Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

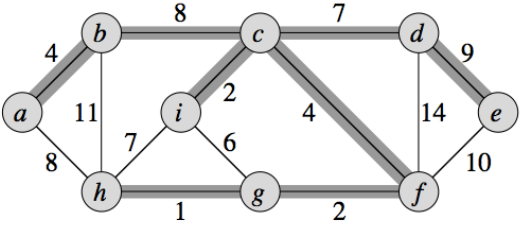
Graph Theoretical Problems

- Basec Definitions
- Graph Representation
- Graph Traversal (BFS, DFS)
- Topological Sort
- Strongly Connected Components
- Shortest Paths
  - Single-Source All Destination (Dijkstra and Bellman-Ford Algorithms)
  - All-Pairs (Matrix Multiplication, Floyd-Warshall, and Johnson's Algorithms)
- Minimum Spanning Tree (Kruskal, Prim)

Minimum Spanning Tree

Definition (Minimum Spanning Tree)

Assume that we have a connected, undirected graph $G = (V, E)$ with a weight function $w : E \mapsto \mathbb{R}$, and we wish to find an acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight $w(T) = \sum_{(u,v)\in T} w(u, v)$ is minimized.



Minimum Spanning Tree

A greedy strategy is captured by the following generic algorithm, which grows the minimum spanning tree one edge at a time. The algorithm manages a set of edges A, maintaining the following loop invariant:

Prior to each iteration, A is a subset of some minimum spanning tree.

At each step, we determine an edge (u, v) that can be added to A without violating this invariant, in the sense that $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree. We call such an edge a safe edge for A, since it can be safely added to A while maintaining the invariant.

GENERIC-MST(G, w)

$A \leftarrow \emptyset$
while A does not form a spanning tree
    do find an edge (u, v) that is safe for A
        $A \leftarrow A \cup \{(u, v)\}$
return A

Minimum Spanning Tree

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

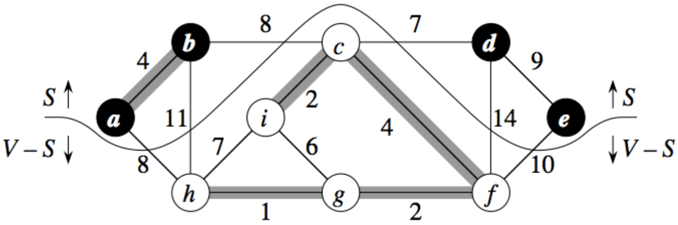
We use the loop invariant as follows:

- **Initialization:** After line 1, the set A trivially satisfies the loop invariant.
- **Maintenance:** The loop in lines $2-4$ maintains the invariant by adding only safe edges.
- **Termination:** All edges added to A are in a minimum spanning tree, and so the set A is returned in line 5 must be a minimum spanning tree.

Minimum Spanning Tree

Definition

A cut $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V. We say that an edge $(u, v) \in E$ 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.



How we can find a safe edge?

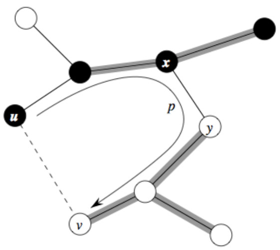
Minimum Spanning Tree

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.

Proof.

- Let T be a minimum spanning tree that includes A.
- Assume that T does not contain the light edge (u, v), since if it does, we are done.
- We shall construct another minimum spanning tree T' that includes $A \cup \{(u, v)\}$.
- The edge (u, v) forms a cycle with the edges on the path p from u to v in T.



Minimum Spanning Tree

Proof.

- Since u and v are on opposite sides of the cut $(S, V-S)$, there is at least one edge in T on the path p that also crosses the cut. Let (x,y) be any such edge. The edge (x,y) is not in A, because the cut respects A.
- Since (x,y) is on the unique path from u to v in T, removing (x,y) breaks T into two components. Adding (u,v) reconnects them to form a new spanning tree $T' = T - \{(x,y)\} \cup \{(u,v)\}$.
- Since (u,v) is a light edge crossing $(S, V-S)$ and (x,y) also crosses this cut, $w(u,v) \leq w(x,y)$. Therefore,

