POSCAT Seminar 10 : Graph 3

yougatup @ POSCAT

Topic

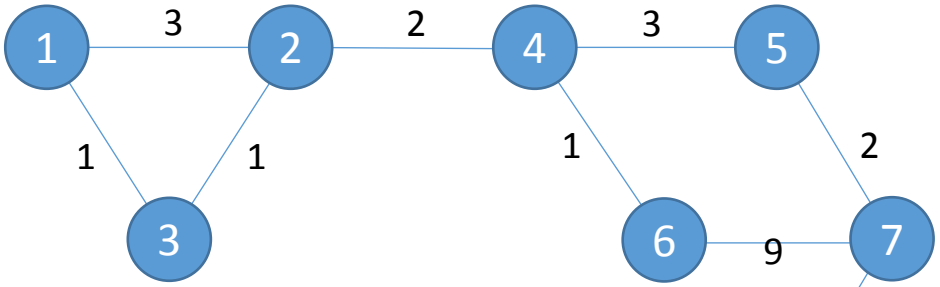
- Topic today
  - Minimum Spanning Tree
    - Cut Property
    - ~~Prim Algorithm~~
    - Kruskal Algorithm

Spanning Tree

- Definition

  A tree which "spans" whole vertices

  i.e. a tree with n vertices

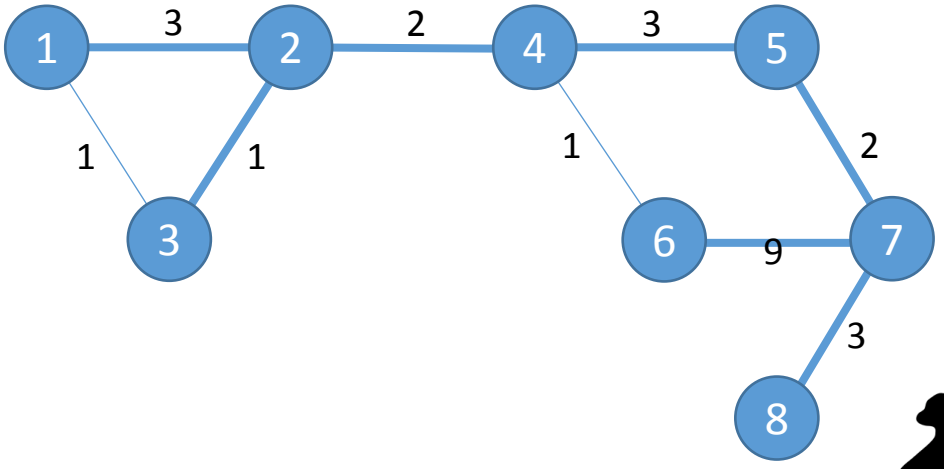


Spanning Tree

- Definition

  A tree which "spans" whole vertices

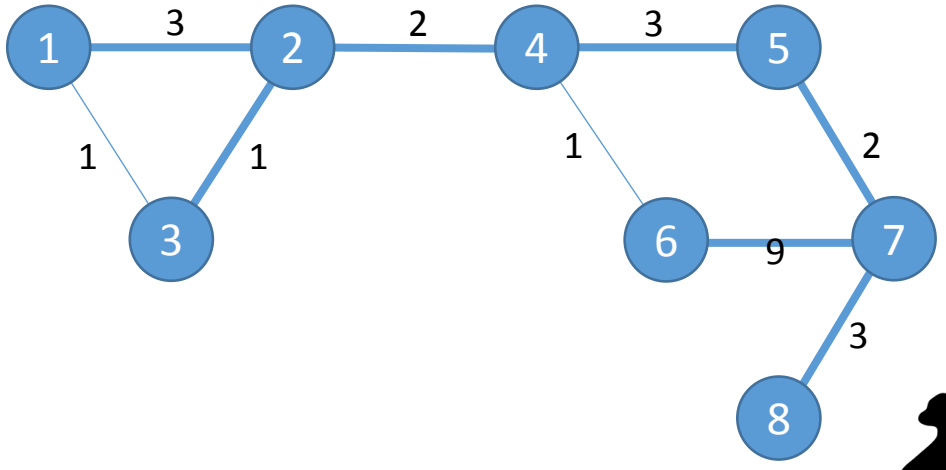
  i.e. a tree with n vertices



Minimum Spanning Tree

- Problem

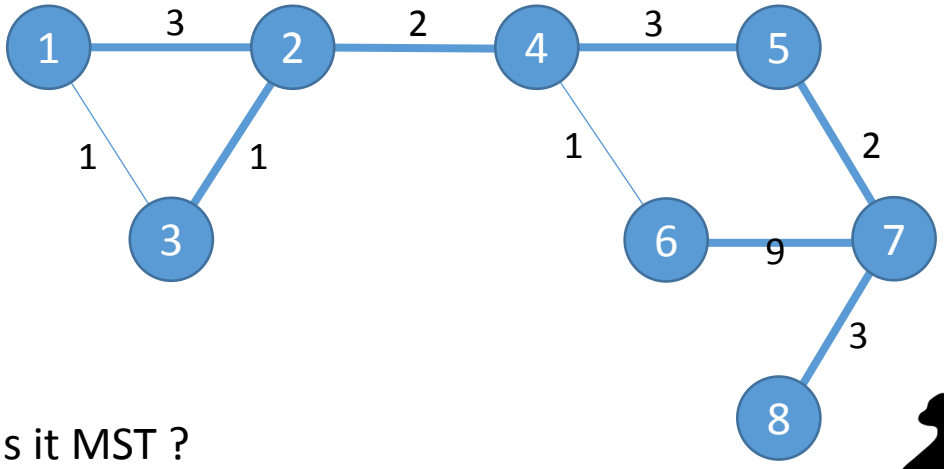
  Find a spanning tree with minimum cost



Minimum Spanning Tree

- Problem

  Find a spanning tree with minimum cost

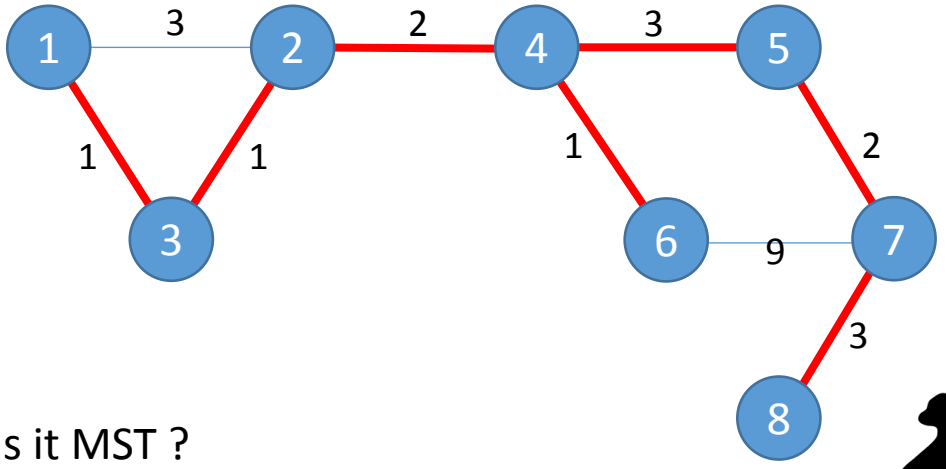


Is it MST ?

Minimum Spanning Tree

- Problem

  Find a spanning tree with minimum cost



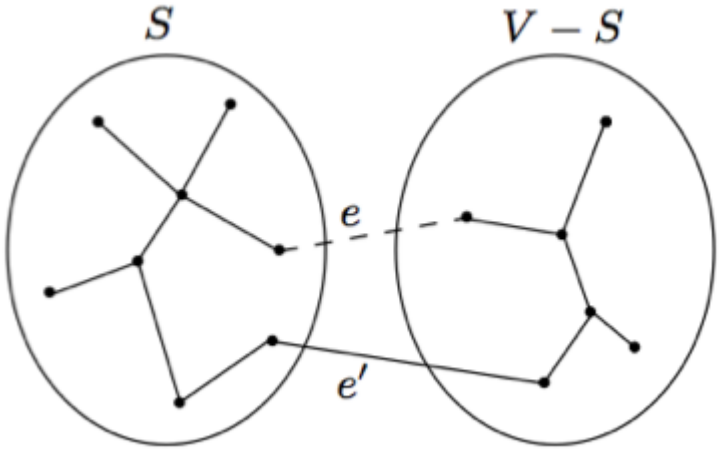
Is it MST ?

Cut Property

- Theorem

  Suppose edge X are part of a MST of $G = (V, E)$. Pick any subset of nodes S for which X does not cross between S and $V \backslash S$, and let e be the lightest edge across this partition. Then $X \cup \{e\}$ is part of some MST

Proof ?



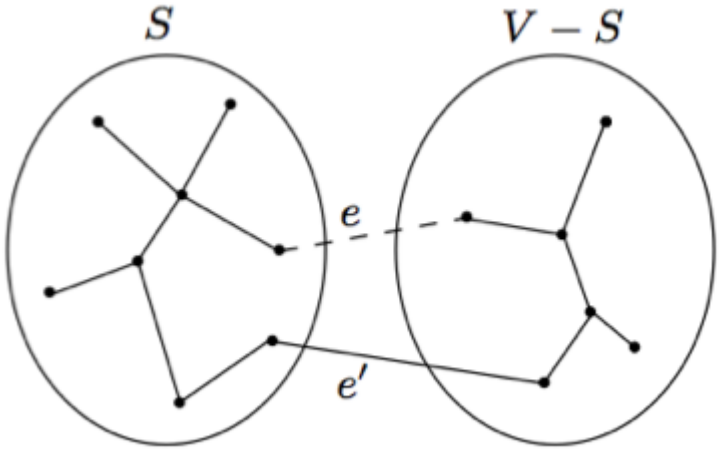
Cut Property

- Theorem

  Suppose X is a set of edges which are part of a MST of $G = (V, E)$. Pick any subset of nodes S for which X does not cross between S and $V \backslash S$, and let e be the lightest edge across this partition. Then $X \cup \{e\}$ is part of some MST

Assume $e \notin T$. Then we can construct a different MST T' containing $X \cup \{e\}$ by altering T slightly.

