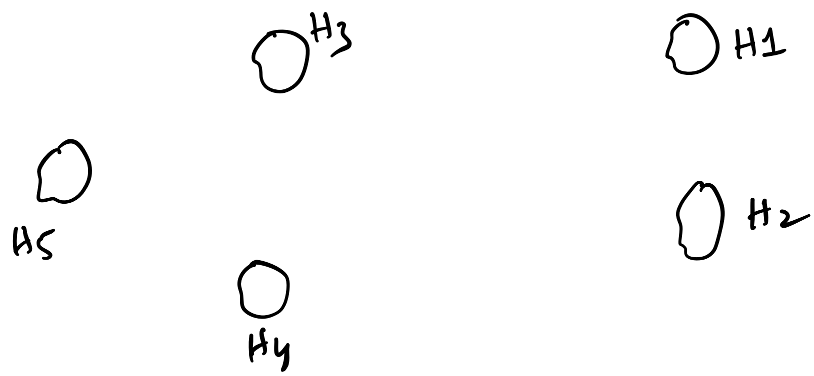
Minimum Spanning Trees [CLRS Chap: 23]

Network design :

Internet Connection Provider



You want to connect the houses with a minimum total cost.

Applications of MST :

1. Planning how to lay network cable to connect several locations to the internet.

2. Designing road networks etc.

In this topic

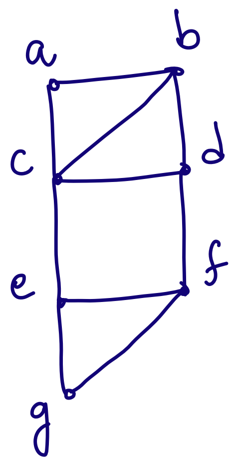
We work with

Undirected graphs

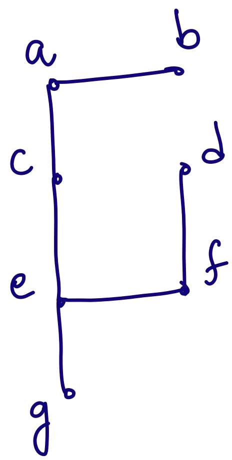
Recap: A Subgraph H of a graph G is a graph whose set of vertices and set of edges are all subsets of G.

- Spanning tree T of an undirected graph G is a subgraph that is a tree which includes all the vertices of G.

E.g :



Graph G



Spanning tree of G

True|False: Every connected graph contains a Spanning tree.

Weighted Graphs

- In many applications, each edge of a graph has an associated numerical value, called a weight.

- usually, the edge weights are non-negative integers.

- Weighted graphs may be either directed or undirected.

- The weight of an edge is often called as the Cost of the edge.

- In application, the weight of an edge may be a measure of the

  length of a route,

  the capacity of a line

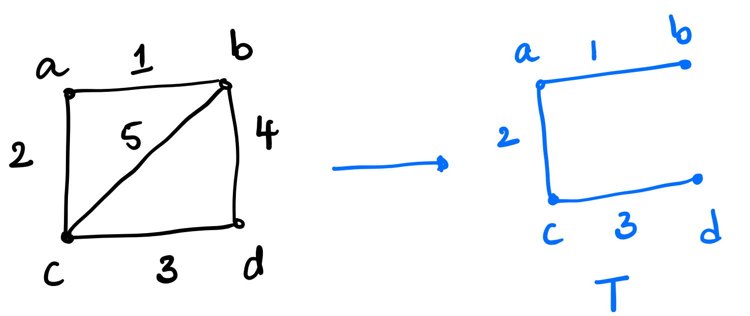
  etc.

Minimum Spanning Trees

Def: The minimum (weight) spanning tree (MST) Problem is given a graph $G = (V, E, w)$ with non-negative edge weights, find a spanning tree of minimum weight, where the weight of a tree T is defined as: $w : E(G) \to \mathbb{R}$

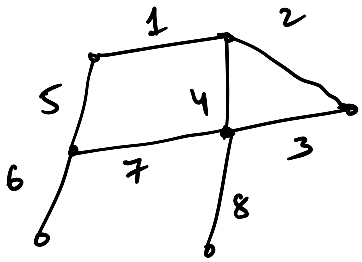
$$w(T) = \sum_{e \in T} w(e)$$

Example:-



$w(T) = 1 + 2 + 3 = 6$

Example ②:



Graph G.

Q: What is the minimum spanning tree of G?

What is its weight?

MST Problem:

Input: A weighted connected graph G (Undirected)

Output: A MST of G.

I/P: G

O/P: MST of G

Any Warm-up Ideas ?

True/False:

There can be more than one MST for a graph.

True/False: If T is a MST of G, then Every minimum weight edge is in T.

True/False: Suppose All edge weights of G are distinct. If T is a MST of G, then the minimum weight edge is in T.

We look at two algorithms for MST Problem

- Kruskal's algorithm
- Prim's algorithm

The two algorithms are greedy algorithms.

