# Algorithm Lab

## Week 11: Dijkstra's Algorithm

Dijkstra's algorithm is a one-to-all shortest path algorithm. Most of shortest path algorithms work on directed weighted graph. Dijkstra's algorithm not always work correctly on an arbitrary graph, it can only apply on graphs without negative cycle. We can define directed weighted graph by $\mathbf{G} = (V, E)$ where $\mathbf{V}$ is its vertices set and $\mathbf{E}$ is its edges set. And we can denote an edge by its 2 ends and its weight, in mathematical description, $e_i = (s_i, t_i, w_i) \in \mathbf{E}$, $\mathbf{s_i}, t_i \in V$. A $s$-to-t path is a sequence of edges $\mathbf{P} = (e_1, e_2, \dots, e_n)$ that $\forall \mathbf{1} \leq \mathbf{i} < \mathbf{n}, \mathbf{t_i} = s_{i+1}$ and $\mathbf{s_1} = s$, $\mathbf{t_n} = t$. The cost of $\mathbf{P}$ is $\sum_{e_i \in P} w_i$. Assume that we have 2 paths $\mathbf{P_a}$ and $\mathbf{P_b}$, we say $\mathbf{P_a}$ shorter than $\mathbf{P_b}$ if cost of $\mathbf{P_a}$ less than cost of $\mathbf{P_b}$. Respectively, the shortest path of a start-target pair is the path with minimal cost. One-to-all shortest path (or Single Source Shortest Path, SSSP) problem, by its name, is ask to find shortest paths with a given start vertex. Defined as followed:

---

Instance: weighted graph $G = (V, E)$ and start vertex $s \in V$.

Result: Some data structure can query shortest $s - t$ path by a given $t$.

---

### Description

Dijkstra's algorithm is to find and to store shortest paths in a kind of rooted spanning tree. We can denote the result by its edges set $\mathbf{T} \subseteq \mathbf{E}$. In Dijkstra's algorithm, we divide vertices to 2 groups, visited group $\mathrm{V}'$ and unvisited group $\mathbf{U}$.

If we want to move a vertex $v_i \in \mathbf{U}$ to $\mathbf{V}'$, we should apply our operation by a bridge edge $e_b = (s_b, v_i, w_b) \in E$ and $\mathbf{s_b} \in V'$.

1. $\mathbf{V}' \leftarrow \{s\}$, $\mathbf{U} \leftarrow \mathbf{V} \setminus \mathbf{V}'$, $\mathbf{T} \leftarrow \{(s, s, 0)\}$.
2. Find the minimum bridge edge $\mathrm{e_b}$ between $\mathbf{V}'$ and $\mathbf{U}$.
3. Find bridged node $\mathbf{e_r} \in \mathbf{T}$ that $\mathbf{t_r} = \mathbf{s_b}$ with minimal $\mathbf{w_b} + w_r$.
4. Make an edge $\mathbf{e'_b} = (s_b, t_b, w_b + w_r)$.
5. Update information $\mathbf{V}' \leftarrow V' \cup \{t_b\}$, $\mathbf{U} = \mathbf{U} \setminus \{t_b\}$, $\mathbf{T} \leftarrow \mathbf{T} \cup \{e'_b\}$.
6. Repeat step 2 to step 5 until $\mathbf{U} = \emptyset$.

## Questions

1 Analyze the time complexity when implement step 2 by sequential searching whole adjacency list.

2 Analyze the time complexity when implement step 2 by a priority queue that maintained the bridge edges.

  2.1 Implement the priority queue by double-linked list.

  2.2 Implement the priority queue by Min-heap.

3 Design an algorithm to reconstruct the path from spanning tree $\mathbf{T}$ (the result).

4 How to find the unreachable (have no path from $s$) set?

5 Solve http://oj.csie.ndhu.edu.tw/problem/ALG07C and analyze your code.