Written Assignment 1 — Solutions

CS 440 September 18, 2025

Name: Yat Long Szeto BU ID: 90479281

Question 1: Shortest Path Composition

Proof

$$
\begin{aligned}
p^* &= p_1 \cdot p_2 && \text{split } a \to b \text{ path at } c \\
\operatorname{cost}(p^*) &= \operatorname{cost}(p_1) + \operatorname{cost}(p_2) && \text{path cost additivity} \\
\exists\, \tilde{p}_1 :\ \operatorname{cost}(\tilde{p}_1) &< \operatorname{cost}(p_1) && \text{assume } p_1 \text{ not shortest} \\
\operatorname{cost}(\tilde{p}_1 \cdot p_2) &= \operatorname{cost}(\tilde{p}_1) + \operatorname{cost}(p_2) && \text{concatenate paths} \\
&< \operatorname{cost}(p_1) + \operatorname{cost}(p_2) && \text{by assumption} \\
&= \operatorname{cost}(p^*) && \text{substitution}
\end{aligned}
$$

This contradicts the optimality of p^*. Therefore p_1 must be a shortest $a \to c$ path and there is no such $\tilde{p}_1$ exists. WLOG, replace the prefix p_1 by the suffix p_2 and c by b. The same reasoning shows that if p_2 were not shortest $c \to b$, then replacing it with a shorter path would also contradict the optimality of p^*. Hence p_2 is a shortest $c \to b$ path as well.

Question 2: Iterative Deepening Returns Shortest Paths

Proof

$$
\begin{aligned}
\ell(v) &= \min\{\text{length}(P) : P \text{ is an } s \to v \text{ path}\} && \text{define shortest distance} \\
\text{When IDDFS, } d < \ell(v) &\ \Rightarrow\ v \text{ not found} && \text{too shallow} \\
d = \ell(v) &\ \Rightarrow\ v \text{ reachable} && \text{path length fits} \\
\text{IDDFS finishes depth } d &= \ell(v) \text{ before } d+1 && \text{order of search} \\
v \text{ first discovered} &= \text{ at depth } \ell(v) && \text{cannot appear earlier}
\end{aligned}
$$

Therefore, the first time v is returned by Iterative Deepening is exactly at depth $\ell(v)$, and the path has the minimum number of edges.

Question 3: Diameter Bound

Proof

$$
\begin{array}{rr}
\text{Fix } u, v \in V,\ u \neq v. & \text{setup} \\
W = (v_0, \ldots, v_k),\ v_0 = u,\ v_k = v & \text{a } u \to v \text{ walk of minimum length} \\
\exists\, i < j:\ v_i = v_j \ \Rightarrow\ W' = (v_0, \ldots, v_i, v_{j+1}, \ldots, v_k) & \text{delete cycle} \\
\mathrm{len}(W') = k - (j - i) < k = \mathrm{len}(W) & \text{strictly shorter} \\
\Rightarrow\ \neg\exists\, i < j:\ v_i = v_j & \text{contradiction to minimality} \\
\Rightarrow\ W \text{ is simple} & \text{no repetitions} \\
\text{Let } P = W,\ |P| = m \text{ (edges)} \ \Rightarrow\ P \text{ visits } m+1 \text{ distinct vertices} & \text{path has one more vertex} \\
m + 1 \le |V| \ \Rightarrow\ m \le |V| - 1 & \text{counting bound} \\
\mathrm{dist}(u, v) = |P| \le |V| - 1 & \text{shortest path equals } |P| \\
\mathrm{diam}(G) = \max_{u \neq v} \mathrm{dist}(u, v) \le |V| - 1 & \text{take max}
\end{array}
$$

Question 4: Dijkstra's Algorithm and Shortest Paths

Notation. For vertices $u, v \in V$:

$$\delta(u, v) = \min\{\mathrm{cost}(P) : P \text{ is a path from } u \text{ to } v\}$$

denotes the true shortest-path cost. Dijkstra maintains tentative labels $d[v]$ which are upper bounds on $\delta(s, v)$. At each step, the algorithm extracts the vertex v with minimal $d[v]$ among unvisited vertices.

Answer

$$
\begin{array}{rr}
\forall v:\ d[v] \ge \delta(s, v) & \text{tentative labels are upper bounds} \\
\text{Suppose } d[v] > \delta(s, v) & \text{assume contradiction} \\
\exists P : s = x_0, \ldots, x_k = v,\ \mathrm{cost}(P) = \delta(s, v) & \text{true shortest path} \\
\text{Let } x_j = \text{first vertex of } P \text{ not yet settled} & \text{prefix in settled set } S \\
x_{j-1} \in S : d[x_{j-1}] = \delta(s, x_{j-1}) & \text{inductive assumption} \\
d[x_j] \le d[x_{j-1}] + w(x_{j-1}, x_j) & \text{relaxation} \\
= \delta(s, x_{j-1}) + w(x_{j-1}, x_j) & \\
= \delta(s, x_j) & \text{shortest-path property} \\
\Rightarrow\ d[x_j] \le \delta(s, x_j) < \delta(s, v) < d[v] & \text{contradiction}
\end{array}
$$

Thus $d[v] = \delta(s, v)$ when v is extracted. By induction over the extraction steps, every vertex is settled with its true shortest-path distance.

Conclusion. When Dijkstra returns a path from s to v, it is guaranteed to be the shortest path because: 1. vertices are processed in nondecreasing order of distance, 2. once extracted, $d[v] = \delta(s, v)$, 3. no unvisited vertex can have a smaller true distance.