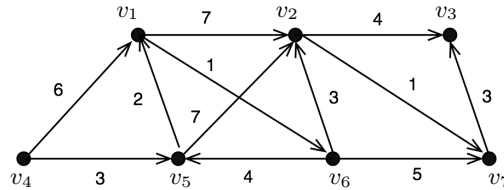


Summary on Lecture 21, May 21, 2018

Digraphs and shortest paths

Let $G = (V, E)$ be a graph or digraph. We say that G is *weighted* if we are given a function $\text{wt} : E \rightarrow (0, \infty)$. In other words, each edge $e \in E$ is given a positive weight $\text{wt}(e)$. It is convenient to have a convention that if $v, v' \in V(G)$ are not connected by an edge, then a *virtual edge* (v, v') has weight ∞ . We also accept a convention that a *virtual edge* (v, v) has zero weight. Then once we have a weighted graph, it makes sense to determine a *shortest distance* $d(v, v')$ from a vertex v to a vertex v' .

Fig. 1. Weighted digraph G_1 .

Example. Consider the digraph G_1 in Fig. 1. It is easy to check that the shortest distance (weight) from v_4 to v_1 is 5 when we take the route $v_4 \rightarrow v_5 \rightarrow v_1$, and it is shorter than a “direct” shot $v_4 \rightarrow v_1$.

Now the “shortest” paths $v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ and $v_4 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3$ from v_4 to v_3 have weights $6 + 7 + 4 = 17$ and $3 + 7 + 4 = 14$, respectively, but the “longer” path $v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_6 \rightarrow v_2 \rightarrow v_7 \rightarrow v_4$ has weight $3 + 2 + 1 + 3 + 1 + 3 = 13$, which is less than either of these. Thus length is not directly related to weight. \square

There are special vertices in G_1 , namely, v_4 is a *source*, and v_3 is a *sink*. Indeed, v_4 has only *outbounded* edges, and v_3 only *inbounded* edges.

In the case when a weighted digraph $G = (V, E)$ with a only one source v_0 and one sink v_* and has no loops and parallel edges), it is called a *scheduling network*. The objective here is to find a shortest path (i.e., with minimal weight) from a source to a sink. This is known as *scheduling problem*. We assume that a weight of every edge is positive. In fact, we resolve more general problem, namely, for each vertex $v_0 \in V$ we determine

- (i) the distance $d(v_0, v)$ for every $v \in V$;
- (ii) a shortest path from v_0 to v if such a path exists.

We subdivide the set of vertices V into two subsets: $V = S \cup \bar{S}$, where $v_0 \in S$, and $\bar{S} = V \setminus S$, so that $S \cap \bar{S} = \emptyset$. Then we define the distance $d(v_0, \bar{S})$:

$$d(v_0, \bar{S}) = \min\{d(v_0, \bar{v}) \mid \bar{v} \in \bar{S}\}.$$

If the distance $d(v_0, \bar{S})$ is finite, then there exists at least one vertex $\bar{v}_* \in \bar{S}$ such that $d(v_0, \bar{S}) = d(v_0, \bar{v}_*)$. We choose a shortest path P from v_0 to \bar{v}_* :

$$P : v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k \rightarrow v_{k+1} = \bar{v}_*$$

Now we make the following observations:

- (1) The vertices v_0, v_1, \dots, v_k are in S . Indeed, if v_i would be in \bar{S} for some $i = 1, \dots, k$, then the path $P_i : v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_i$ would be shorter than P .
- (ii) The path $P_i : v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_i$ is the shortest path in G from v_0 to v_i for each $i = 1, \dots, k$. Indeed, if there would be a shorter path P'_i from v_0 to v_i , then we would find a path P'_i composed with the path $v_i \rightarrow \cdots \rightarrow v_{k+1} = \bar{v}_*$ which is shorter than P .

Lemma 1. $d(v_0, \bar{S}) = \min\{ d(v_0, u) + \text{wt}(u, \bar{v}) \mid u \in S, \bar{v} \in \bar{S} \}$.

Then if the minimum occurs when $u = u_* \in S$ and $\bar{v} = \bar{v}_* \in \bar{S}$, then we have that

$$d(v_0, \bar{S}) = d(v_0, u_*) + \text{wt}(u_*, \bar{v}_*). \quad (1)$$

We use the formula (??) to explain the idea of the **Dijkstra's Shortest Path Algorithm**. It goes as follows. Let $S_0 = \{v_0\}$, and $\bar{S}_0 = V \setminus S_0$. We find $d(v_0, \bar{S}_0)$:

$$d(v_0, \bar{S}_0) = \min\{ \text{wt}(v_0, \bar{v}) \mid \bar{v} \in \bar{S}_0 \}.$$

Let $v_1 \in \bar{S}_0$ be such that $d(v_0, \bar{S}_0) = d(v_0, v_1)$. Then we define $S_1 = S_0 \cup \{v_1\}$, and $\bar{S}_1 = V \setminus S_1$. Then we find $d(v_0, \bar{S}_1)$:

$$d(v_0, \bar{S}_1) = \min\{ d(v_0, v_1) + \text{wt}(v_1, \bar{v}) \mid \bar{v} \in \bar{S}_1 \}.$$

We find $v_2 \in \bar{S}_1$ such that $d(v_0, \bar{S}_1) = d(v_0, v_1) + \text{wt}(v_1, v_2)$. If we proceed in such a way, we construct a set $S_i = \{v_0, v_1, \dots, v_i\}$, and find the distance $d(v_0, \bar{S}_i)$:

$$d(v_0, \bar{S}_i) = \min\{ d(v_0, v_i) + \text{wt}(v_i, \bar{v}) \mid \bar{v} \in \bar{S}_i \},$$

and find the next vertex $v_{i+1} \in \bar{S}_i$ such that $d(v_0, \bar{S}_i) = \min\{ d(v_0, v_i) + \text{wt}(v_i, v_{i+1}) \}$. The algorithm will stop if either $d(v_0, \bar{S}_i) = \infty$, or $|S_i| = |V|$. Fig. 2 shows the resulting path for the graph G_1 as above.

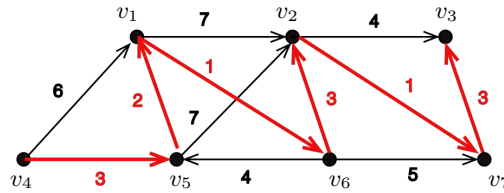


Fig. 2. A shortest path from v_4 to v_3 .

Next time we analyze the Dijkstra's Shortest Path Algorithm in detail.