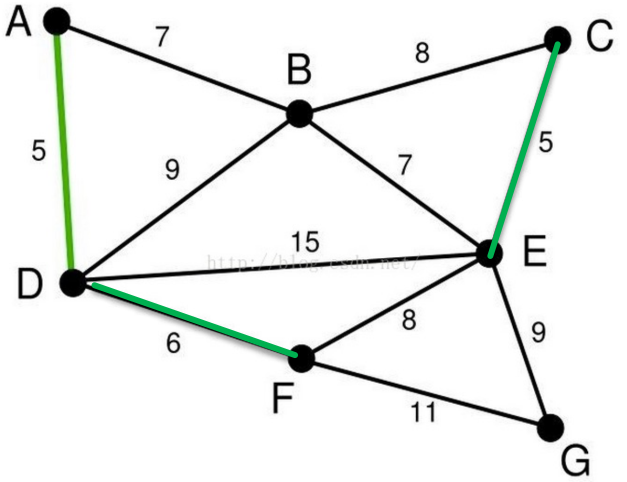
index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	0	0	1	0	0	0
weight	1	0	0	0	0	0	0

Sample Kruskal-2



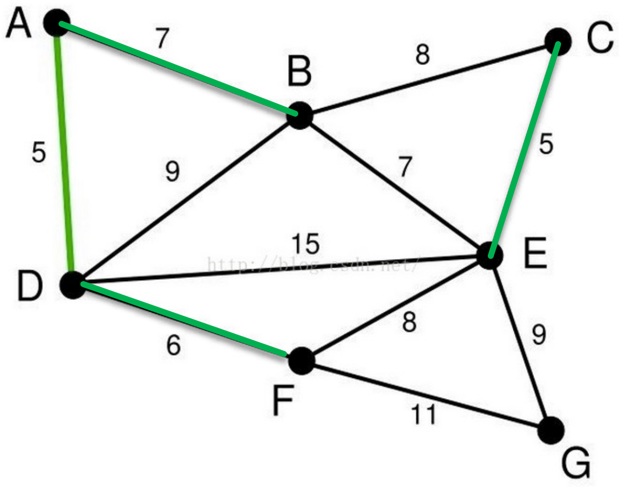
index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	0	0	1	3	0	0
weight	1	0	1	0	0	0	0

f(root).weigth<d(root).weight



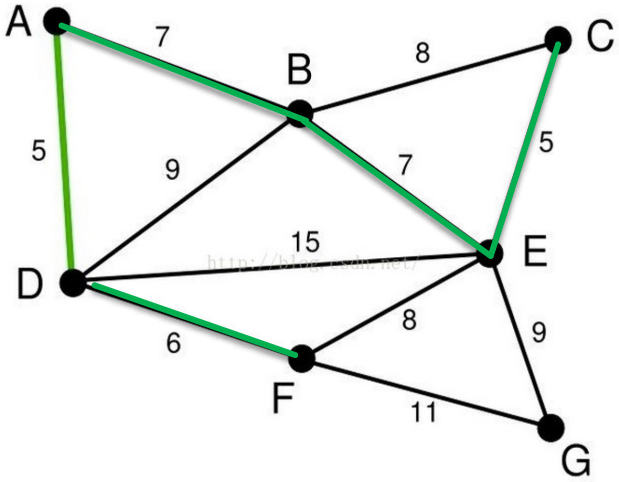
index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	0	0	1	3	1	0
weight	1	0	1	0	0	0	0

Sample Kruskal-3



b(root).weigth<a(root).weight
b.parent =a index

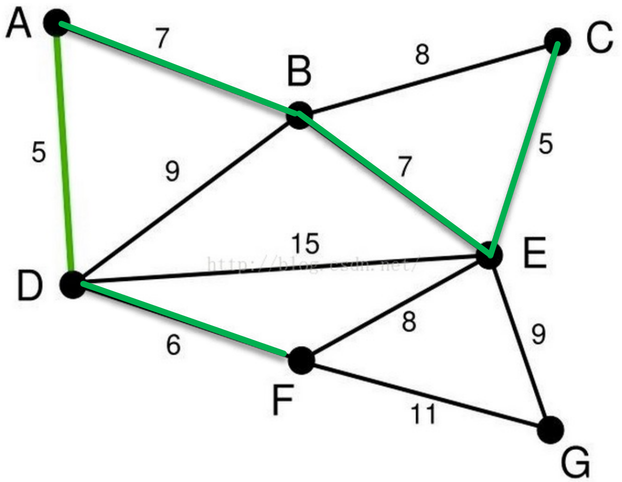
index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	1	0	1	3	1	0
weight	1	0	1	0	0	0	0

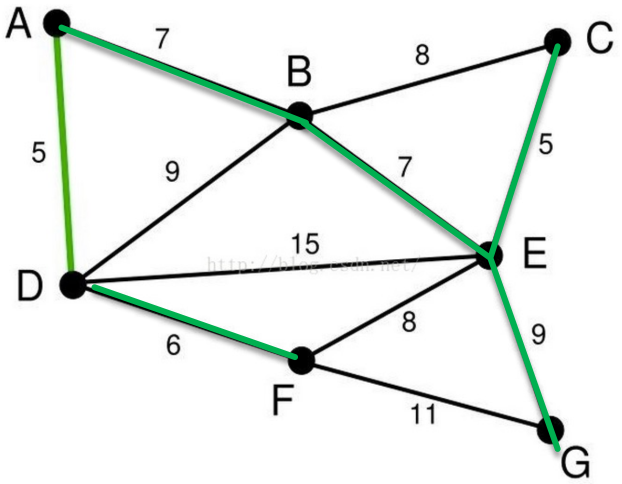


e(root).weigth==b(root).weight
c.parent =a index
a.weight++

index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	1	1	1	3	1	0
weight	2	0	1	0	0	0	0

Sample Kruskal-4





B (root)==C (root) skip
F (root) ==E (root) skip

e(root).weigth>g (root).weight
g.parent=a

index	1	2	3	4	5	6	7
node	A	B	C	D	E	F	G
parent	0	1	1	1	1	1	1
weight	2	0	1	0	0	0	0