

This and other notes are available from <http://www.cs.berkeley.edu/~yozo/cs170.fa05/>

Dijkstra's Algorithm

```

1  $Q \leftarrow \{s\}$ 
2  $d[s] \leftarrow 0$ ;  $d[v] \leftarrow \infty$  for all other vertices  $v$ .
3  $\pi[v] \leftarrow \text{NIL}$ 
4 while  $Q$  is not empty do
5    $v \leftarrow \text{DeleteMin}(Q)$ 
6   Mark  $v$ .
7   for each edge  $(v, w)$  out of  $v$  do
8     if  $w$  is unmarked and  $d[w] > d[v] + wt(v, w)$  then
9        $d[w] \leftarrow d[v] + wt(v, w)$ 
10       $\pi[w] \leftarrow v$ 
11      Insert( $Q, w$ ).
12   end
13 end
14 end

```

Proof of Dijkstra's Algorithm

Consider the Dijkstra's Algorithm as presented in lecture. For any vertex v in G , denote by $\delta(v)$ the length of the shortest path from s to v , which must exist (why?).

We'll start with a lemma.

Lemma. At any step in the Dijkstra's Algorithm, we have

$$d[v] \geq \delta(v) \quad \text{for any vertex } v \in G.$$

Proof. We will prove by induction on the number of edge examined. At the beginning of the algorithm, the above relation holds (why?). Now suppose that the relation holds up to when $k - 1$ first edges are examined. We need to show that after examination of the k -th edge, the relation still holds. Let (u, v) be the k -th edge being examined. If there is no update to $d[v]$, then there is nothing to show (why?). Thus suppose that $d[v]$ gets updated:

$$d[v] \leftarrow d[u] + wt(u, v)$$

If $d[v] < \delta(v)$ were to be true, we must have

$$\begin{aligned} d[u] + wt(u, v) &= d[v] \\ &< \delta(v) \\ &\leq \delta(u) + wt(u, v) \end{aligned}$$

But this implies that $d[u] < \delta(u)$, contradicting our inductive hypothesis. Thus $d[v] \geq \delta(v)$ for all u at all times. Note that once the minimum is reached, it will stay that way since $d[v]$ can never increase (why?). \square

Proof of Correctness of Dijkstra's Algorithm. We claim that when a vertex v is marked, then $d[v] = \delta(v)$. This is shown by induction on the number of vertices marked so far. Let S be the set of vertices marked so far. At the beginning of the algorithm, $S = \{s\}$, $d[s] = 0$, as expected.

Now suppose that S is non-empty, and we are about to mark a vertex v . Consider the shortest path from s to v , perhaps not yet discovered. Since $s \in S$ but $v \notin S$, the shortest path must cross the boundary of S . Let the edge (x, y) be the first such crossing. Then since $x \in S$, we must have $d[x] = \delta(x)$ (why?). Also note that since $d[y]$ was updated when x was marked, we must have $d[y] = \delta(y)$ (why?). Thus we must have

$$d[y] = \delta(y) \leq \delta(v) \leq d[v].$$

But v was chosen out of priority queue (heap) before y , so $d[v] \leq d[y]$. Combining this and the above, we conclude that $d[v] = d[y]$. Thus we must have

$$\delta(v) = d[v] \quad (\text{why?})$$

Hence all the vertices in $S \cup \{v\}$ have the correct distance, and we are done. \square

Questions

- (Review) Prove or disprove the following statements. Assume that DFS was run on a graph G .
 - The vertex with the highest post-visit number lies in a source component.
 - The vertex with the lowest post-visit number lies in a sink component.
- (Review) Why is it that you have to reverse the graph G during the strongly connected component algorithm presented in class?
- (Review) Can you swap G and G^R in the Strongly Connected Component algorithm? Why?
- In the above proof of Dijkstra's algorithm, fill in the argument where (why?) appears.
- Does the above proof valid when we allow edges of zero length?
- Point out exactly where the above proof fails when we allow negative edges. (Note that there are more than one places).
- Use similar argument as above to prove that Bellman-Ford Algorithm is correct.
- Let G be a graph, and s a vertex. Suppose we run Dijkstra's Algorithm and pick out the edges in the shortest paths. What kind of graph do they form? Do the same for Bellman-Ford algorithm.
- Suppose there is only one edge with negative edge weight. Can you modify Dijkstra's algorithm to solve the single-source shortest path problem for graph with a single negative edge weight?