$$w(T') = w(T) - w(x,y) + w(u,v) \leq w(T).$$

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

$\square$

Minimum Spanning Tree: Kruskals algorithm

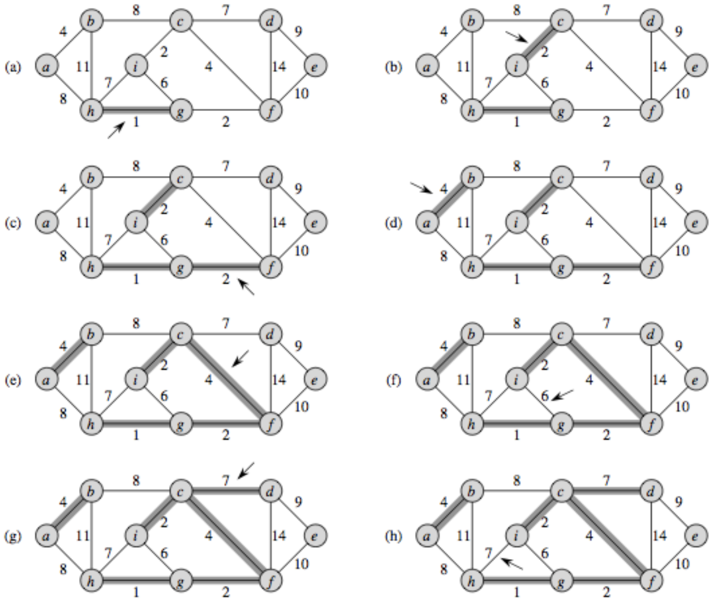
The Kruskals algorithm finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge (u, v) of least weight.

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

It uses a disjoint-set data structure:

- Each set contains the vertices in a tree of the current forest.
- The operation FIND-SET(u) returns a representative element from the set that contains u. Thus, we can determine whether two vertices u and v belong to the same tree by testing whether FIND-SET(u) equals FIND-SET(v).
- The combining of trees is accomplished by the UNION procedure.

Minimum Spanning Tree: Kruskals algorithm



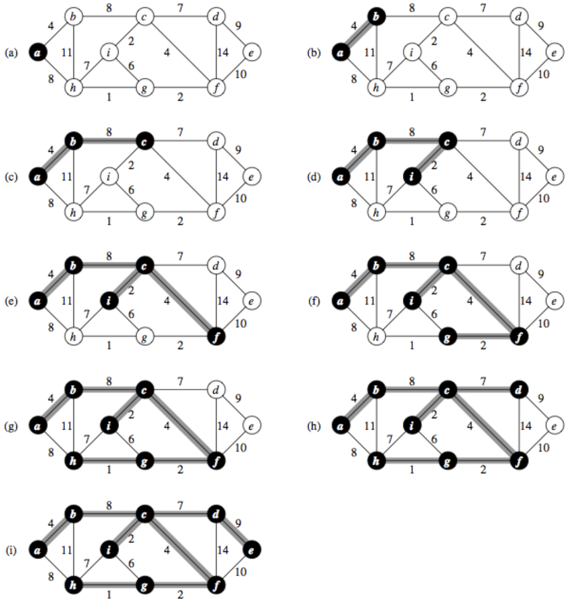
Minimum Spanning Tree: Prim's algorithm

- Prim's algorithm has the property that the edges in the set A always form a single tree.
- The tree starts from an arbitrary root vertex r and grows until the tree spans all the vertices in V.
- At each step, a light edge is added to the tree A that connects A to an isolated vertex.
- All vertices that are not in the tree reside in a min-priority queue Q based on a key field. For each vertex v, $key[v]$ is the minimum weight of any edge connecting v to a vertex in the tree; by convention, $key[v] = \infty$ if there is no such edge.
- The field $\pi[v]$ names the parent of v in the tree.

Minimum Spanning Tree: Prim's algorithm

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

Minimum Spanning Tree: Prim's algorithm



Exercises

1. Show that if an edge (u, v) is contained in some minimum spanning tree, then it is a light edge crossing some cut of the graph.
2. Give a simple example of a graph such that the set of edges $\{(u, v) \ : \ \text{there exists a cut } (S, VS) \text{ such that } (u, v) \text{ is a light edge crossing } (S, VS)\}$ does not form a minimum spanning tree.
3. Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counterexample.
4. Let T be a minimum spanning tree of a graph $G = (V, E)$, and let V' be a subset of V. Let T' be the subgraph of T induced by V', and let G' be the subgraph of G induced by V'. Show that if T' is connected, then T' is a minimum spanning tree of G.
5. Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Kruskals algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W? Do the same for Prim's algorithm.