Intro on graphs

Undirected graphs. A graph $G = (V, E)$ is a pair, where V is some (finite) set of vertices (nodes) and $E \subseteq [V]^2$ is a set of edges (unordered pairs).

Dictionary.

(i) For a vertex $v \in V$, $\deg(v) = |\{x \in V : \{x, v\} \in E\}|$ is its **degree**.

(ii) A **path** in G is a sequence $x_0, x_1, \ldots, x_n$ of vertices such that $\{x_i, x_{i+1}\} \in E$ for every $i = 0, \ldots, n-1$.

(iii) A **cycle** is a path such that $x_0 = x_n$.

(iv) G is **connected** if every pair of distinct vertices can be joined by a path.

Spanning trees

Definition. A tree is a connected graph without cycles.
Given a graph $G = (V, E)$, its **spanning tree** is any tree of the form $T = (V, E')$ where $E' \subseteq E$.

Basic properties of trees.

(1) Every tree of at least two vertices contains a leaf, that is a vertex of degree 1.

(2) If $T = (V, E)$ is a tree then $|E| = |V| - 1$.

(3) A graph $G = (V, E)$ is a tree if and only if G is connected and $|E| = |G| - 1$.

(4) Every (finite) connected graph has a spanning tree.

Minimal Spanning Tree (MST)

MSP. Consider a connected graph $G = (V, E)$ and let $c : E \to \mathbb{R}_+$ be the cost function. Find the cheapest spanning tree $T = (V, E')$, the one minimizing

$$c(E') := \sum_{e \in E'} c(e).$$

A (greedy) algorithm for MST. Let $n = |V|$.

(1) Take any $v_1 \in V$ and set $V_1 = \{v_1\}$, $E_1 = \emptyset$.
(2) Given a tree $T_k = (V_k, E_k)$, if $k = n$ then STOP.
(3) For $k < n$ consider a family F of all edges $e = \{x, y\}$, where $x \in V_k$, $y \in V \setminus V_k$. Choose $e^* = \{x^*, y^*\} \in F$ such that $c(e^*) = \min\{c(e) : e \in F\}$ and put

$$V_{k+1} = V_k \cup \{y^*\}, \quad E_{k+1} = E_k \cup \{e^*\}$$

GoTo 2.

The proof that it works.

Note that every T_k is a tree and T_n is a spanning tree. We need to check that $c(T_n)$ minimizes the costs.

Verify inductively that T_k is 'contained' (can be extended) to some optimal spanning tree. This is obvious for $k = 1$. Assume the claim form some k and check it for $k + 1$.

We know that $E_k \subseteq E'$, where (V, E') is some optimal tree. At step $k + 1$ we added some edge e^*; if $e^* \in E'$ then there is nothing to prove. Otherwise, $e^* \notin E'$ so the family od edges $E' \cup \{e^*\}$ must contain a cycle C. Take e^{**} which is in that cycle and in F. Then the edges from $E'' = E' \setminus \{e^{**}\} \cup \{e^*\}$ again form a tree. We know that $c(e^*) \leqslant c(e^{**})$ by our choice. On the other hand $c(E'') \geqslant c(E')$ gives $c(e^*) \geqslant c(e^{**})$. Hence E'' also form an optimal tree and it extends T_{k+1}.

Directed graphs

Definition. A directed graph G is a pair (V, A), where $A \subseteq V \times V \setminus \Delta$.

Shortest paths. Consider a directed graph $G = (V, A)$ and a function $c : A \to \mathbb{R}_+$ (where $c(a)$ is a cost or length of $a \in A$). Find the shortest path between two given vertices.

Dijkstra's algorithm[1]

Suppose that $V = \{1, \dots, n\}$. We find, for every $i \neq n$, the length of the shortest path from i to n.

We can assume that in G there are all possible arcs; for those virtual a we put $c(a) = \infty$.

Algorithm.

(1) If there is only one vertex then STOP.

(2) Find $k \neq n$ such that

$$c(k, n) = \min_{i \neq n} c(i, n).$$

Put $d_k = c(k, n)$.

(3) For $i \neq k, n$ set

$$c(i, n) := \min(c(i, n), c(i, k) + c(k, n)).$$

(4) Remove the vertex k; GoTo (1).

While removing k we update the distances:

$$c(i, j) := \min(c(i, j), c(i, k) + c(k, j)).$$

It works!

Theorem. When the algorithm terminates we get the shortest distances $d_1, \dots, d_{n-1}$ from vertices $1, \dots, n-1$ to n.

Why? If $c(k, n) = \min_{i \neq n} c(i, n)$ then $d_k = c(k, n)$ and $d_i \geqslant d_k$ for other i.

Recovering the shortest path

Notation. In a directed graph $G = (V, A)$ for $v \in V$ we write

$$\mathrm{Out}(v) = \{x \in V : (v, x) \in A\},$$

$$\mathrm{In}(v) = \{x \in V : (x, v) \in A\}.$$

[1] Edsger W. Dijkstra (1930–2002)

Once we have the shortest distances $d_1, \dots, d_{n-1}$ given we define the shortest path from 1 to n by the rule: if you are at the vertex x then go to y such that

$$c(x, y) + d_y = \min_{z \in \mathrm{Out}(x)} \left(c(x, z) + d_z \right).$$