

Graphs

Definition 1. An **undirected graph** G is a pair (V, E) where

- V is the set of vertices,
- $E \subseteq V^2$ is the set of edges (unordered pairs)

$$E = \{(u, v) \mid u, v \in V\}.$$

In a **directed graph** the edges have directions (ordered pairs).

A **weighted graph** includes a weight function

$$w : E \rightarrow R$$

attaching a value (weight) to each edge.

Running time of a graph algorithm on a graph $G = (V, E)$ is usually in terms of $|V|$ and $|E|$.

Sometimes, the running time can be a function of weights on the edges.

Definition 2. A path in a graph $G = (V, E)$ is a sequence of vertices v_1, v_2, \dots, v_k such that for $1 \leq i \leq k - 1$, $(v_i, v_{i+1}) \in E$.

Definition 3. A cycle in a graph $G = (V, E)$ is a path v_1, v_2, \dots, v_k such that $(v_k, v_1) \in E$.

Definition 4. A tree is a graph with no cycles.

Definition 5. A graph $H = (V', E')$ is a **subgraph** of $G = (V, E)$ iff $V' \subseteq V$ and $E' \subseteq E$. A **spanning subgraph** if $V' = V$.

Graph Representation

Adjacency List: A link list for each vertex. The list contains a pointer to each neighbor of the vertex. (and a weight for weighted graph.)

Total space $O(V + E)$.

Sequential (linear) access.

Adjacency Matrix: A $|V| \times |V|$ array, $A[i, j] = 1$ iff (i, j) is an edge in the graph, otherwise $A[i, j] = 0$. ($A[i, j] = w_{i,j}$ for weighted graph.)

Total space $O(V^2)$.

Random access.

Graph Search Algorithms

1. **Breadth-first search.** Starting from a vertex s , visits vertices in increasing order of distance from s .

Can be implemented in $O(|V| + |E|)$.

2. **Depth-first search.** Starting from a vertex s , edges are explored out of the most recently visited vertex.

Can be implemented in $O(|V| + |E|)$.

Spanning Tree

Given a graph $G = (V, E)$ a **spanning tree** T in G is a subgraph of G that

- is connected;
- includes all the vertices of G ;
- has no cycles;

Spanning Tree Characterizations

Theorem 1. *A spanning tree T in a connected graph $G = (V, E)$ is a maximal spanning subgraph of G that is a tree (i.e. any edge added to T closes a cycle).*

Proof. Proof by contradiction:

Assume that T is a spanning tree in G , and adding edge (v, u) to T does not close a cycle.

There is a path p from u to v using only edges of T (since it's a spanning tree).

Thus, the path p and the edge (v, u) must include a cycle. \square

The Size of a Spanning Tree

Lemma 1. *A spanning tree of a connected graph $G = (V, E)$ has $|V| - 1$ edges.*

Proof. By induction on $|V|$.

Base case ($|V| = 1$) is trivial.

Assume that the lemma is true for all graphs of size $< |V|$.

Consider a connected graph G of size $|V|$ and a spanning tree T of G .

Remove an edge (u, v) of T . This breaks T into two components — T_1 and T_2 .

T_1 (T_2) is a spanning tree of the subgraph induced by the vertices of T_1 (T_2).

The number of edges in T is $|T_1| - 1 + |T_2| - 1 + 1 = |T| - 1 = |V| - 1$. \square

Minimum Spanning Tree (MST)

Given a weighted undirected graph $G = (V, E)$ find a spanning tree of G with minimum total weight (i.e., sum of the weights of its edges is minimum).

Fundamental graph optimization problem with applications in many settings, e.g., broadcasting in a communication network.

We will design two greedy algorithms for this problem.

A generic MST algorithm

$MST(G)$

```
1  $A = \phi$ 
2 while  $A$  does not form a spanning tree
3     find an edge  $(u, v)$  that is safe to add to  $A$ 
4      $A = A \cup \{(u, v)\}$ 
5 return  $A$ 
```

Assume that A is a subset of a minimum spanning tree of G .

An edge is **safe** for A if $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree.

Key Theorem

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

An edge **crosses** the cut if one of its endpoints is in S and the other is in $V - S$.

A cut **respects** a set A of edges if no edge in A crosses the cut.

Theorem 2. *Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function w defined on E . Let $A \subseteq E$ be a subset of a minimum spanning tree of G . Let $(S, V - S)$ be any cut of G that respects A and let (u, v) be a minimum weight edge crossing the cut. Then (u, v) is safe for A .*

Proof

Let T be a MST that includes A .

If T contains (u, v) , we are done. Otherwise, we will construct another MST T' that includes $A \cup \{(u, v)\}$.

(u, v) completes a cycle with the edges on the path p from (u, v) in T .

There is at least one edge $(x, y) \notin A$ in p that crosses the cut $(S, V - S)$.

Remove (x, y) from T and add (u, v) to T to get a new spanning tree T' .

The weight of T' is:

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

Thus T' is also a MST.

Kruskal's Algorithm

Here the set A is a forest, starting initially with singleton vertices.

Pick a minimum weight edge that does not close a cycle with A and add it to A . This edge will be safe for A .

MST – Kruskal(G)

```
1  $A = \phi$ 
2 for each vertex  $v \in V$ 
3   Make-Set( $v$ );
4 sort the edges of  $E$  into nondecreasing order by weight  $w$ 
5 for each edge  $(u, v) \in E$  (according to the sorted order)
6   if Find-Set( $u$ )  $\neq$  Find-Set( $v$ )
7      $A = A \cup \{(u, v)\}$ 
8     Union( $u, v$ );
9 return  $A$ 
```

Can be implemented in $O(|E| \log |V|)$ time.