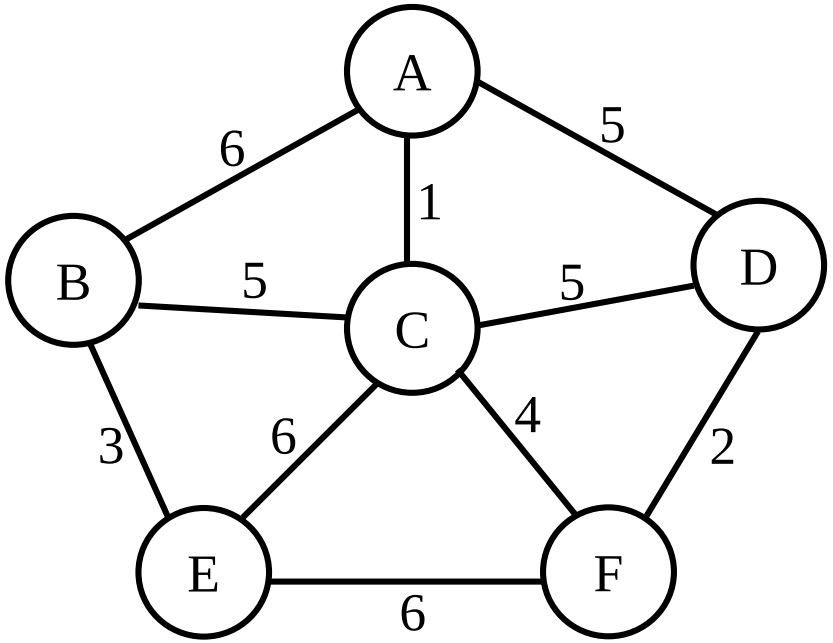
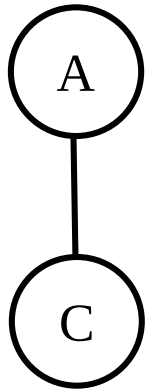
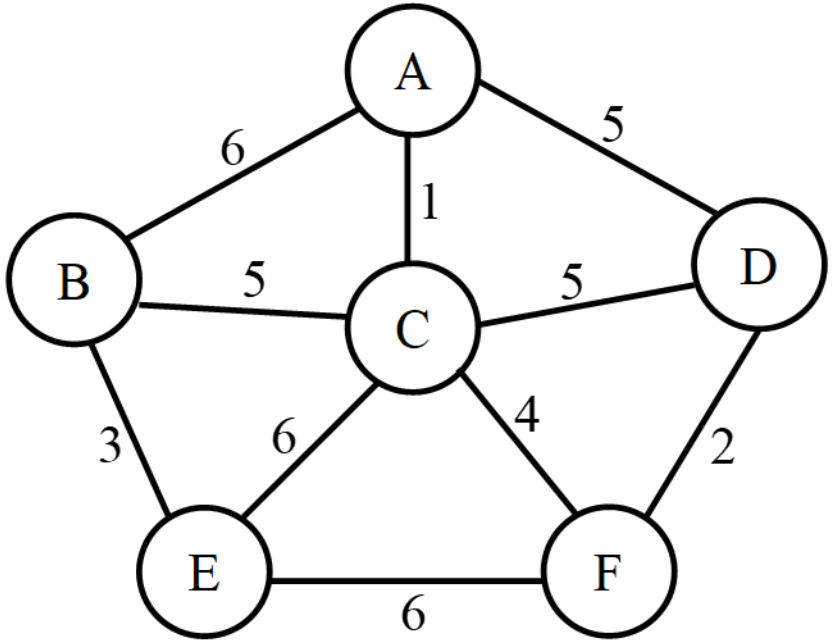
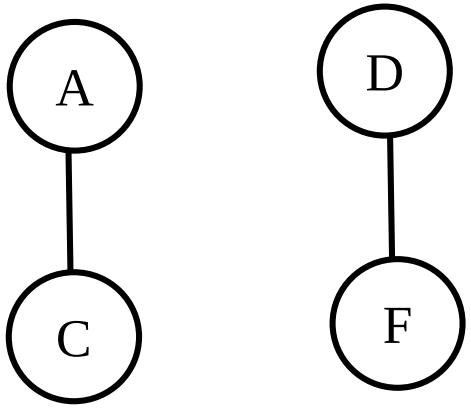
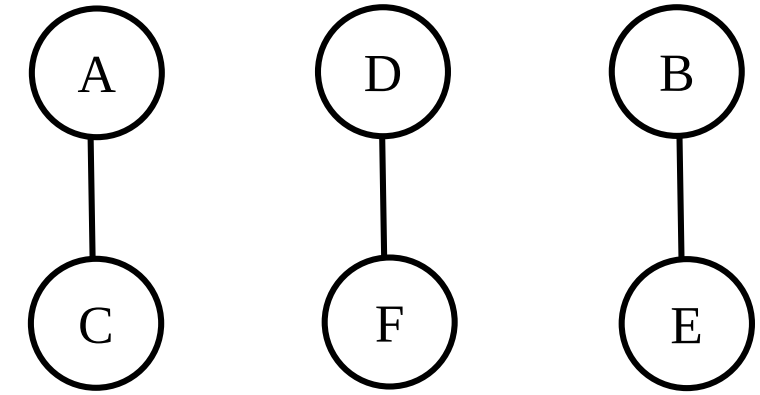
Implementing Kruskal's Algorithm

