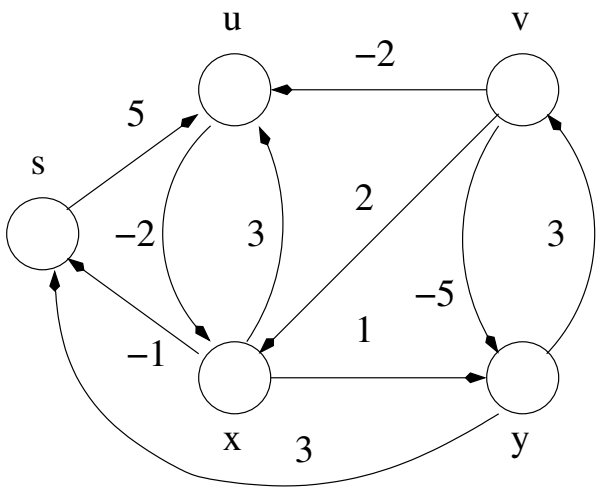
Computer Science 6901 (Fall 2026):
Problem-set #3
Answers

**1.** Consider the following edge-weighted directed graph:




**a)** Run Dijkstra's algorithm (p, 595) on the directed graph above using vertex $y$ as the source vertex. In the style of Figure 24.6 in the textbook, show the $d$ and $\pi$ values and the vertices in set $S$ after each iteration of the **while** loop.

**Answer:** See Table 1.

**b)** Run the Bellman-Ford algorithm (p, 588) on the directed graph above using vertex $y$ as the source vertex. Relax edges in lexicographic order in each pass, and in the style of Figure 24.4 on the textbook, show the $d$ and $\pi$ values after each pass. Finally, give the boolean value returned by the algorithm.

**Answer:** See Table 2. As it is the case for all edges $(a, b)$ that $d(b) \leq w(a, b) + d(a)$, the algorithm returns `true`.

**2.** Consider the following decision problems:

Dumbbell subgraph (DS)
*Input*: An undirected graph $G = (V, E)$ and two positive integers $k, l \geq 1$.
*Question*: Are there two cliques $C_1$ and $C_2$ and a simple path $P$ in $G$ such that $C_1$ and $C_2$ have $\geq k$ vertices apiece, $P$ has $\geq l$ edges, $P$ connects $C_1$ and $C_2$, the cliques and path

