Computer Science 6901 (Fall 2026):
Problem-set #2

1. Determine the optimal 0/1 knapsack-load for the set of items $U = \{1, 2, 3, 4, 5, 6\}$ and $B = 6$ using the algorithm given in Lecture #11 of the class notes. The sizes and values of the items in U are given in part (a) of Table 1. Show the filled-in dynamic programming value and backpointer matrices, the "backpointer path" in the backpointer matrix that gives an optimal knapsack-load, and the knapsack-load associated with that path.

Answer: The requested table is given in part (b) of Table 1. The optimal-value knapsack-load derived by the algorithm is $\{2, 3, 6\}$ with summed size 6 and summed value 11.

2. For the problem below, give a pseudocode algorithm and a parameterized asymptotic worst-case time complexity for that algorithm. Note that these algorithms must run in parameterized polynomial time, *i.e.*, all terms excluding the variables denoting the time complexities of used operations must be polynomials in the input size.

- Given a connected undirected graph $G = (V, E)$ and three non-overlapping vertex-subsets $S, I, F \subset V$, a **transit path** in G from $x \in S$ to $y \in F$ is a path in G from x to y that passes through at least one vertex in I. Given G, S, I, F, x, and y, compute the length of the shortest transit path (in terms of number of edges in the path) from x to y in G. You may use the following operations:
  - `SP(G, x, y)`: Returns the length of the shortest path in G between x and y.
  - `size(X)`: Returns number of vertices in vertex-set X.
  - `getVertex(X, i)`: Returns the ith vertex in vertex-set X, where $1 \leq i \leq size(X)$.

(a)

i	$s(i)$	$v(i)$
1	4	5
2	2	4
3	3	4
4	5	4
5	2	1
6	1	3

(b)

