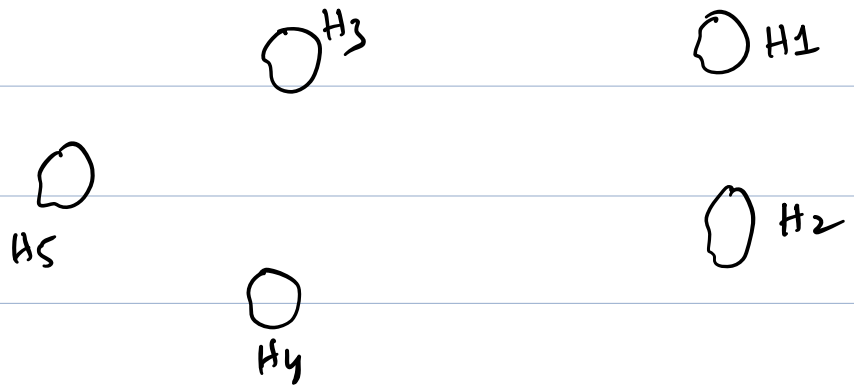
Minimum Spanning Trees [CLRS: Chapter 23]

Motivation

Network design:

Internet Connection Provider



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

Applications of MST:

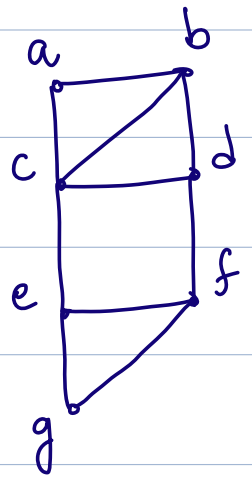
① Planning how to lay network cable to connect several locations to the internet.

② Designing road networks etc.

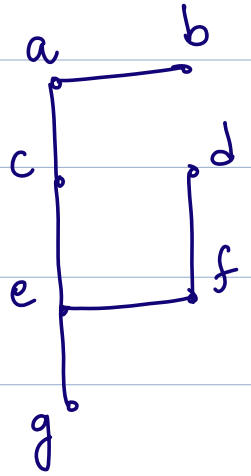
Recap: A **Subgraph** H of a graph G is a graph whose set of vertices and set of edges are all subsets of G.

– **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



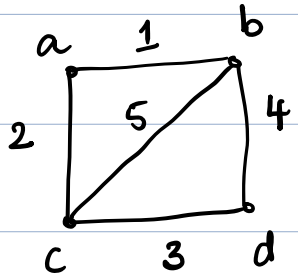
Spanning tree of G

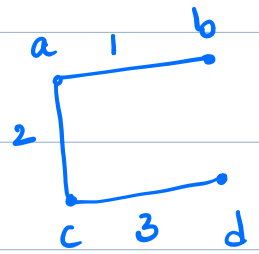
Minimum Spanning Trees

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: $w : E(G) \longrightarrow \mathbb{R}$

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

Example①:

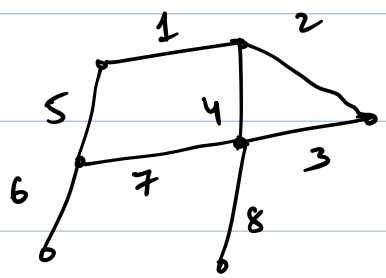


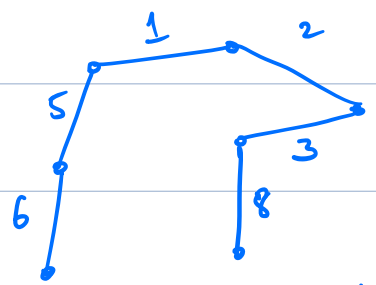


T

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

Example②:





$w(T) = 1 + 2 + 3 + 5 + 6 + 8$

$= 25$

Lemma: Every connected graph contains a spanning tree.

Proof: Iteratively deleting edges from cycles in a connected graph yields a connected acyclic subgraph

Warmup Questions

(Q1) Does a minimum weight edge be a Part of Some MST ?

Ans: YES.

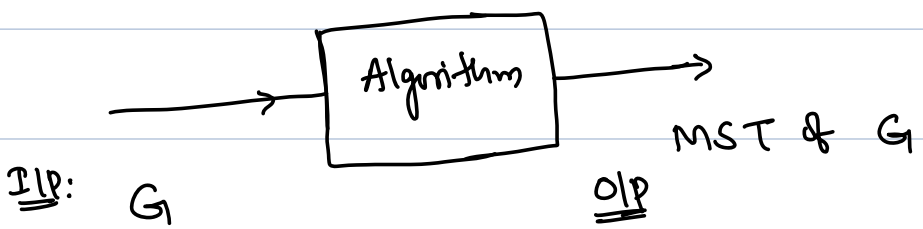
(Q2) [True/False]: Let e be a maximum-weight edge in a connected graph G, then e is not included in any MST.