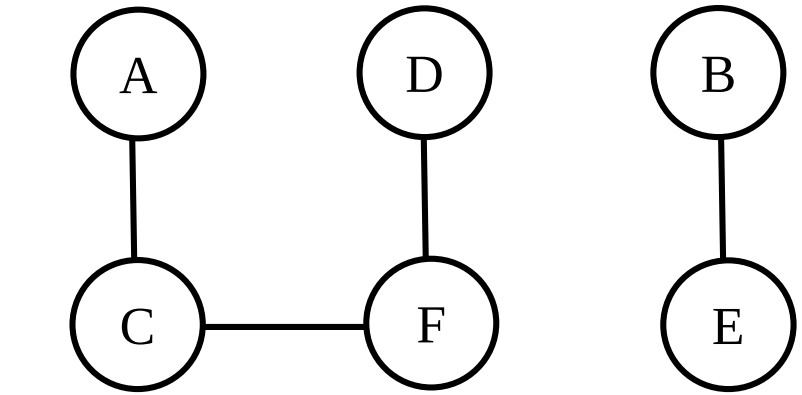


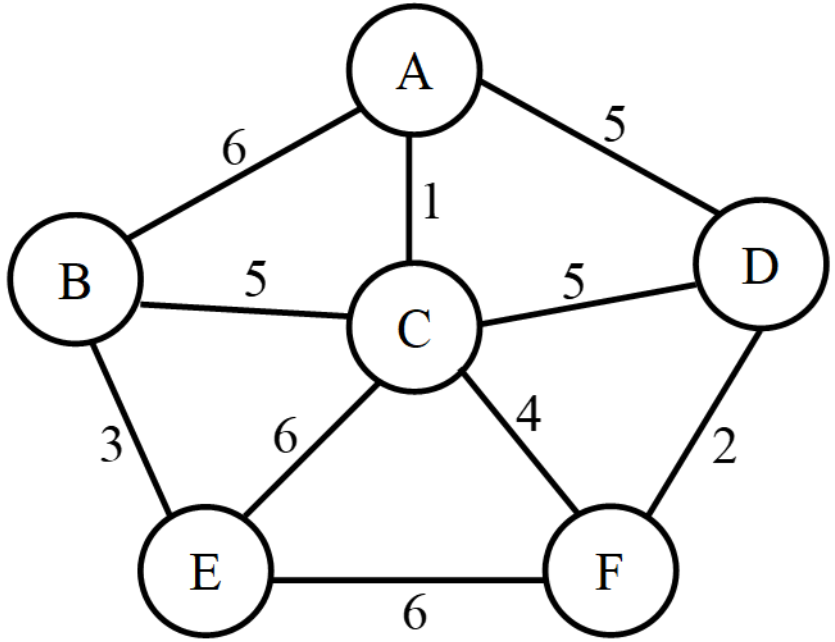

A
C

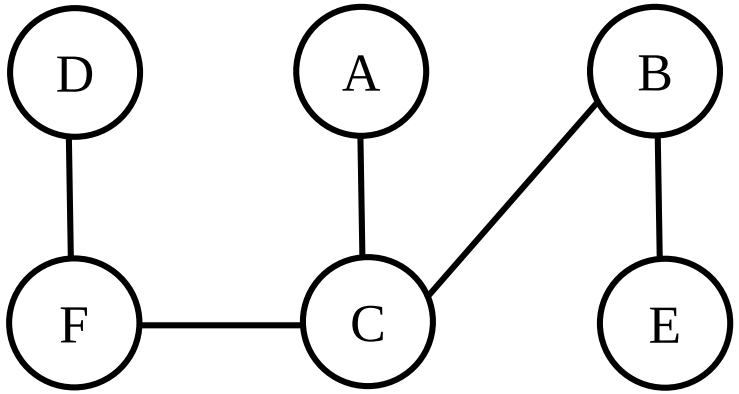

A
B
C
D
E
F
6
5
1
5
5
3
6
4
2
6

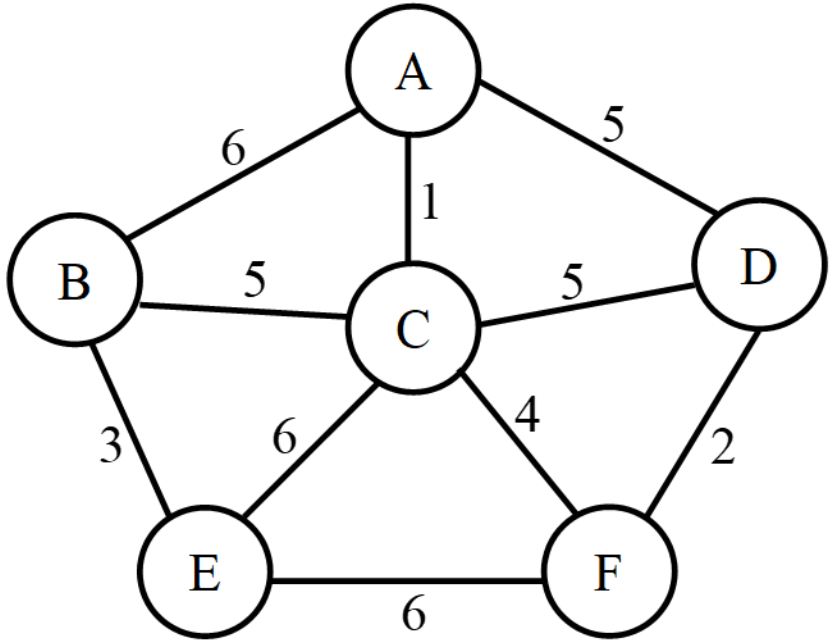

A
C
D
F


A
D
B
C
F
E


A
D
B
C
F
E


A
B
C
D
E
F
6
5
1
5
5
3
6
4
2
6


D
A
B
F
C
E


A
6
5
1
B
5
C
5
D
3
6
4
2
E
F
6

Disjoint Sets

- Data structure for grouping elements.
- Each element belongs to only one group at any given time.
- Each group has a Representative (Marker).