Compare the cost(T') and cost (T)

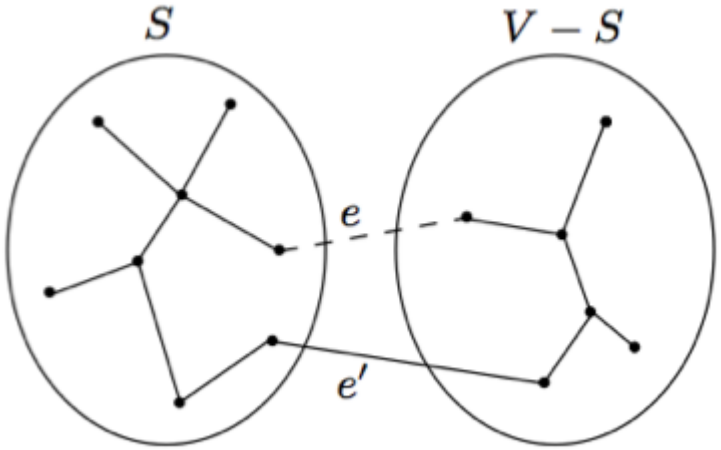


Cut Property

- Theorem

  Suppose X is a set of edges which are part of a MST of $G = (V, E)$. Pick any subset of nodes S for which X does not cross between S and $V \backslash S$, and let e be the lightest edge across this partition. Then $X \cup \{e\}$ is part of some MST

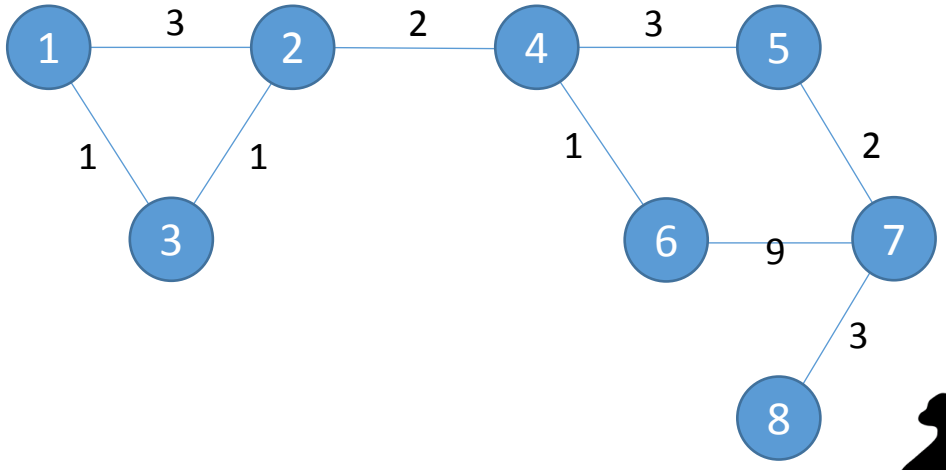
by cut property, we can derive beautiful greedy algorithm !



Kruskal Algorithm

- Approach

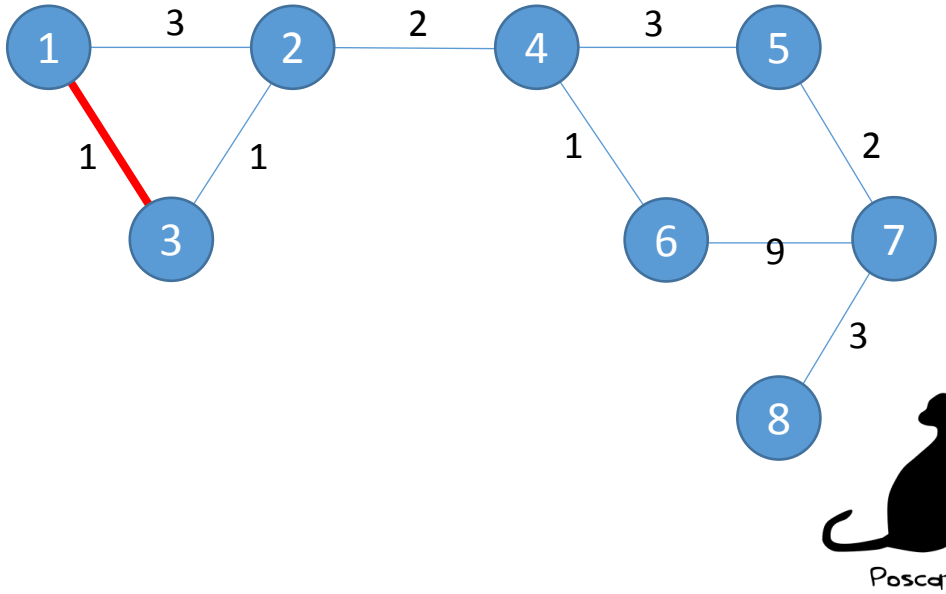
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

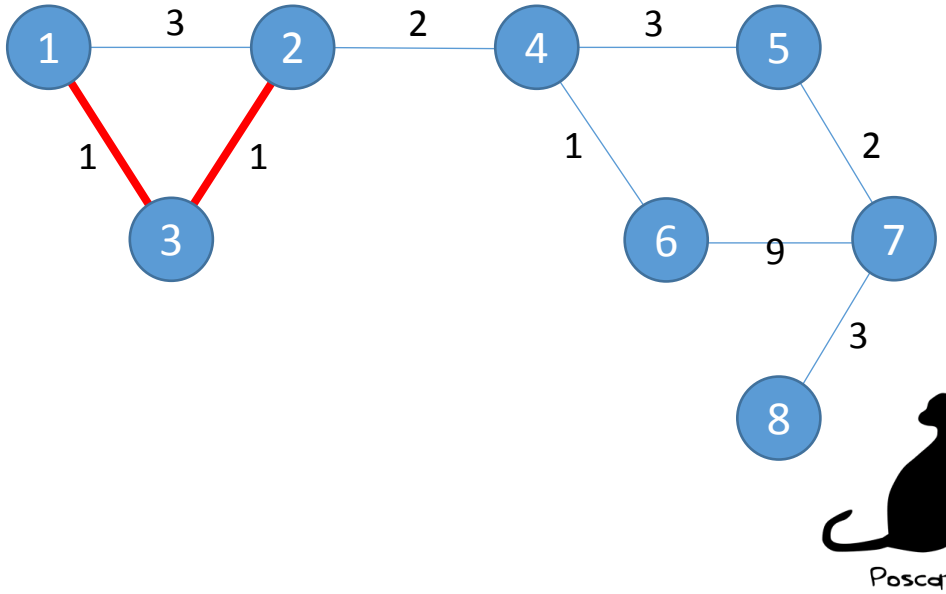
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

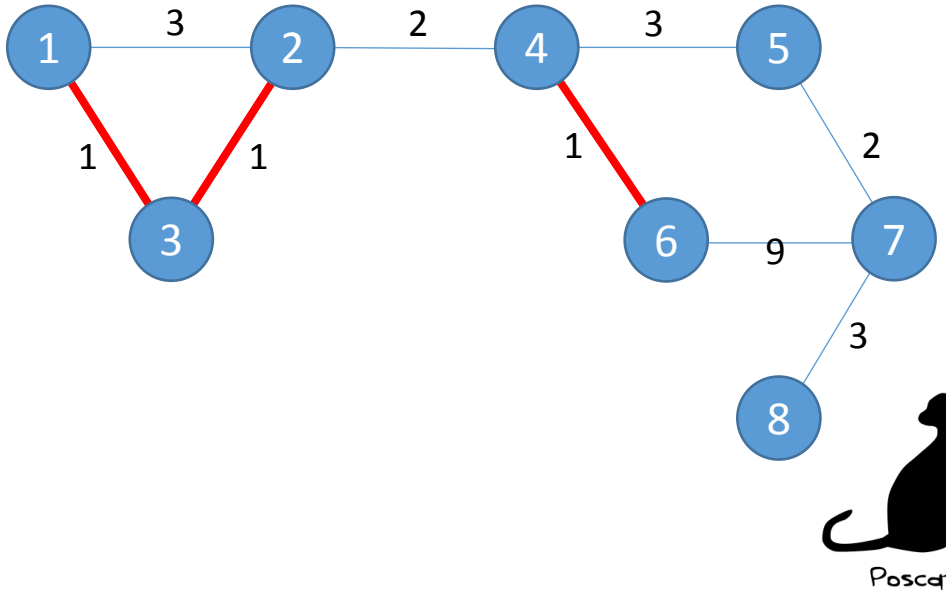
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

