Graph Algorithm

- Graph, a pervasive data structure in computer science. Hundreds of interesting computational problems defined in terms of graph.
- A graph $G = (V, E)$, V set of vertices $\{v_1, v_2, \ldots, v_n\}$, E set of edges $\{(u, v) : u, v \in V\}$.

A graph $G = (V, E)$

V a set of vertices, $|V|$: number of vertices.
E a set of edges, $|E|$: number of edges.
In the book, it might uses V to represent $|V|$ and E to represent $|E|$.
Time complexity is defined in terms of the two variables V and E.
Vertex set of a graph G: $V[G]$, edge set $E[G]$.

Representation of Graphs

- Adjacency list,
  - provide a compact way to represent sparse graph ($|E|$ is much less than $|V|^2$)
  - Adjacency matrix, $G = (V, E)$ consists of an array Adj of $|V|$ lists.
  - For each $u \in V$, $Adj[u]$ contains pointers to all the vertices v, s.t., $(u, v) \in E$.
  - $Adj[u]$ consists of all the vertices adjacent to u in G.

Adjacency Matrix

- $G = (V, E)$, vertices are numbered $1, 2, \ldots, |V|$.
- A $|V| \times |V|$ matrix $A = (a_{i,j})$ s.t.,

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- A graph is directed, edges are arcs.
- Weighted, edges has an associated weight,
- weight function : $w : E \to R$,
- $w(u, v)$: weight of edge $(u, v) \in E$.

Minimum Spanning Tree

- Design of electronic circuit, to connect n pings, try to use the least amount of wire, there are $n-1$ wires.
- Model the problem as a weighted graph $G=(V,E)$, weights are distance between pings.
- Find a spanning tree T that total weight $w(T)=\sum_{(u,v)\in T} w(u,v)$ is the least.
- spanning tree, the tree span the graph,
- the least cost, minimum-spanning-tree problem.

Growing a minimum spanning tree

- Input: a connected, undirected graph $G = (V, E)$,
- with weight function $w : E \to R$.
- wish to find the minimum spanning tree of G.

Greedy Strategy, A "Generic Strategy"

- Grwoing a minimum spanning tree one at a time.
- maintain the loop invariant, "prior to each iteration, A is a subset of some minimum spanning tree".
- At each step, determine an edge (u, v) that can be added to A without violating the invariant ($A \cup \{(u, v)\}$ is also a subset of the minimum spanning tree).
- (u, v) a safe edge of A.
- Keep on inserting safe edges until MST is formed.
- Tricky part is to find a safe edge.

Some definitions

- A **cut** $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V.
- An edge $(u, v) \in E$ **crosses** the cut $(S, V - S)$ if one of its endpoints is in S and and the other is in $V - S$.
- A cut **respects** the set A of edges if no edges 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.

Theorem: Let $G = (V, E)$ be a connected undirected graph with a real-value 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 $(S, V - S)$. The edge $(u.v)$ is safe for A.

Proof Let T be an minimum spanning tree that includes A and assume that T does not contain the light edge (u, v).
We shall construct another minimum spanning tree T' that includes $A \cup \{(u, v)\}$.
Since T does not include (u, v), inserting (u, v) forms a cycle with the edges on the path p from u to v in T.
u and v are on opposite sides of the cut $(S, V - S)$, there must be an edge on the path crosses the cut $(S, V - S)$. Let (x, y) be the edge.
Note that both (x, y) and (u, v) are edge crossing and (u, v) is the light edge.

Removing (x, y) breaks the tree T; adding (u, v) reconnects a tree T'. We have $T' = T - \{(x, y)\} \cup \{(u, v)\}$.

$$\begin{aligned} w(T') &= w(T) - w(x, y) + w(u, v) \\ &\leq w(T), \ \text{since } (u, v) \text{ is light.} \end{aligned}$$

But T is minimum spanning tree, so $w(T) \leq w(T')$; thus T' must be a minimum spanning tree.
It remain to show that (u, v) is a safe edge for A. We have $A \subseteq T'$, since $A \subseteq T$ and $(x, y) \notin A$; $A \cup \{(u, v)\} \subseteq T'$. And, since T' is the minimum spanning tree, (u, v) is safe for A.

Corollary Let $G = (V, E)$ be a connected weighted undirected graph. Let A be a subset of E that is included in some minimum spanning tree of G, and let C be a connected component (tree) in the forest $G_A = (V, A)$. If (u, v) is a light edge connecting C to some other component in G_A, then (u, v) is safe for A.
proof The cut $(C, V - C)$ respect A, and (u, v) is a light edge for the cut.

Kruskal's Algorithm

- Given a weighted undirected graph $G = (V, E)$, preprocess the edges
- Maintain a set A that is a forest.
-
    - sort the edges according their weights.
    - put edges in a priority queue
- Iteratively do the following,
    - Take out the least weight edge, check if it is safe (form cycle?).
    - Include the edge if the edge is a safe
    - until a single tree is formed.

Prim's Algorithm

- Maintain a set A that is a single tree.
- Start with a tree having a single node s,
- choose the least-weight edge connecting the tree to a vertex not in the tree.
- until the a tree connecting all vertices.

Time Complexity

- Kruskal's Algorithm,
  - Presort or heap operation $O(E \log E)$,
  - For each edge, check if it is safe, $\alpha(V)$ (two FINDs). There are at most E edges.
  - Insert it into the tree and make two trees in the forest become one, $O(1)$, (a UNION, $n - 1 times$).
  - Total cost $O(E \log E) + O(E\alpha(V)$
- Prim's Algorithm
  - V elements stored in the Fibonacci Heap.
  - EXTRACT-MIN can be done in $O(\log n)$ amortized time.
  - DECREASE-KEY can be done in $O(1)$ amortized time. There are at most E times DECREASE-KEY.
  - Total cost is $O(E + V \lg V)$.