Ans:- False, when G itself a tree.

MST Problem:

Input: A weighted connected graph G,

Output: A MST of G.



Greedy strategy is captured by the following generic method, which grows the MST one edge at a time.

We maintain a set of edges A, such that "Prior to each iteration, A is a subset of some MST." —(I)

At each step, we determine an edge (u,v) that we can add to A without violating above property (I), in the sense that $A \cup \{(u,v)\}$ is also subset of a MST. We call such an edge a safe edge for A.

```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3      find an edge (u, v) that is safe for A
4      A = A ∪ {(u, v)}
5  return A
```

Q How to find a safe-edge in line 3?

Some definitions

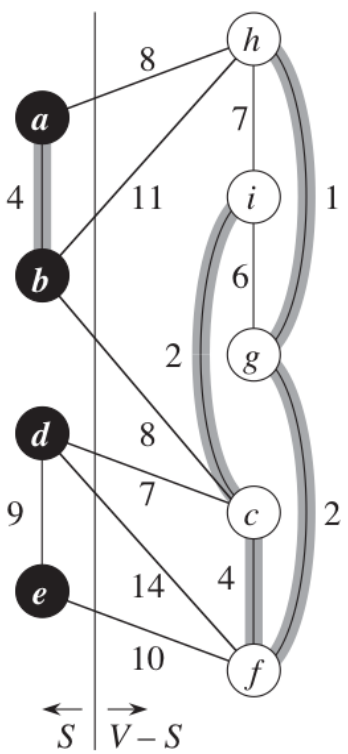
- A **cut** $(S, V-S)$ of an undirected graph $G = (V, E)$ is a partition of V.

- We say that an edge $uv \in E$ **crosses** the cut $(S, V-S)$ if one of its endpoints is in S and the other is in $V-S$.

- We say that a cut **respects** a set A of edges if no edge in A crosses the cut.

- An edge is a **light edge** crossing a cut if its weight is the minimum of any edge crossing the cut.

Example:



Theorem:

Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V - S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V - S)$. Then, edge (u, v) is safe for A.

Prof: let T be a MST that includes A.

Assume that T does not contain the edge uv

otherwise we are done.

We construct a MST T' that contains

$A \cup \{uv\}$, implies uv is safe for A.

Consider the path P from u to v in T.

then $P \cup \{uv\}$ forms a cycle.

Since u & v are on opposite sides of the

cut (S, V-S) at least one edge in T

also crosses the cut. let xy be any such edge.

The edge $xy \notin A$, because the cut respects A.

Removing xy disconnects T into two components. Adding the edge uv reconnects them to form a new spanning tree $T' = (T - \{xy\}) \cup \{uv\}$

Claim: T' is MST

uv is the light edge crossing the cut & xy is also crossing the cut

$$w(uv) \leq w(xy)$$

ie, $$w(T') = w(T) - w(xy) + w(uv)$$

$$w(T') \leq w(T) \quad \text{—①}$$

But T is MST, so $w(T) \leq w(T')$ — (2)

Thus $w(T) = w(T')$

Thus T' is MST.

Also $A \subseteq T$ and $xy \notin A$

$\therefore A \cup \{uv\} \subseteq T'$

ie, T' is MST & uv is safe for A.

Kruskal's Algorithm

Input - A weighted connected graph G, $w : E(G) \to \mathbb{R}$

Output - A MST of G.

Step 1: order the edges of G in non-decreasing order (say σ). Let T_0 be the graph with vertex set $V(G)$ & no edges.