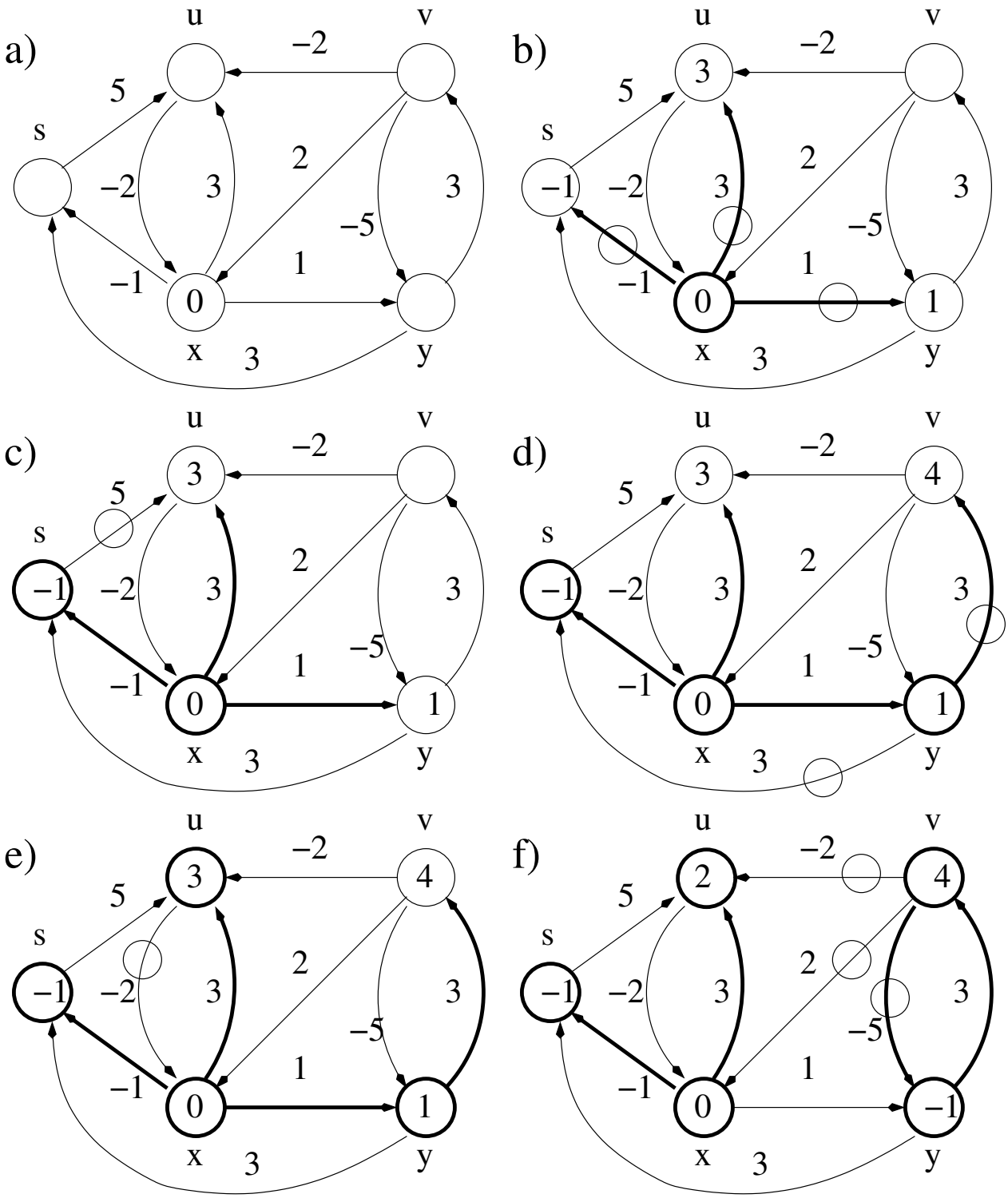



Table 1: Answer for Question #1(a). The shortest-path estimates are shown within the vertices; empty vertices have estimates equal to infinity ($\infty$). Bold edges indicate predecessor values. Bold vertices are in the set $S$ and regular vertices are in the priority queue $Q = V - S$. **(a)** The situation before the first execution of the **while** loop on lines 4–8. **(b)**–**(f)** The situation after each successive iteration of the **while** loop. Note that edges for which relaxation is attempted in each successive iteration of the **while** loop are marked with circles.

