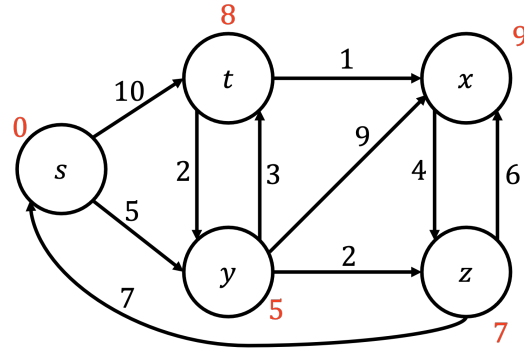


Greedy I: Proving Correctness of Dijkstra for Shortest Path

Algorithm 1 Dijkstra's Algorithm(G, w)

Input: Graph $G = (V, E)$ and weights w .
 Let S be the set of explored nodes
for each $u \in S$, store a distance $d(u)$
 Initially $S = \{s\}$ and $d(s) = 0$
while $S \neq V$ **do**
 let $d'(v) = \min_{(u,v) \in E, u \in S} d(u) + w_{uv}$
 select $v \in \operatorname{argmin}_{v \notin S} d'(v)$
 add v to S and set $d(v) = d'(v)$
end while



Let P_v denote the shortest path to v from s —that is, $P_v = P_u \cup (u, v)$ when $v \in \operatorname{argmin}_{v \notin S} d'(v)$ and $(u, v) \in \operatorname{argmin}_{(u,v) \in E, u \in S} d(u) + w_{u,v}$.

Lemma 1. Consider the set S at any point in the algorithm's execution. For each $u \in S$, P_u is a shortest $s - u$ path.

(Proof of Correctness: When the algorithm terminates, S contains all nodes, thus by Lemma 1, we'll have found shortest paths from s to all nodes.)

Proof. By induction on $|S|$.

Base case ($|S| = 1$): $|S| = 1$, so $S = \{s\}$ and $d(s) = 0$.

Inductive Hypothesis: Suppose the claim holds for some $k \geq 1$.

Inductive Step ($|S| = k + 1$):

□

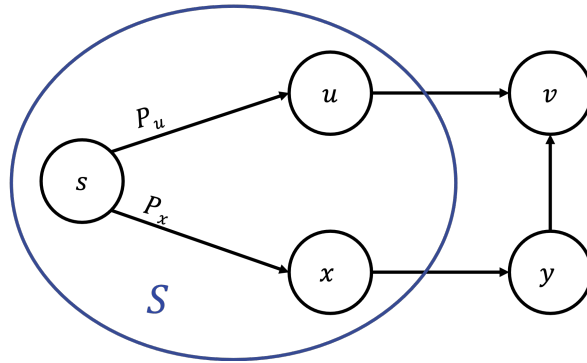


Figure 1: Illustration of the argument in the inductive step.

Dijkstra is really continuous/waterfilling BFS.

Greedy II: Interval Scheduling

Suppose you are given n jobs to schedule on a machine. Each job i (where $i \in \{1, \dots, n\}$) has a start time s_i and a finish time f_i . You would like to schedule *as many* jobs as possible given that the machine can only process one job at a time, and the jobs must run from their start time to finish time uninterrupted to be processed. That is, the machine cannot process two jobs that overlap.

What *greedy* algorithm should you use to schedule the jobs? By what metric is it greedy? (See **Step 2**.)

Prove that your algorithm is optimal by a **Greedy-Stays-Ahead** proof.

Step 1: Define your solutions. Describe the form your greedy solution takes, and what form some other solution takes (possibly the optimal solution). For example, let A be the solution constructed by the greedy algorithm, and let O be a (possibly optimal) solution.

Step 2: Find a measure. Find a *measure* by which greedy stays ahead of the other solution you chose to compare with. Let a_1, \dots, a_k be the first k measures of the greedy algorithm, and let o_1, \dots, o_m be the first m measures of the other solution ($m = k$ sometimes).

Step 3: Prove greedy stays ahead. Show that the partial solutions constructed by greedy are always just as good as the initial segments of your other solution, based on the measure you selected.

- For all indices $r \leq \min(k, m)$, prove (often by induction) that $a_r \geq o_r$ or that $a_r \leq o_r$, whichever the case may be. Don't forget to use your algorithm to help you argue the inductive step.

Step 4: Prove optimality. Prove that since greedy stays ahead of the other solution with respect to the measure you selected, then it is optimal.