- Example: MST using Kruskal's Algorithm

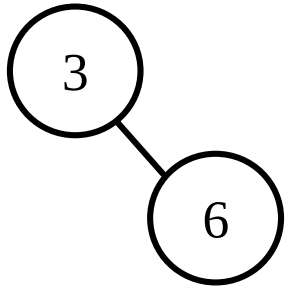
Two Operations

- find(e):
  - find the representative (marker) for the group the element e is in

- union(e_1, e_2):
  - merge the two groups the elements e_1 and e_2 are currently in (merge the two groups into one group)

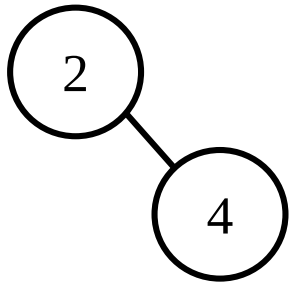
Conceptual Representation

`Union(3,6)`

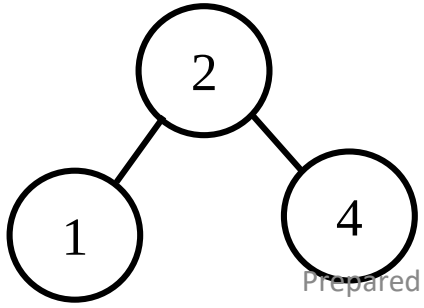


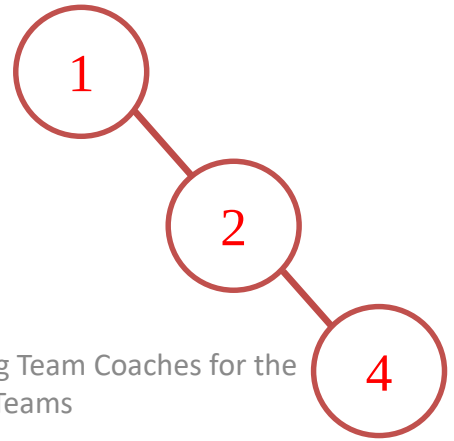
Conceptual Representation

Union(2,4)