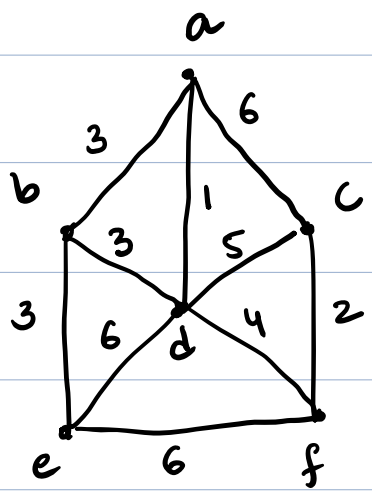
Step 2: Pick the first edge from σ and add it to T_0. Call the resulting graph as T_1

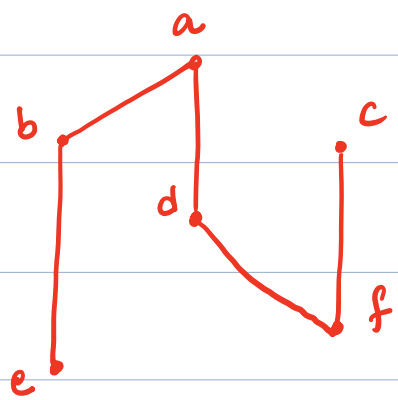
Step 3: (Recursive Step)

After selecting i edges $e_1, e_2, \ldots e_i$ and forming a graph T_i, find the next edge say e_{i+1} in the ordering σ after e_i, which does not create a cycle after adding to T_i

Step 4: STOP when $n-1$ edges are selected and output $T = T_{n-1}$ as MST of G.

Ex 1



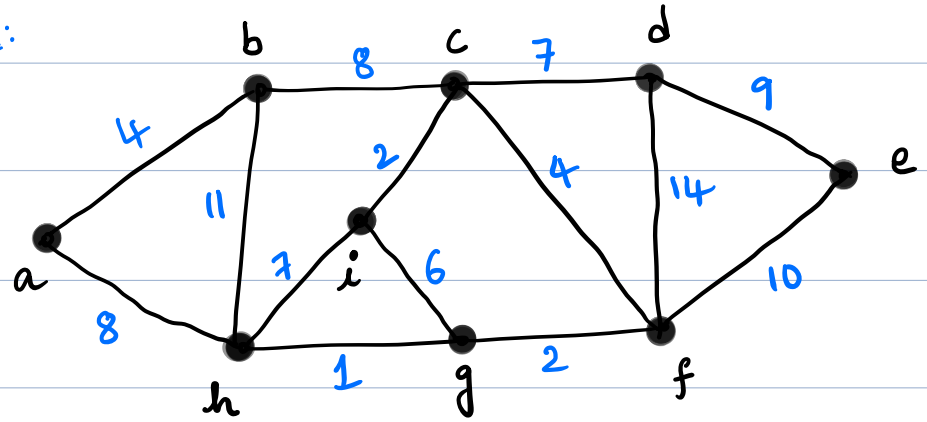


MST obtained from Kruskal's Algorithm.

The order in which edges are picked is

ad, cf, be, ab, fd

Ex2:



Pseudo code

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  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.

Running time: $O(E(V+E))$

By using better data structures we can show that running time is $O(E \log V)$

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) ≠ FIND-SET(v) using either BFS or DFS (But this not optimal)

UNION(u, v): merging two connected components $O(V+E)$

Proof of Correctness : ie,

The tree T constructed by kruskal's algorithm is a MST of G.

Proof:- First it is easy to see that T is indeed a tree (from step 3 of the algorithm)

Let T' be a MST of G.

if $E(T) = E(T')$ then we are done.

Suppose $E(T) \neq E(T')$ and e_{i+1} be the first edge in T which is not T'.

so $E(T) = \{e_1, e_2, \ldots e_i, e_{i+1} \ldots e_{n-1}\}$ &

$E(T') = \{e_1, e_2 \quad e_i, f_{i+1}, \ldots f_{n-1}\}$

Consider the graph $T' \cup \{e_{i+1}\}$

It contains a unique cycle C,

Since $E(C) \not\subseteq E(T)$, there exists an edge $f \in E(C)$ which is $E(T') - E(T)$

Consider the tree $T' + e_{i+1} - f$

$w(T') \leq w(T' + e_{i+1} - f)$ (because T' is MST)

$= w(T') + w(e_{i+1}) - w(f)$

$w(f) \leq w(e_{i+1})$ — (A)

At the same time, f doesn't create an cycle with the edges $e_1, e_2, \ldots e_i$, Since $e_1, e_2, \ldots e_i \in T'$

But we have selected e_{i+1} at $(i+1)^{th}$ iteration.

$\therefore w(e_{i+1}) \leq w(f)$ — (B)

$\therefore w(e_{i+1}) = w(f)$ (from (A) & (B))

So $w(T' + e_{i+1} - f) = w(T')$

ie, $T' + e_{i+1} - f$ is also a MST with one more edge in common with T.

If $E(T) \neq E(T' + e_{i+1} - f)$, we continue the above Process enough # of times to conclude that T is MST.