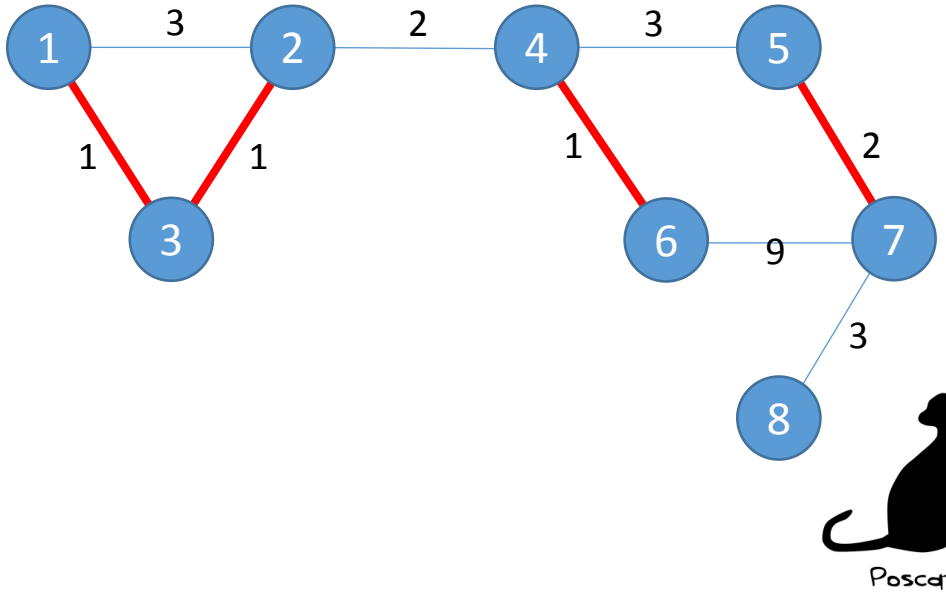
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

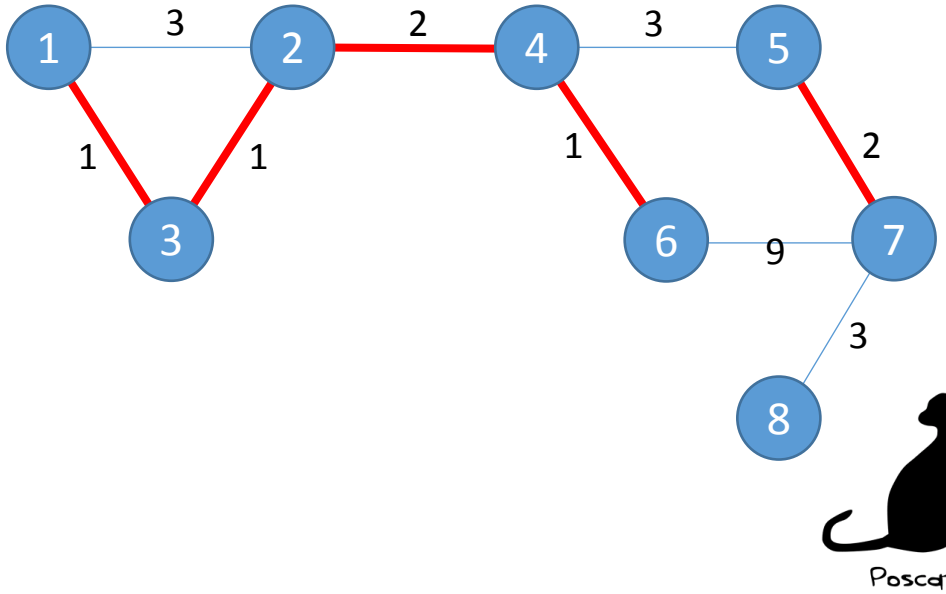
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

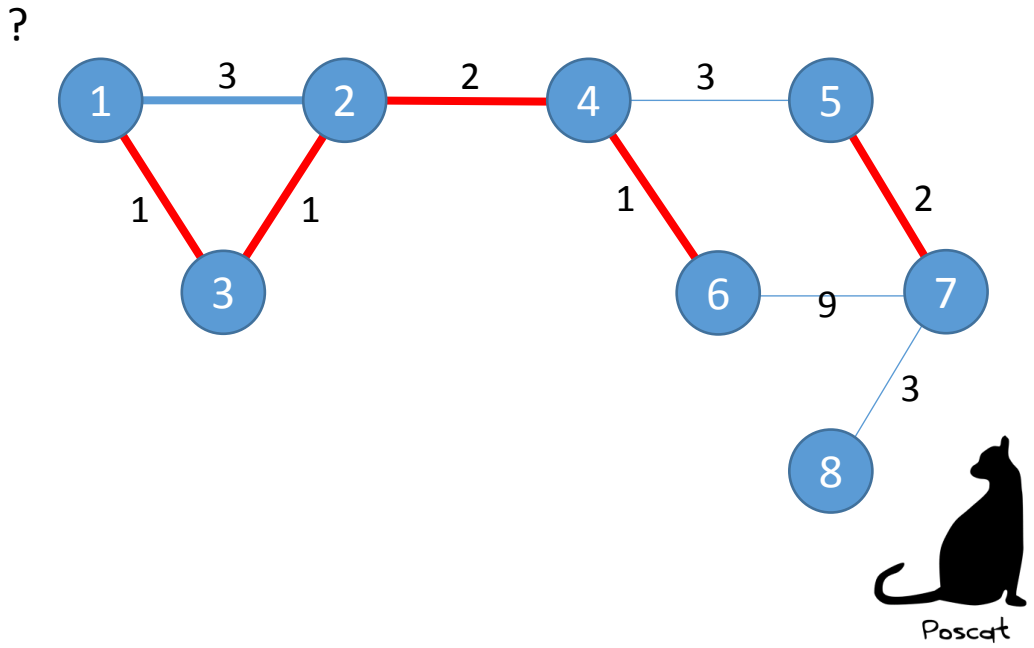
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

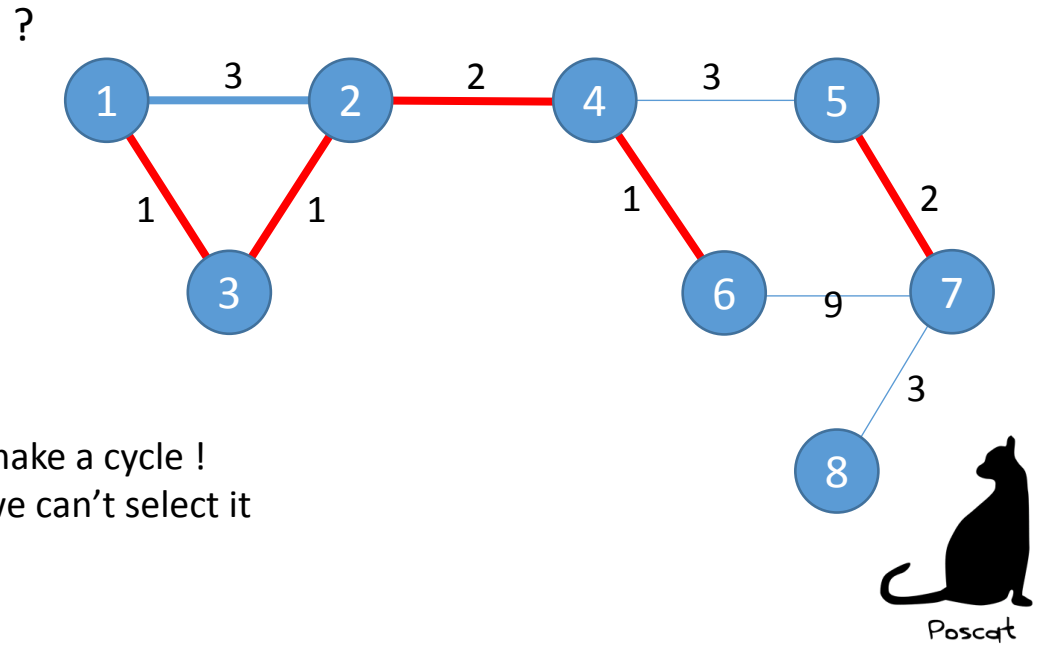
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

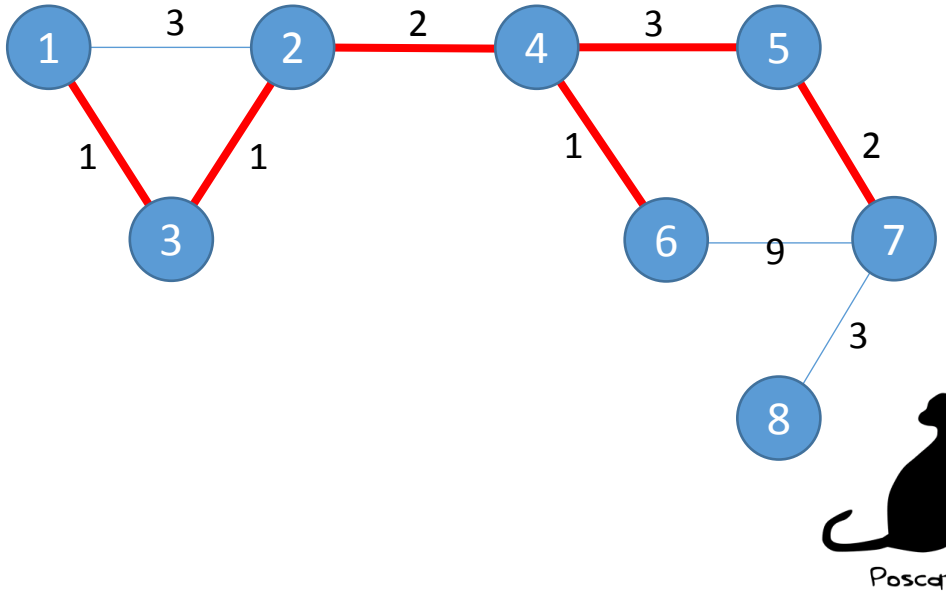
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