Pseudo code **Kruskal's algorithm**

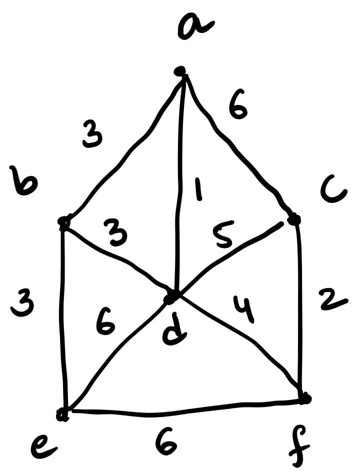
```
MST-KRUSKAL(G, w)
    A = ∅
    for each vertex v ∈ G.V
        MAKE-SET(v)
    sort the edges of G.E into nondecreasing order by weight w
    for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
        if FIND-SET(u) ≠ FIND-SET(v)
            A = A ∪ {(u, v)}
            UNION(u, v)
    return A
```

Creates a set whose only member is v.

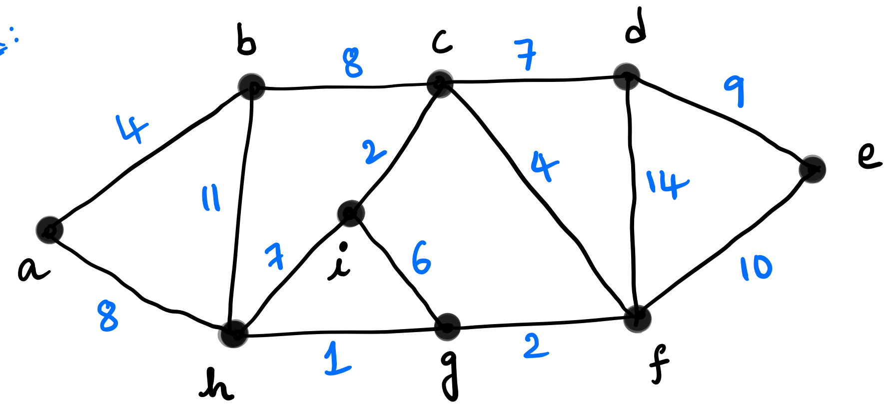
returns the representative element of the set containing v.

Unites the sets that contain u & v.

Ex 1



Ex2:



Q: Find the minimum spanning tree of G using Kruskal's algorithm. What is its weight?

Runtime analysis

Creates a set whose only member is v.

```
MST-KRUSKAL(G, w)
A = Ø
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

returns the representative element of the set containing v.

Unites the sets that contain u & v.

line 1 - $O(1)$

lines 2&3 - $O(V)$

line 4 - $O(E \log E)$

lines 5 to 8 - line 5 runs $O(E)$ times

we can check FIND-SET (u) $\neq$ FIND-SE (v) using either BFS or DFS (But this not optimal)

UNION (u, v) : merging two connected components

$O(V+E)$

Running time

$O(E(V+E))$

↓

Can be improved to

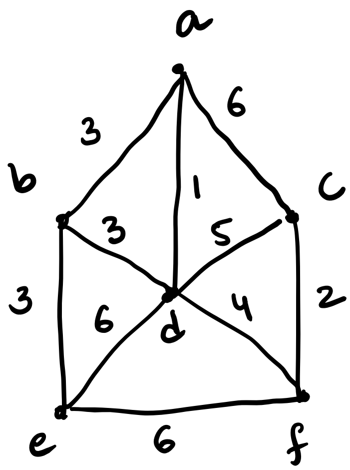
$O(E \log V)$

Prim's algorithm

- All the Selected edges always form a Single tree.

- The Algorithm starts from an arbitrary vertex r and grows until the tree spans all the vertices of the graph.

Ex 1



r = root vertex

$u.key$ = minimum weight of any edge connecting v to a vertex in the tree.

$u.\pi$ = Parent of v in the tree.

Q = min Priority queue

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

The MST A for G is

$$A = \{ (v, v.\pi) : v \in V - \{r\} \}$$

Running time

MST-PRIM(G, w, r)

```
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

$O(|V|)$

$O(|V|)$

$O(\log |V|)$

$O(|E|)$

decrease key operation

$O(\log |V|)$

$$= O(|V|) + O(|V| \log |V| + |E| . \log |V|)$$

$$= O(|E| \log |V|)$$

Can be improved to $O(|E| + |V| \log |V|)$ using Fibonacci heaps.

Exercise Problems

① Suppose we are given both an undirected graph G with weighted edges and a MST T of G.

ⓐ Describe an algorithm to update the MST when the weight of a single edge e is decreased.

ⓑ Describe an algorithm to update the MST when the weight of a single edge e is increased.

In both the cases, the input to your algorithm is the edge e and its new weight.

Your algorithm should modify T so that it is still a MST.

(2) The second smallest spanning tree of a given graph is the spanning tree of G with smallest total weight except for the MST.

Describe an algorithm to find the second smallest spanning tree of a given graph G.