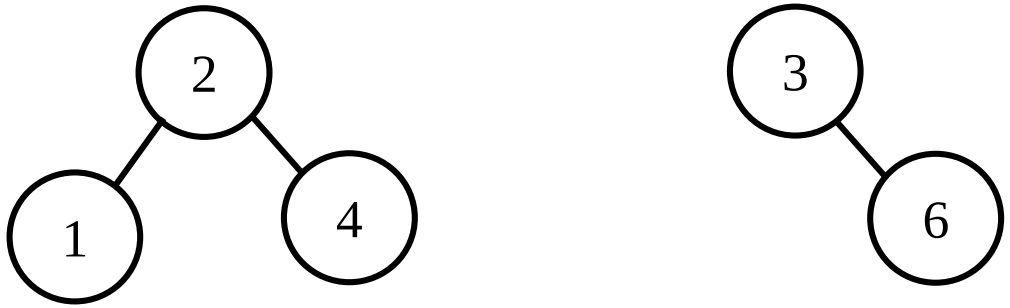


Union(1,4) [or union (1,2)]

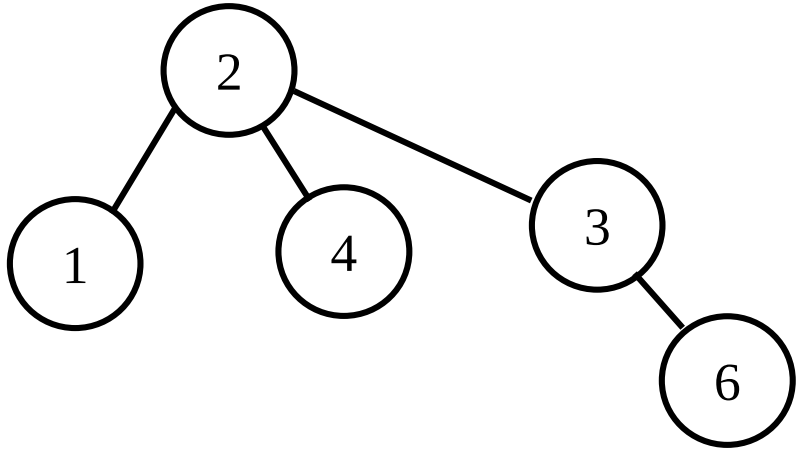




Conceptual Representation



Union(1,6) **[or union (2,6) or …]**



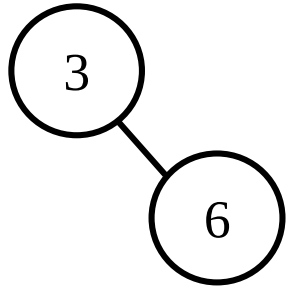
Implementation

Height table	0	0	0	0	0	0	0
Parent table	1	2	3	4	5	6	7
index	1	2	3	4	5	6	7

Implementation

Union(3,6)

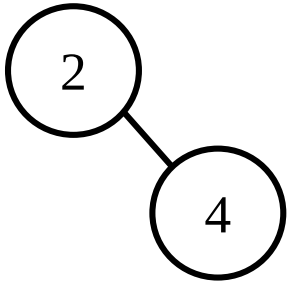
Height table	0	0	~~0~~1	0	0	0	0
Parent table	1	2	3	4	5	~~6~~3	7
index	1	2	3	4	5	6	7



Implementation

Union(2,4)

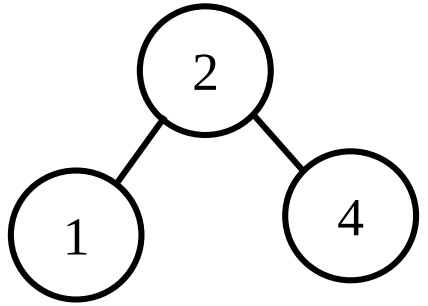
Height table	0	~~0~~1	1	0	0	0	0
Parent table	1	2	3	~~4~~2	5	3	7
index	1	2	3	4	5	6	7



Implementation

Union(1,4) [or union (1,2)]

Height table	0	1	1	0	0	0	0
Parent table	~~1~~2	2	3	2	5	3	7
index	1	2	3	4	5	6	7



Implementation

Union(1,6) [or union (2,6) or …]

Height table	0	~~1~~2	1	0	0	0	0
Parent table	2	2	~~3~~2	2	5	3	7
index	1	2	3	4	5	6	7