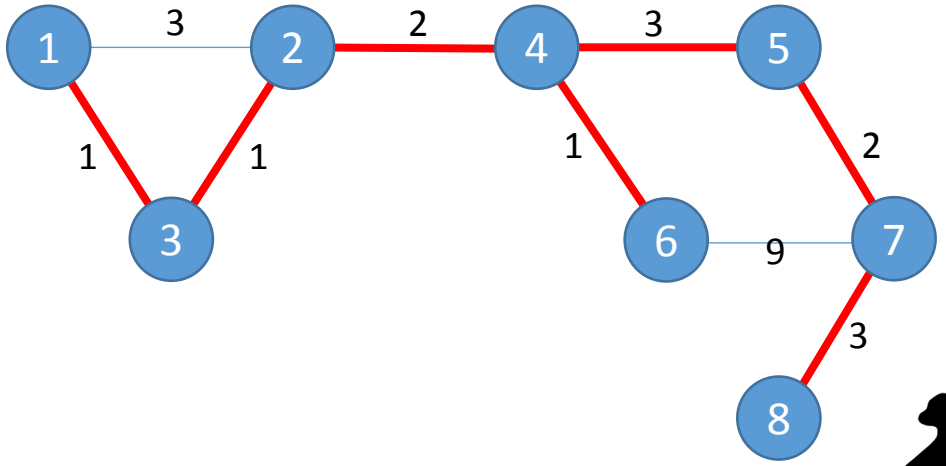
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

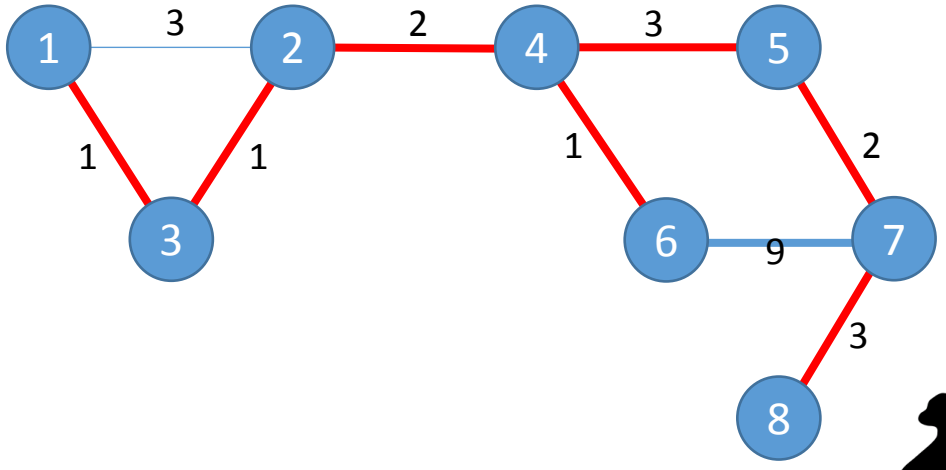
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

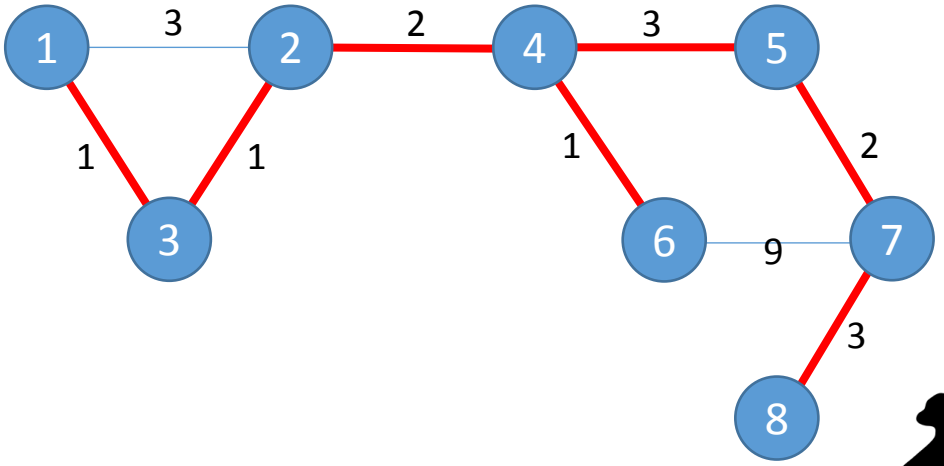
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

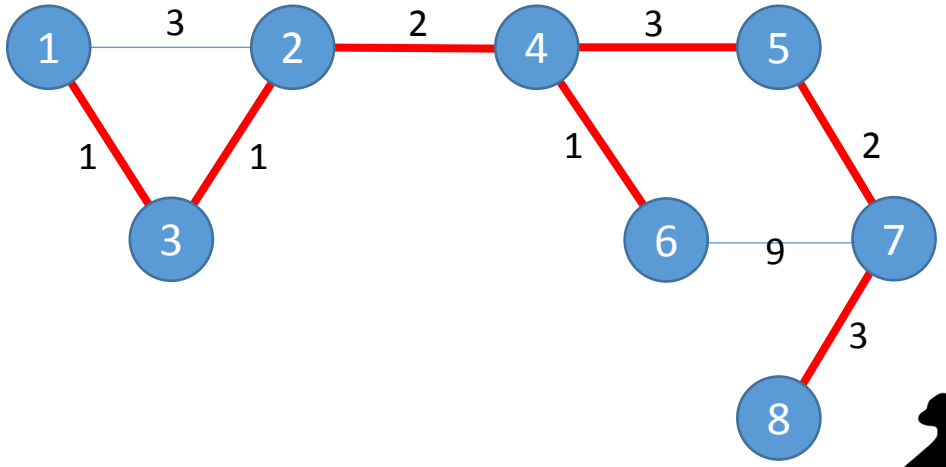
  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.



Kruskal Algorithm

- Approach

  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.

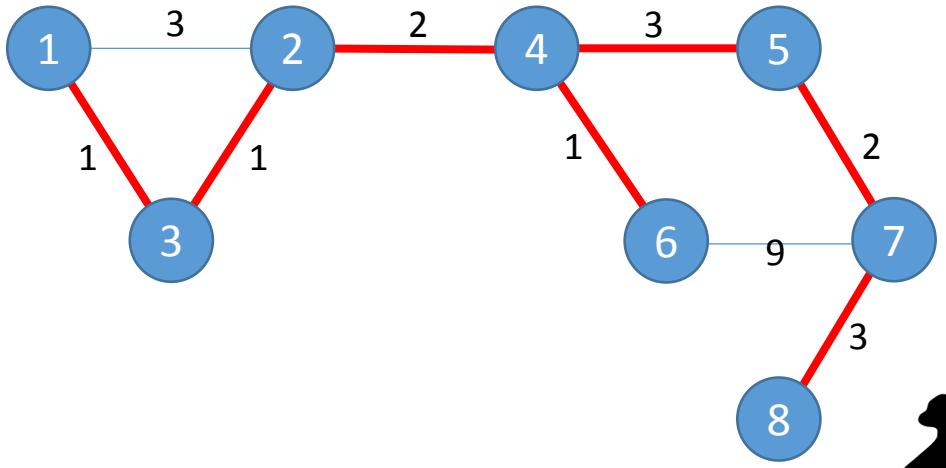


Done ! we have no remaining edges

Kruskal Algorithm

- Approach

  For each iteration, choose minimum edge which doesn't make a cycle. Then it will make a MST by cut property.

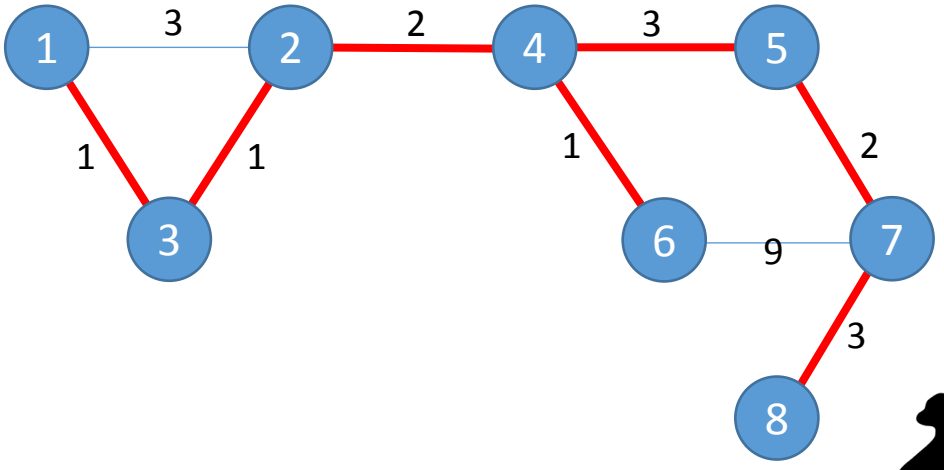


Done ! we have no remaining edges

Kruskal Algorithm

- Question

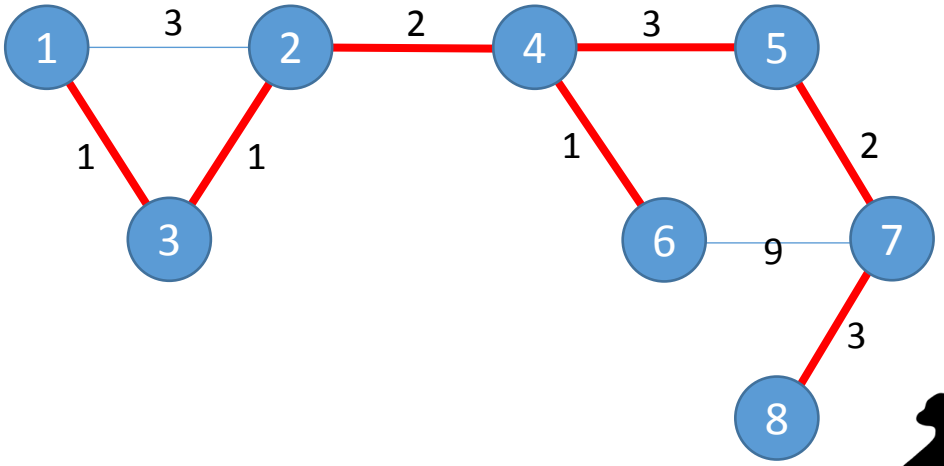
  How can we determine whether adding a edge makes a cycle or not ?



Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

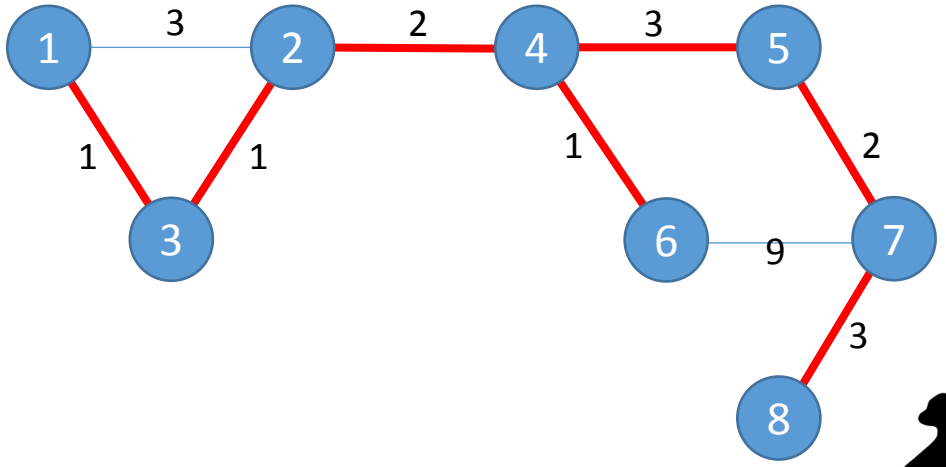


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

  If a edge connects two vertices with different group, it will never make a cycle.

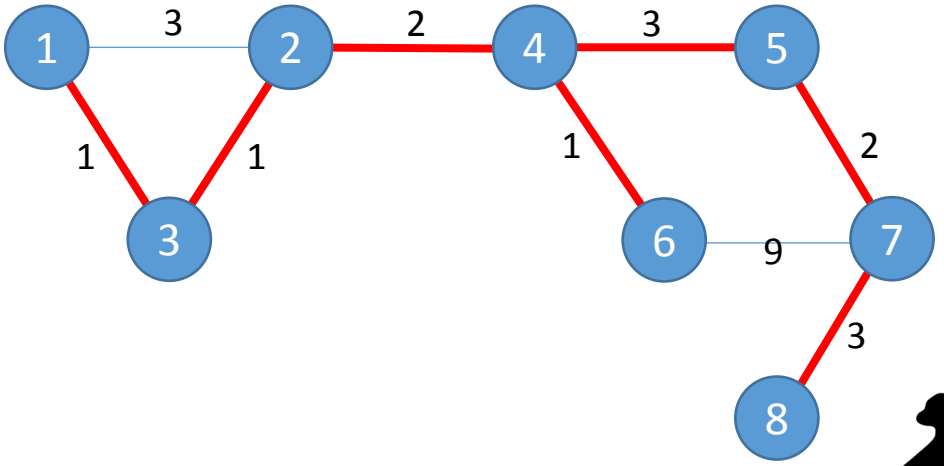


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

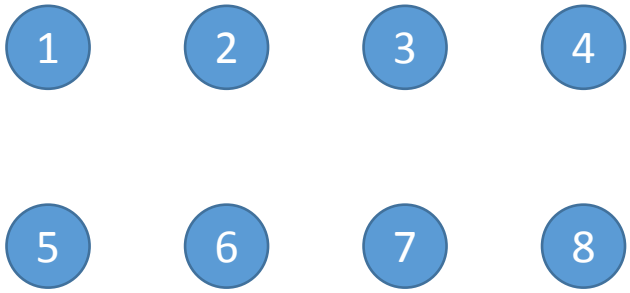
  If not, it will make a cycle !