Prim's algorithm

- The set A forms a single tree

- The safe edge added to A is always a least-weight edge connecting the tree to a vertex not in the tree.

Prim's Algorithm :

Step 1: Select a vertex of G arbitrarily (Say v_1).

Select an edge of minimum weight among the edges incident on v_1, Say v_1v_2

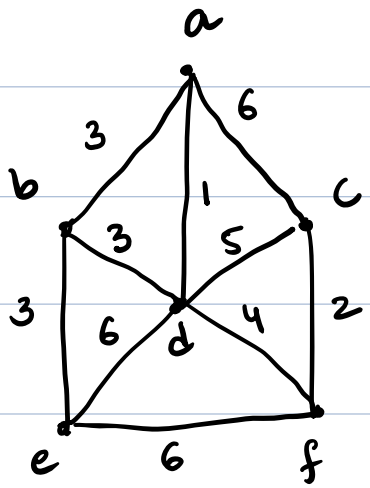
Let T_1 be the tree with vertices v_1, v_2 & the end v_1v_2

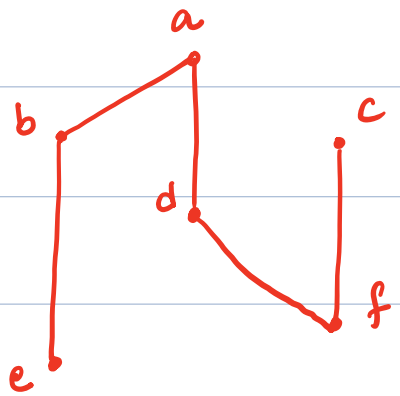
Step 2: (Recursive): Suppose we have selected the vertices $v_1, v_2, \ldots v_k$ & $k-1$ edges and forming a tree T_{k-1}, Select a minimum weight edge among all the edges such that one end point is in $\{v_1 \ldots v_k\}$ & the other is in $V \setminus \{v_1 \ldots v_k\}$.

Say v_iv_{k+1}. Define T_k to be the tree obtained by joining v_{k+1} to v_i in T_{k-1}.

Step 4: STOP when $n-1$ edges are selected and output $T = T_{n-1}$ as MST of G.

Ex 1





MST obtained from Prim's Algorithm.

The order in which edges are picked is

ad, ab, be, df, fc

Theorem: Any Tree T constructed by Prim's algorithm is a MST.

Proof: Clearly T is a tree.

Let $e_1, e_2, \ldots e_{n-1}$ be the order of edges in which they were selected to construct T.

We show the following

[*] if there is a MST T with $e_1, e_2, \ldots e_{k-1} \in T$ and $e_k \notin T$, then there is a MST T' with $e_1 e_2 \ldots e_{k-1}, e_k \in T'$.

$e_k = (v_i, v_{k+1})$ where $v_i \in \{v_1 \ldots v_k\}$ &

$v_{k+1} \in V \setminus \{v_1 \ldots v_k\}$

$T + e_k$ Contains a Unique cycle.

So there exists uv, $u \in \{v_1 \ldots v_k\}$, $v \in V \setminus \{v_1 \ldots v_k\}$

$T' = T + e_k - f$

At k^{th} iteration we have added e_k so

$$w(e_k) \leq w(f)$$

$$\therefore \quad w(T + e_k - f) \leq w(T) + w(e_k) - w(f)$$

$$\leq w(T)$$

But T is a MST

$$w(T + e_k - f) = w(T)$$

So $T + e_k - f$ is also a MST of G

containing $e_1, e_2, \ldots e_{k-1}, e_k$.

$\therefore$ Result follows from [*].

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)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

The MST A for G is

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

Running time of Prim's algorithm

Running time depends on the implementation of min-priority queue Q.

For lines 1-5 we can use BUILD-MIN-HEAP Procedure, which takes $O(|V|)$ time.

While loop executes $|V|$ times and EXTRACT-MIN operation takes $O(\log |V|)$ time.

Lines 8-11 executes in $O(|E|)$ time

- line 8 runs at most $2|E|$ times
- line 9, we can test whether a vertex is present in Q by keeping a bit for each vertex that tells whether or not it is in Q.
- line 11, involves an implicit DECREASE-KEY operation on the min-heap, which takes $O(\log |V|)$ time.