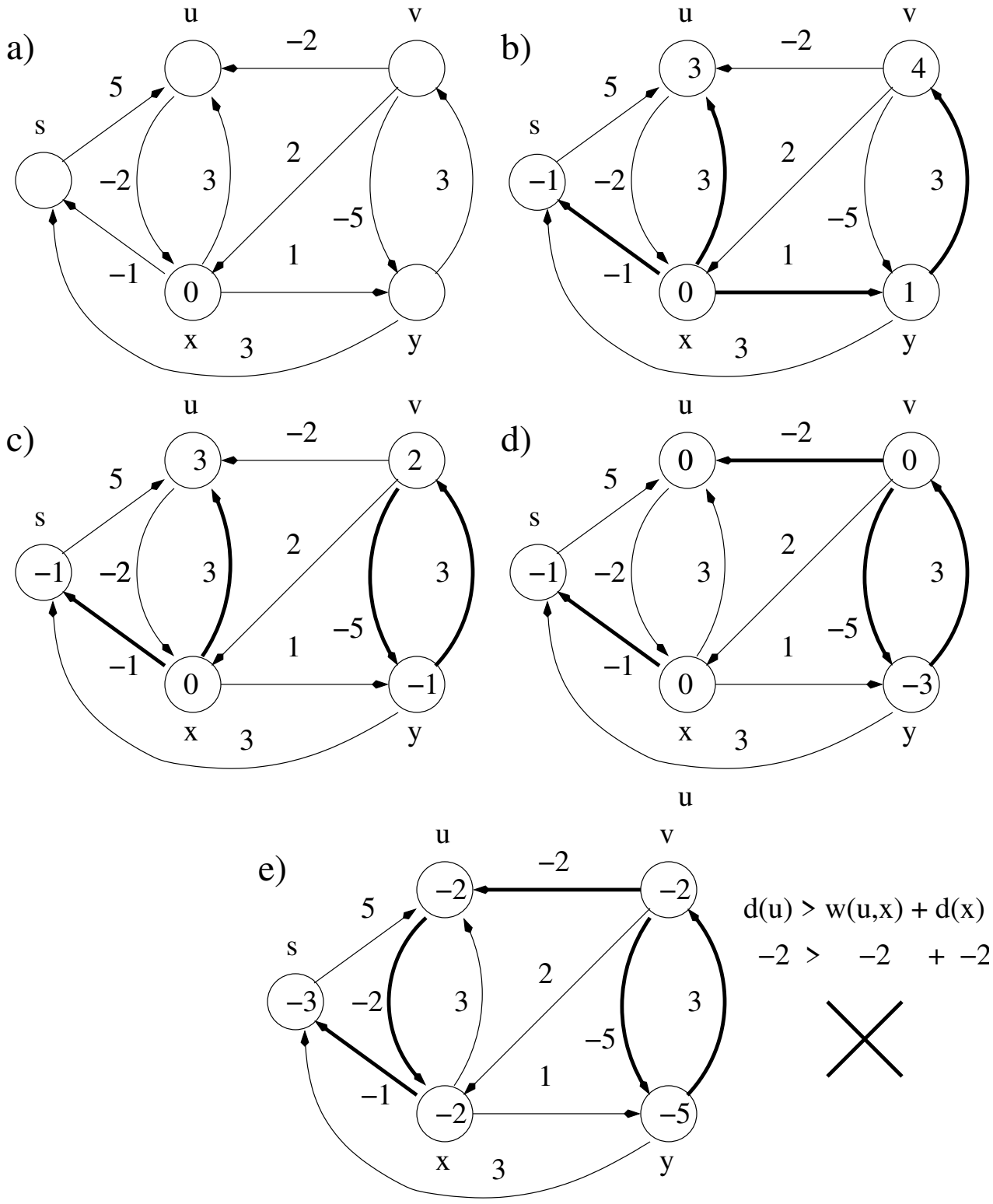



Table 2: Answer for Question #1(b). The shortest-path estimates are shown within the vertices; empty vertices have estimates equal to infinity ($\infty$). Bold edges indicate predecessor values. **(a)** The situation just before the first pass over the edges. **(b)**–**(e)** The situation after each successive pass over the edges.

do not have any edges in common, and the only vertices that $P$ shares with $C_1$ ($C_2$) is its connection-vertex?

Bounded-Weight Subset Cover (BWSSC)
*Input*: A set $I = \{i_1, \ldots, i_a\}$ of items, a set $R = \{r_1, \ldots, r_b\}$ of subsets of $I$, an integer-valued subset-weight function $w()$ such that for each $r_x \in R$, $w(r_x) > 0$, a subset $N \subseteq I$, and integers $0 < k_1 \leq k_2$.
*Question*: Is there is a subset $R' \subseteq R$ such that $\cup_{r \in R'} r = N$ and $k_1 \leq \sum_{r \in R'} w(r) \leq k_2$?

**a)** Prove that problem DS is $NP$-complete by (1) showing that this problem is in $NP$ and (2) giving a polynomial-time many-one reduction (algorithm + proof of correctness) to this problem from an $NP$-hard problem.

**Answer:** As any candidate solution has size $s$ such that $\frac{2k(k-1)}{2} \leq s \leq |V|$ and this solution can be checked in time polynomial in the input size (ensure all vertices are distinct and that the required edges exist between them), this problem is in $NP$. To show $NP$-hardness, we reduce from the following $NP$-complete problem (p. 983, textbook):

Hamiltonian Path (HP)
*Input*: An undirected graph $G$.
*Question*: Is there a simple path in $G$ that links all vertices in $G$?

Given and instance ($G$) of HP, we construct an instance $(G, k, l)$ of DS in which $k = 1$ and $l = |V| - 1$. Any solution to this instance of DS contains a simple path with at least $|V| - 1$ edges which must by definition include all vertices in $G$ and hence is a Hamiltonian path; conversely, any solution to the given instance of HP is a simple path in $G$ with $|V| - 1$ edges which connects two cliques of size $k = 1$, *i.e.*, the endpoints of the path. As the construction described above can also be done in polynomial time, this construction is a polynomial-time many-one reduction from HP to DS which establishes the $NP$-hardness of DS. As DS is $NP$-hard and also in $NP$, DS is $NP$-complete.

**b)** Prove that problem BWSSC is $NP$-complete by (1) showing that this problem is in $NP$ and (2) giving a polynomial-time many-one reduction (algorithm + proof of correctness) to this problem from an $NP$-hard problem.

**Answer:** As any candidate solution for an instance of BWSSC can be checked in time polynomial in the input size (ensure that the selected subset of subsets covers $N$ and that the sum of the weights of these selected subsets is between $k_1$ and $k_2$ inclusive), this problem is in $NP$. To show $NP$-hardness, we reduce from the following $NP$-complete problem (p. 1033–1034, textbook):

Set Cover (SC)
*Input*: A set $U = \{u_1, \ldots, u_n\}$ of items, a set $S = \{s_1, \ldots, s_m\}$ of subsets of $U$, and an integer $k \leq m$.

*Question*: Is there a subset $S' \subseteq S$ such that $|S| \leq k$ and $\cup_{s \in S'} s = U$?

Given an instance $\langle U, S, k \rangle$ of SC, we construct an instance $\langle I, R, w, N, k_1, k_2 \rangle$ of BWSSC such that $I = N = U$, $R = S$, $w(r) = 1$ for all $r \in R$, $k_1 = 0$, and $k_2 = k$. Observe that any solution to the constructed instance of BWSSC is also a solution to the given instance of SC, and vice versa. As the construction described above can be done in polynomial time, this construction is a polynomial-time many-one reduction from SC to BWSSC which establishes the $NP$-hardness of BWSSC. As BWSSC is $NP$-hard and also in $NP$, BWSSC is $NP$-complete.