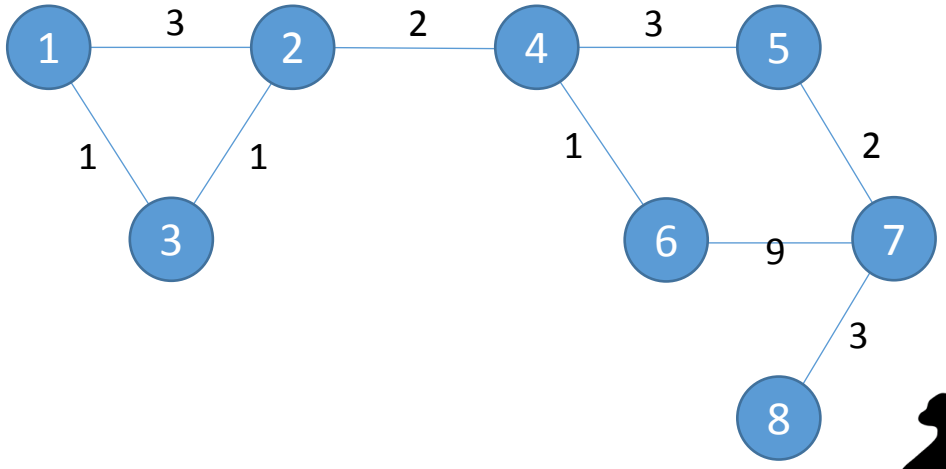


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

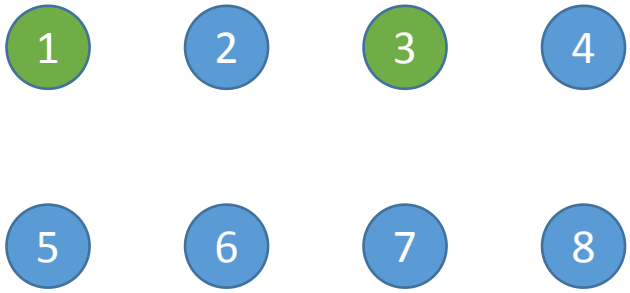


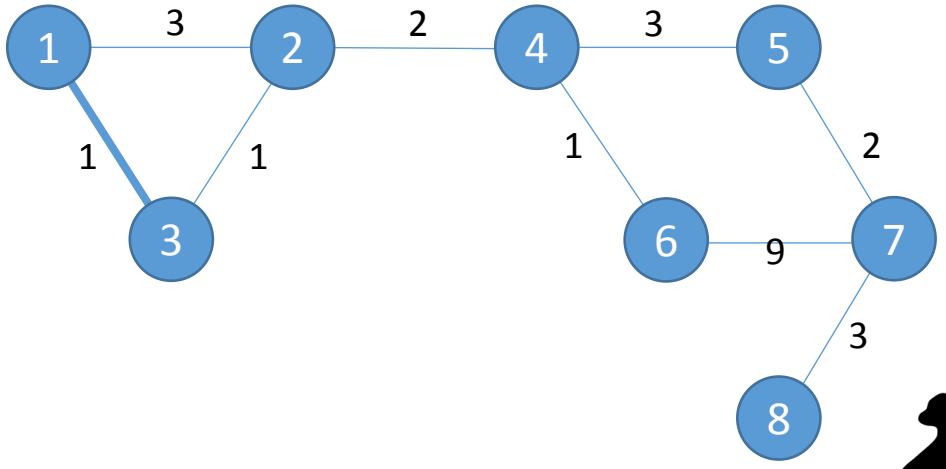


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

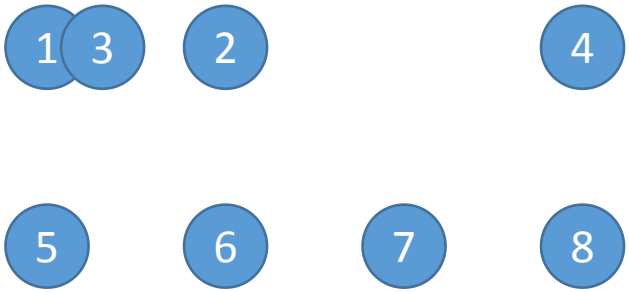


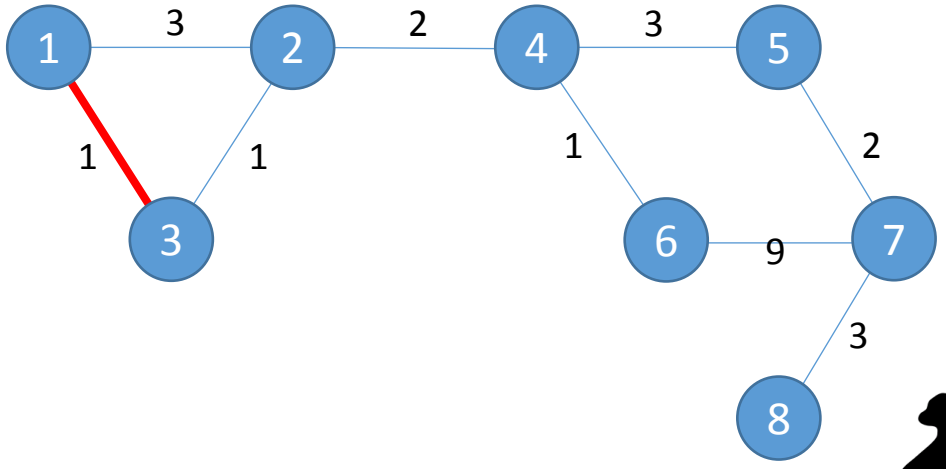


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

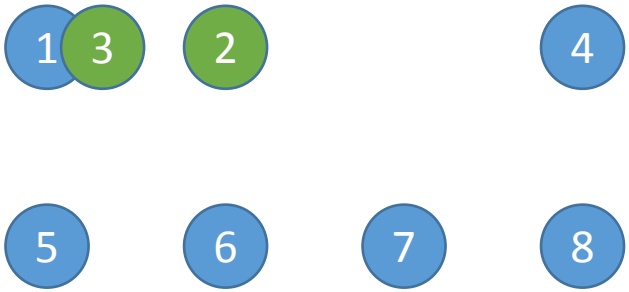


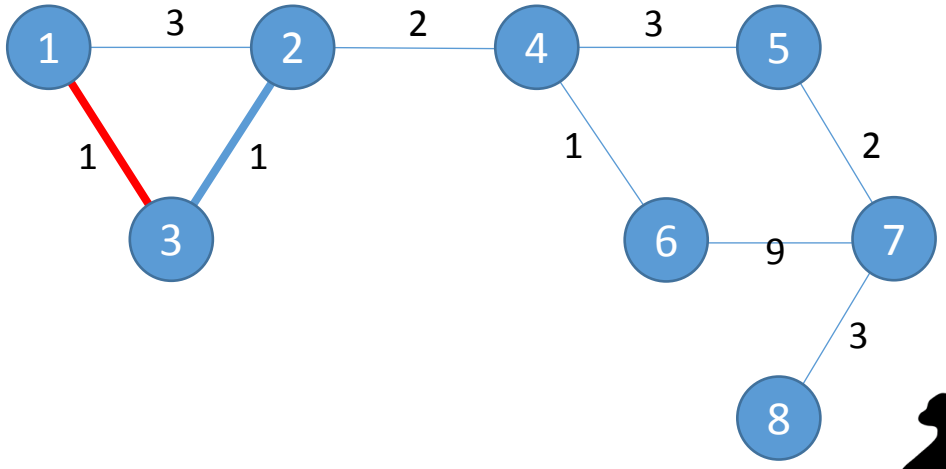


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

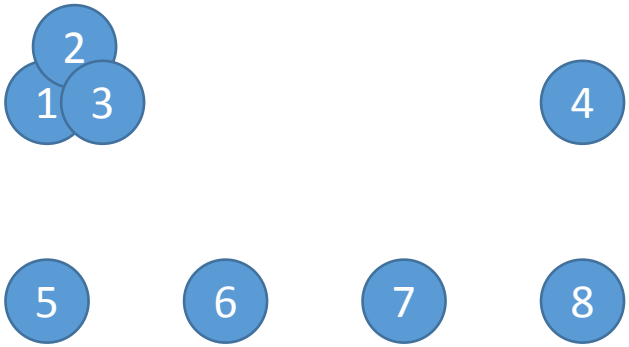


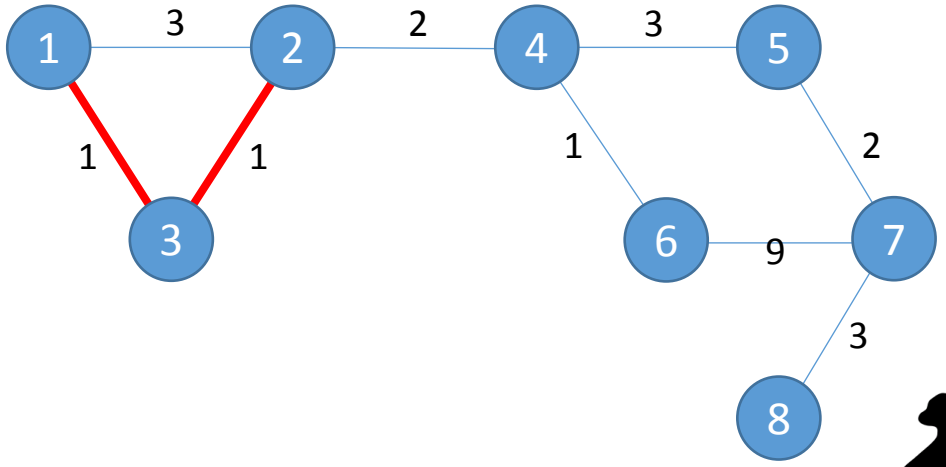


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

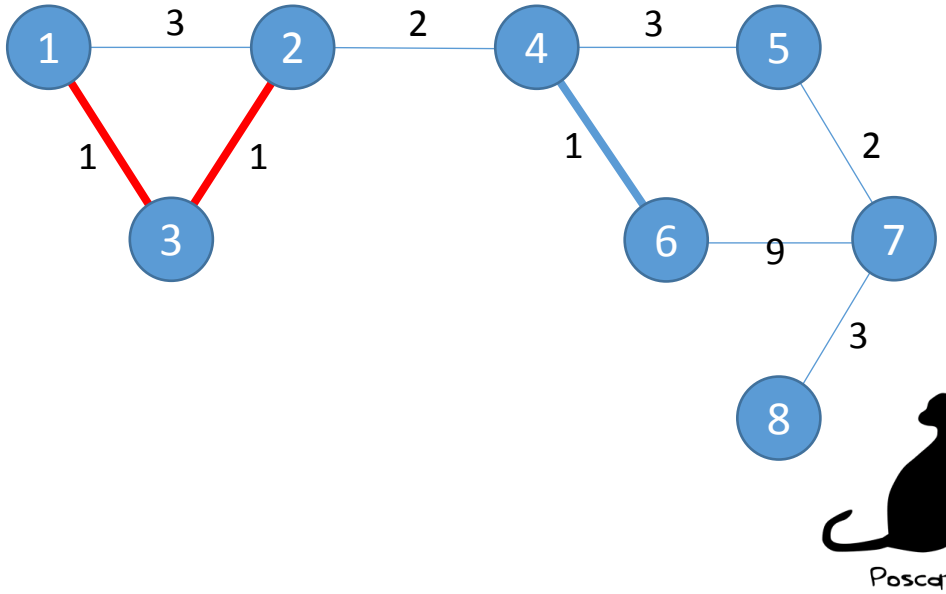




Kruskal Algorithm

- Question

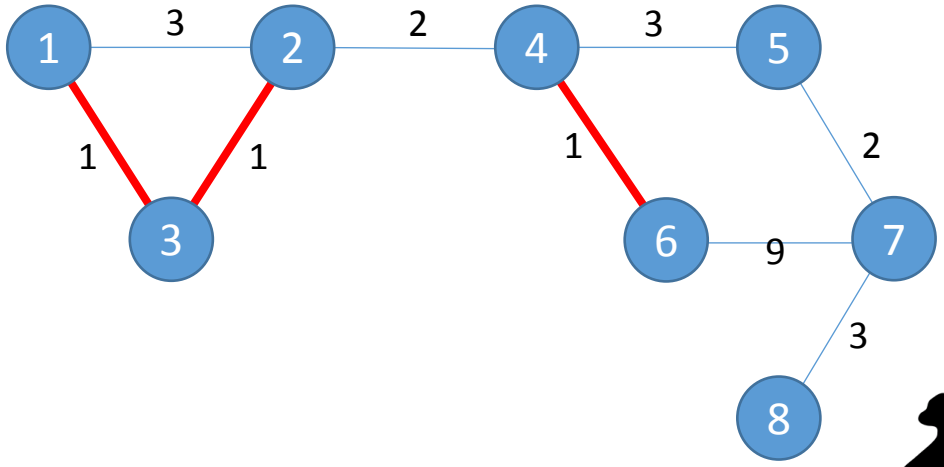
  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !



Kruskal Algorithm

- Question

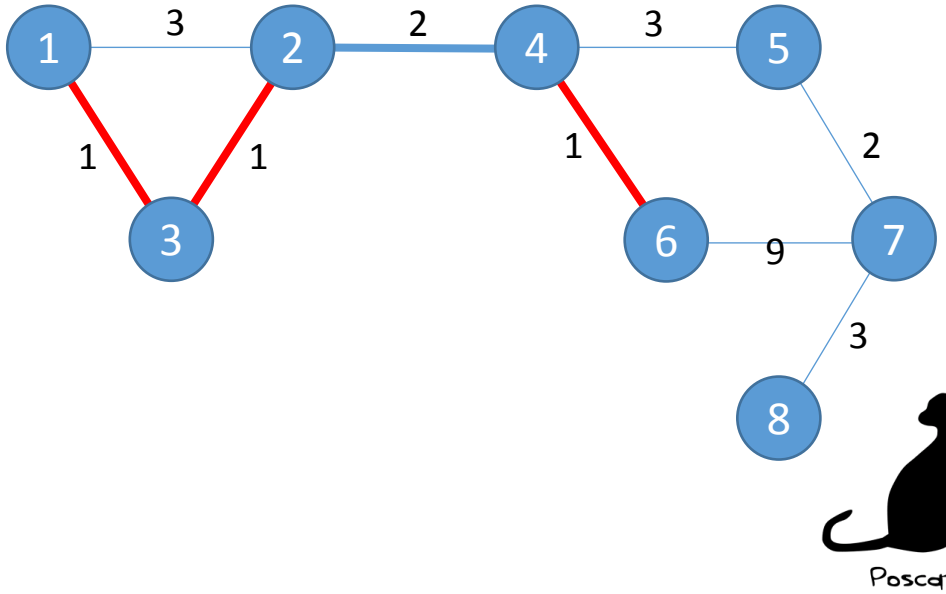
  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !



Kruskal Algorithm

- Question

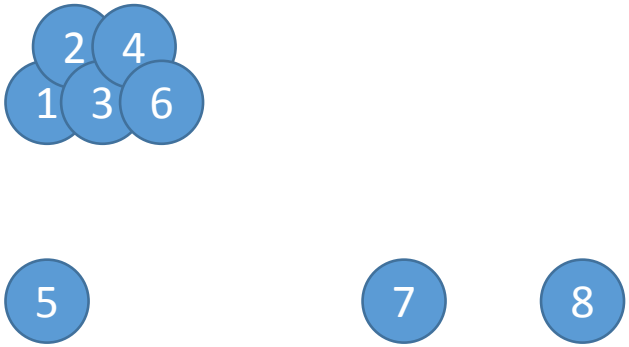
  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

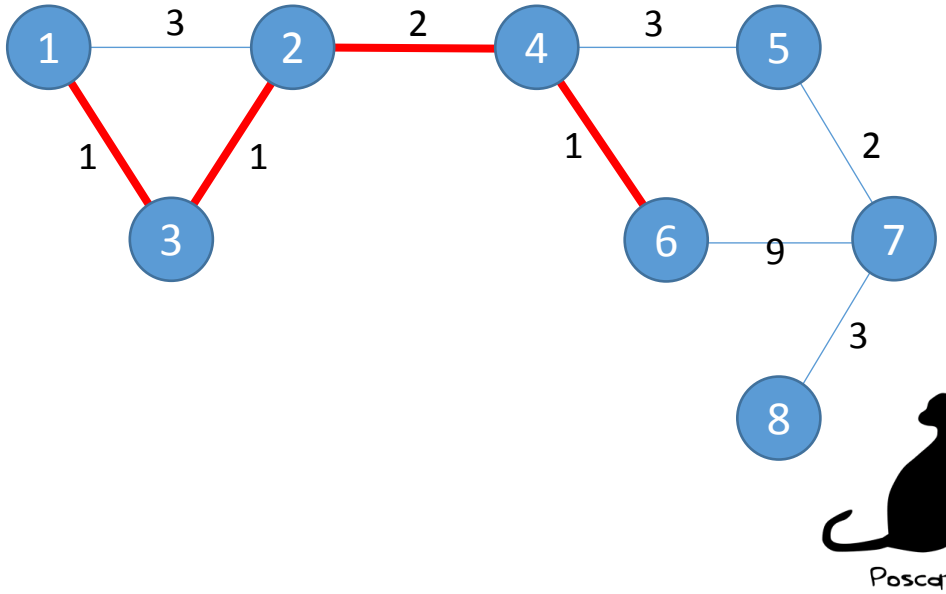


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

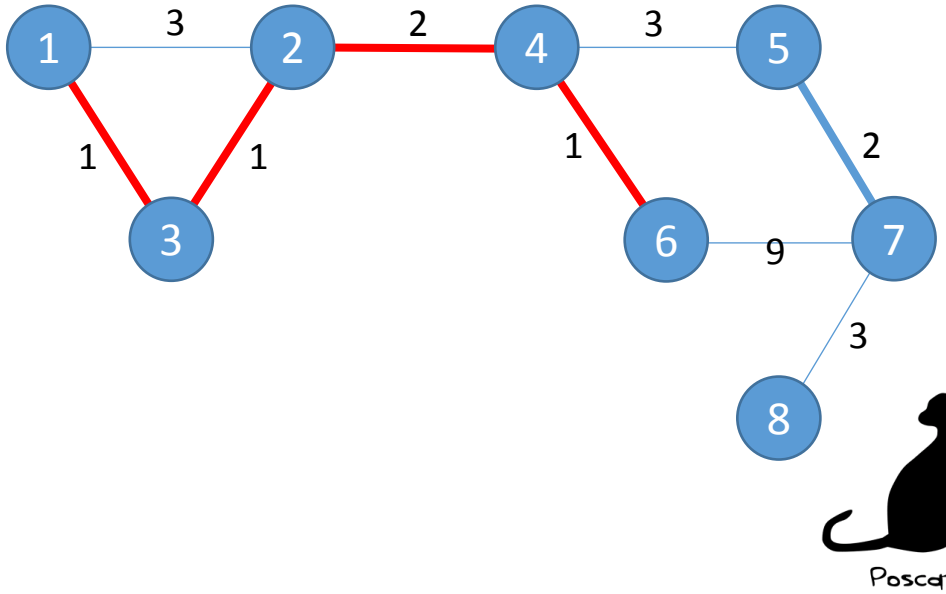




Kruskal Algorithm

- Question

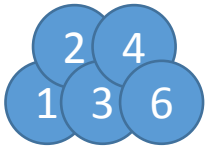
  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

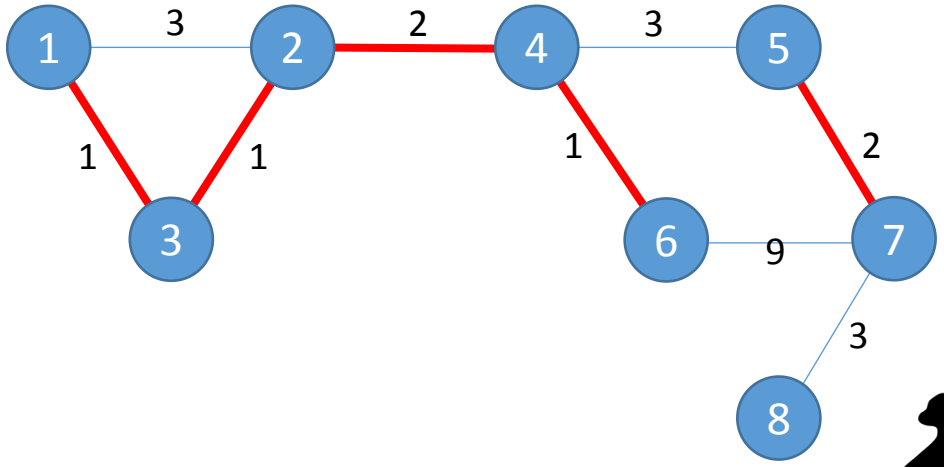


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !



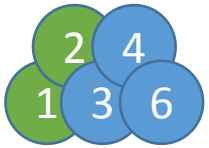


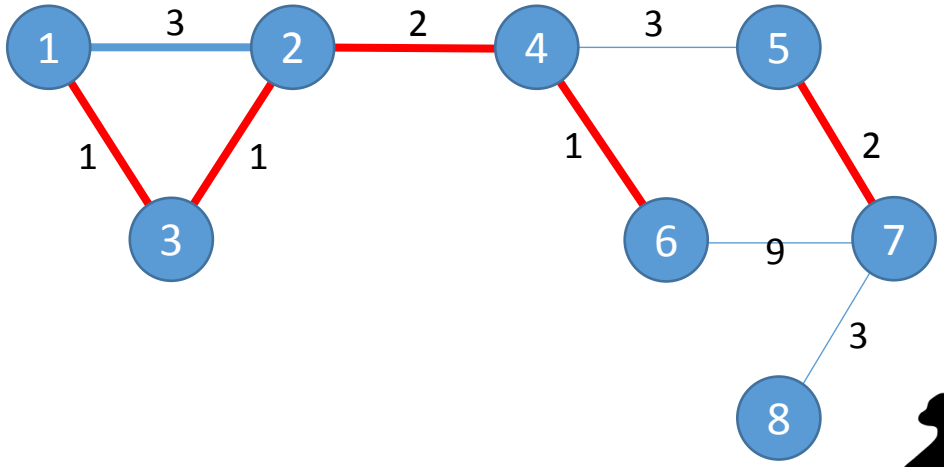
Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

No !!