Total time $= O(|V| \log |V| + |E| \log |V|) = O(|E| \log |V|)$

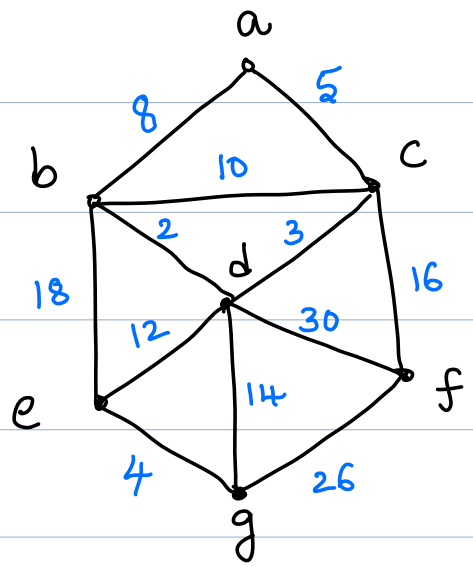
The asymptotic running time of Prim's algorithm can be improved to $O(|E| + |V|\log(|V|)$ by using Fibonacci heaps. [Ref Cormen Chap 23 For details]

Borůvka's algorithm [Ref: chap 7.3 in Jeff Erickson book]

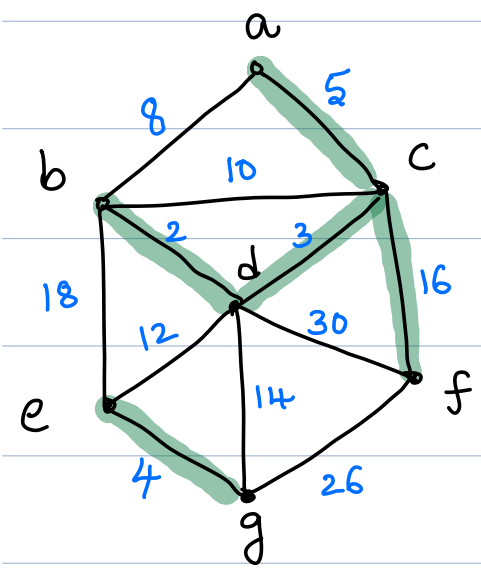
Idea is to add all the safe edges and recurse.

Example:

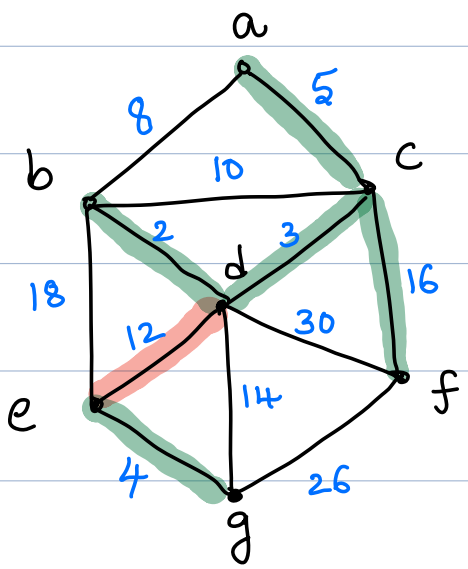
We apply Boruvka's algorithm on the following example.



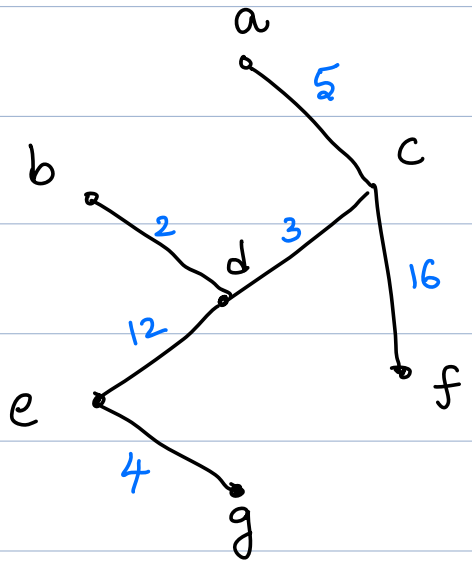
1st step: All green edges are added.



2nd step only edge de is added.



Final MST Obtained



Running time: $O(|E| \log |V|)$ (worst case)

Exercises :

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.

(3) Strange Spanning tree :- (SST)

A SST of a graph G is a Spanning tree of G whose largest edge weight is minimum over all Spanning trees of G.

The weight of the SST is the weight of the maximum weight edge in T.

(Q1) True/False : Any MST is a SST

(Q2) I/P: Graph G , integer b

Question: Does G have SST of value $\leq b$?

Design an algorithm to Solve the above Problem.