Combinatorial optimization for undergraduate students

Lecture note 4. Shortest path-BFS algorithm

Lecturer : O-joung Kwon
Spring, 2018

One of the most common applications of graphs in everyday life is representing networks for traffic or for data communication. We often look for paths that are good or even best in some respect : shortest, fastest, or cheapest.. So, finding a shortest path in graphs or digraphs is very natural and fundamental problem.

Shortest Path problem

Let G be a graph and $w : E(G) \to \mathbb{R}$ be a mapping. We call the pair (G, w) a *network*, and $w(e)$ is called the *length* of e. For a path $W = (e_1, e_2, \ldots, e_n)$, the length of W is defined as

$$w(W) := w(e_1) + w(e_2) + \cdots w(e_n).$$

For two vertices a, b, we define the *distance* between a and b as the minimum over all lengths of paths from a to b. If a cannot reach to b, then let distance be ∞. Any path from a to b that has minimum length is called a *shortest path* from a to b.

Question : for two vertices a and b, can we find a shortest path from a to b?

Some examples.

1. A trading ship travels from port a to port b, where the route may be chosen freely. The routes are represented by trails in a digraph G, and the length $w(e)$ of an edge $e = xy$ signifies the profit gained by going from x to y. For some edge e, we may have negative $w(e)$ when the ship does not carry something, but use oil.

2. In many applications, the length of an edge indicates the probability of its failing, for instance in networks of telephone lines. One may look for the route having the highest probability for not failing. For this, let $p(i, j)$ be the probability that edge (i, j) does not fail. For a path $(e_1, e_2, \ldots, e_n)$, the probability that does not fail is the product $p(e_1)p(e_2)\cdots p(e_n)$.

Motivated by this observation, let $w(e) = -\log p(e)$. Then instead of maximizing $p(e_1)p(e_2)\cdots p(e_n)$, we can consider to minimize $w(e_1)+w(e_2)+\cdots w(e_n) = -\log(p(e_1)p(e_2)\cdots p(e_n))$.

Moore's algorithm (1957) for case where every edge has weight 1. It is known as *breadth first search*, or, for short BFS. It is one of the most fundamental methods in algorithmic graph theory.

Let G be a graph (or digraph) given by adjacency lists A_v.

(main idea)

- We set $D_0 := \{v\}$.
- Recursively, for each i, we set all the vertices in $V(G) \setminus (D_0 \cup D_1 \cup \cdots D_i)$ having a neighbor in D_i as D_{i+1}.

Then D_i becomes the set of vertices having distance exactly i to v.

We write this algorithm as follows:

Theorem 1. *Moore's Algorithm has complexity* $\mathcal{O}(|E(G)|)$. *At the end of the algorithm, every vertex* t *get the value*

$$d(s,t) = \begin{cases} d(t) & \text{if } d(t) \text{ is defined,} \\ \infty & \text{otherwise.} \end{cases}$$

We now turn to the problem of determining shortest paths in a general network.

We assume two things.

- The given network (G, w) does not contain any cycles of negative length.
- G is a directed graph.

We can reduce an undirected graph into a directed graph by replacing each edge with two directed edges in both directions.

Lemma 1. *Let W be a shortest path from s to t in the network (G, w), and let v be a vertex on W. Then the subpath of W from s to v is a shortest path from s to v.*

Theorem 2. *Let P be a path from s to t in G. Then P is a shortest path if and only if each edge uv in P satisfies that*

$$(*) \qquad d(s, v) = d(s, u) + w(uv).$$

Theorem 2 leads to an efficient way of storing a system of shortest paths with starting vertex s.

A vertex a from which every other vertex is accessible is called a *root* of G. A spanning subdigraph which is a directed tree with root s is called a *spanning arborescence*.

A spanning arborescence is called a *shortest path tree*, shortly SP-tree, for the network (G, w) if for each v, the unique path from s to v in the tree T has length $d(s, v)$.

Corollary 1. *Let T be a spanning arborescence of G with root s. Then T is a shortest path tree if and only if each edge uv of T satisfies the condition $(*)$.*

Theorem 3. *Let G be a digraph with root s and let $w : E \to \mathbb{R}$ be a length function of G. If the network (G, w) does not contain any directed cycles of negative length, then there exists an SP-tree with root s.*

Exercise 1. *Describe how to find an SP-tree with root s.*