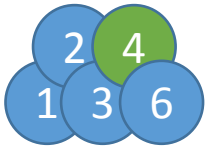


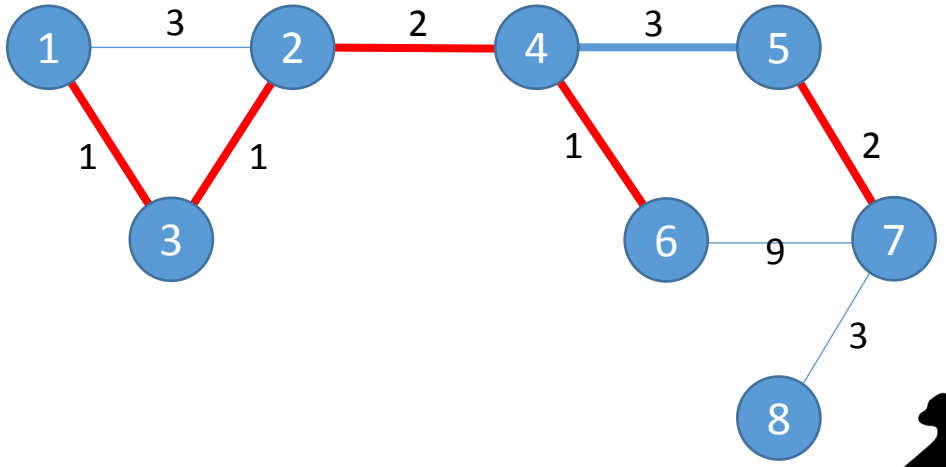


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

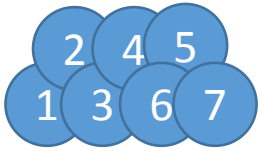


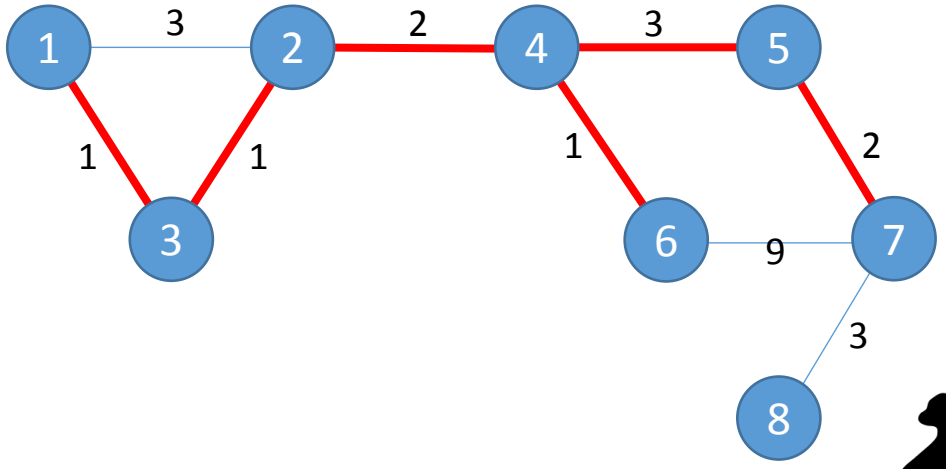


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

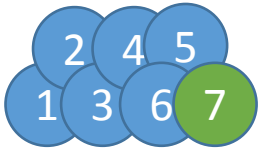


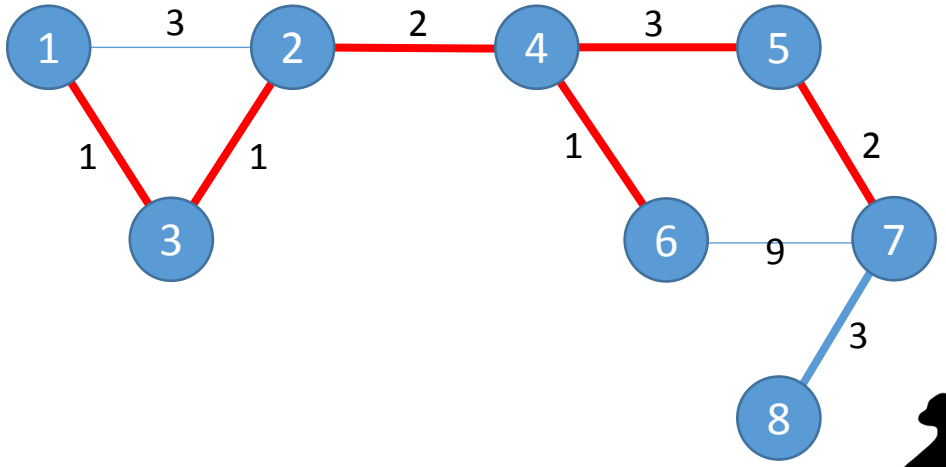


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

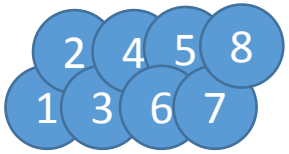


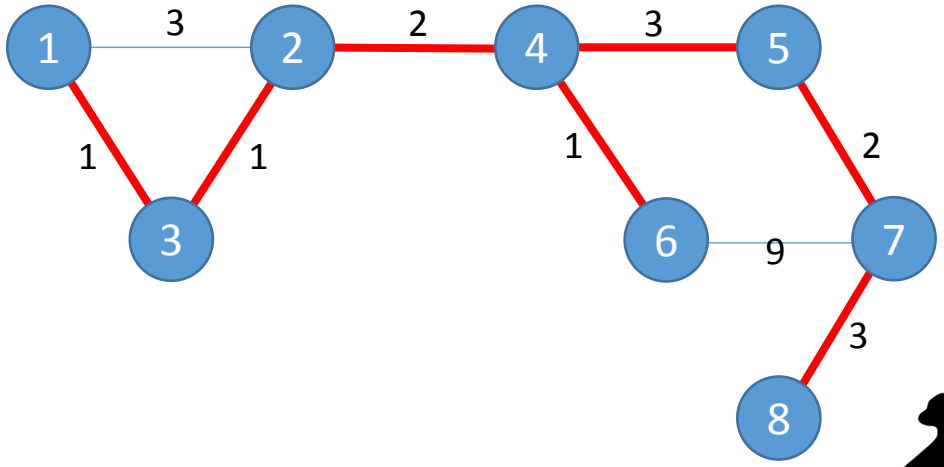


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

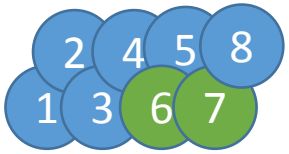


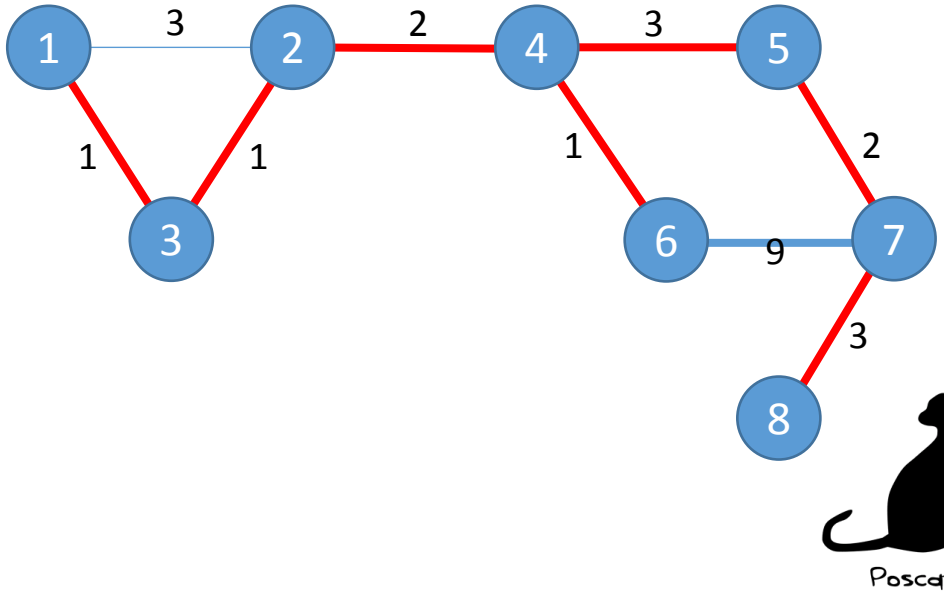


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !

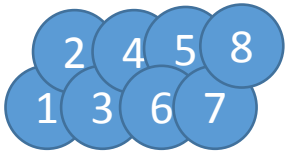


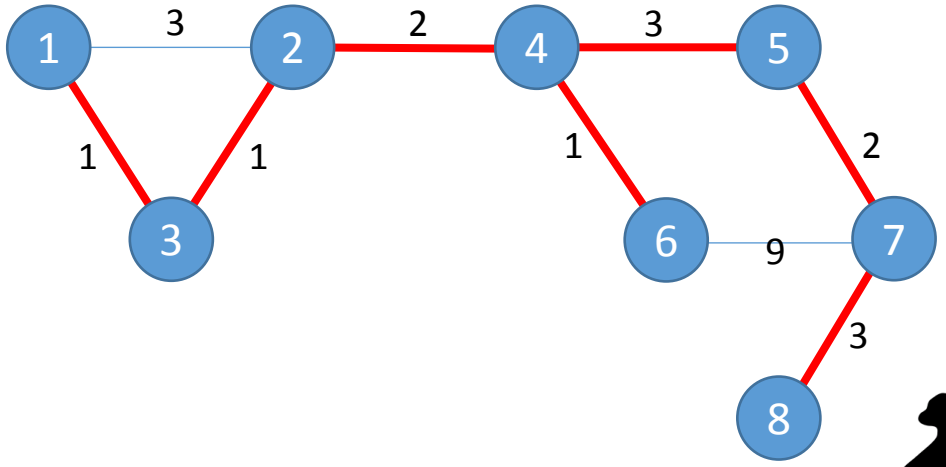


Kruskal Algorithm

- Question

  How can we determine whether adding a edge makes a cycle or not ? → by using Disjoint Set !





Kruskal Algorithm

- Implementation
  - We have to sort the set of edges via their costs → with STL
  - Union & Find

→ Quite simple !

Kruskal Algorithm

- Implementation
  - We have to sort the set of edges via their costs → with STL
  - Union & Find

→ Quite simple !

Analysis ?

Kruskal Algorithm

- Implementation
  - We have to sort the set of edges via their costs → with STL
  - Union & Find

→ Quite simple !

Analysis ? Sorting takes O($E \log E$)

Each iteration takes almost O(1)
because we use Union & Find

Kruskal Algorithm

- Implementation
  - We have to sort the set of edges via their costs → with STL
  - Union & Find

→ Quite simple !

Analysis ? Sorting takes O($E \log E$)

Each iteration takes almost O(1)
because we use Union & Find → **O($E \log E$)**