

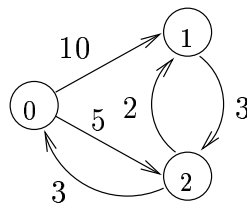
Problem 2 (20%)

Dijkstra's single source shortest path algorithm requires $O((V + E) \log V)$ time (when implemented using a binary heap) to find the minimum distance from the source to each vertex. However, it is easier to *check* the answer than to find it. In this problem, we consider a directed, connected, weighted graph (with non-negative weights) $G = (V, E)$ with n vertices $V = \{v_0, v_1, \dots, v_{n-1}\}$ and weights $w(v_i, v_j)$.

Question 2.1

Give an algorithm with running time $O(V + E)$, which given G and a sequence of distances d_i with $0 \leq i \leq n - 1$, verifies whether d_i is the minimum distance from the source v_0 to v_i .

For example, for the graph below, the algorithm should return “yes” for the sequence of distances $d = \langle 0, 7, 5 \rangle$ while it should return “no” for $d = \langle 0, 8, 5 \rangle$ and for $d = \langle 0, 5, 5 \rangle$.



Solution: We assume that the graph G is represented using an adjacency-list datastructure. This datastructure should also contain a list of vertices incident to each vertex in the graph.

CHECKDIJKSTRA(G, d)

```

1  if  $d_0 \neq 0$  return “no”
2  for  $j \leftarrow 1$  to  $n - 1$  do
3       $x \leftarrow \min\{d_i + w(v_i, v_j) : v_i \text{ is incident to } v_j\}$ 
4      if  $x \neq d_j$  return “no”
5  return “yes”
  
```

Question 2.2

Argue that your algorithm is correct. That is, show that (a) if the d_i 's are the correct shortest path values, your algorithm returns “yes”, and (b) if the d_i 's are not correct shortest path values, your algorithm returns “no.”

Solution: (This is a more detailed proof than what would be required for the written exam.)

a) Under the assumption that the d_i 's are correct shortest path values, i.e., $d_i = \delta(v_0, v_i)$, we show that the algorithm returns “yes.”

Since all weights are non-negative, there are no negative-weight cycles in G and thus $d_0 = 0$ and the algorithm will not return “no” in line 1.

Let p be a shortest path from the source v_0 to v_j for some j , $1 \leq j \leq n-1$, i.e., $v_0 \xrightarrow{p} v_j$. Since $j > 0$, we can write p as

$$p = v_0 \xrightarrow{p'} v_i \rightarrow v_j$$

for some vertex v_i and path p' . From corollary 25.2, it follows that $d_j = d_i + w(v_i, v_j)$.

We have to argue that x computed in line 3 is equal to d_j when $d_j = d_i + w(v_i, v_j)$ for (at least) one of the nodes v_i incident to v_j . The value of x cannot be larger than d_j since it is computed as a minimization over a number of terms and (at least) one of these is the term $d_i + w(v_i, v_j)$. The value of x cannot be smaller than $d_i + w(v_i, v_j)$ since then path p above is not a shortest path. Thus, x is equal to d_j and the algorithm returns “yes.”

- b)** It follows from lemma 25.3 that the weight of the shortest path from v_0 to v_j is the shortest path to a vertex v_i incident to v_j plus $w(v_i, v_j)$, i.e., $\delta(v_0, v_j)$ is given by

$$\delta(v_0, v_j) = \begin{cases} 0 & \text{if } j = 0 \\ \min\{\delta(v_0, v_i) + w(v_i, v_j) : v_i \text{ is incident to } v_j\} & 1 \leq j \leq n-1 \end{cases}$$

Clearly, the algorithm returns “no” if for some j , $d_j \neq \delta(v_0, v_j)$.