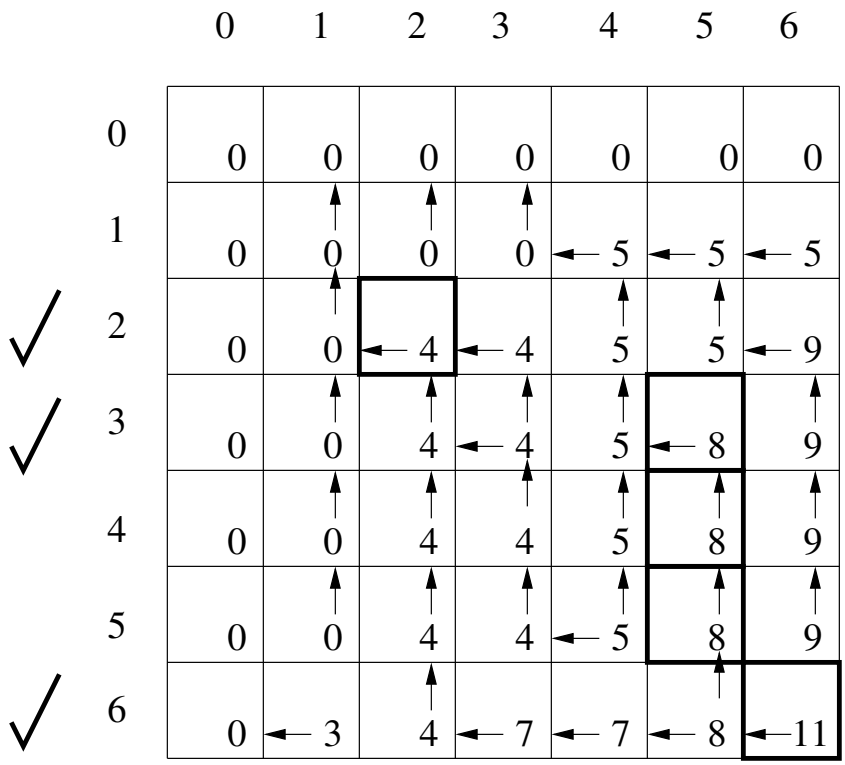


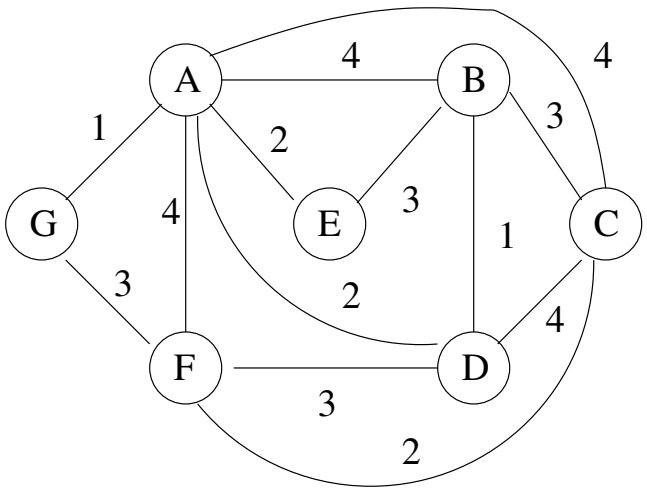
Table 1: Tables for Question #1. a) Input item size- and value-function specification. b) Answer.

Answer: One possible pseudocode for solving this problem is as follows:

```
xValid = yValid = false
mtpl = INFTY
n = size(S)
for i = 1 to n do
    if (getVertex(S. i) == x) then
        xValid = true
n = size(S)
for i = 1 to n do
    if (getVertex(F. i) == y) then
        yValid = true
if (xValid and yValid) then
    n = size(I)
    for i = 1 to n do
        z = getVertex(I, i)
        ctpl = SP(G, x, z) + SP(G, z, y)
 if (ctpl < mtpl) then
     mtpl = ctpl
print mtpl
```

As the sizes of S, I, and F are all less than the number of vertices in G, all `for`-loops execute $O(|V|)$ times. Hence, the parameterized asymptotic worst-case running time of this algorithm is $O(3T(size) + 3|V|T(getVertex) + 2|V|T(SP)) = O(T(size) + |V|(T(getVertex) + T(SP)))$.

3. Consider the following weighted undirected graph:



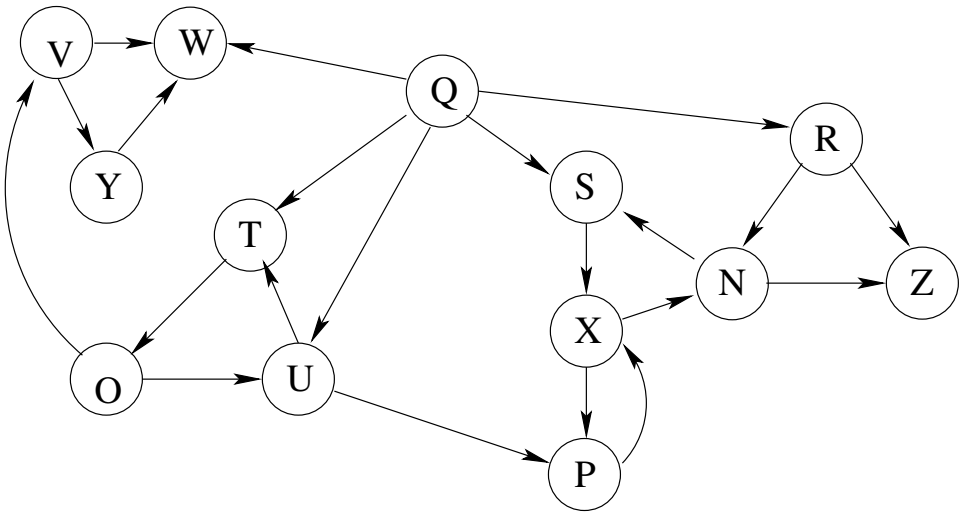
a) Show how Kruskal's minimum spanning tree algorithm works on this graph. Give the graph at the end of the execution of the algorithm on page 569 of the textbook, with all tree-edges and the order in which each tree-edge was added clearly marked.

Answer: See part (a) of Figure 1.

b) Show how Prim's minimum spanning tree algorithm works on this graph relative to root-vertex r = G. Give the graph at the end of the execution of the algorithm on page 572 of the textbook, with all tree-edges and the order in which each tree-edge was added clearly marked.

Answer: See part (b) of Figure 1.

4. Consider the following directed graph:



Assume that the algorithms below consider vertices in alphabetical order and that each adjacency list is ordered alphabetically.

a) Show how breadth-first search works on this graph. Give the graph at the end of the execution of the BFS algorithm on page 532 of the textbook when the search is started at vertex X, with the d- and π-values for all vertices as well as all BFS-search tree edges clearly marked.

Answer: See part (a) of Figure 2.

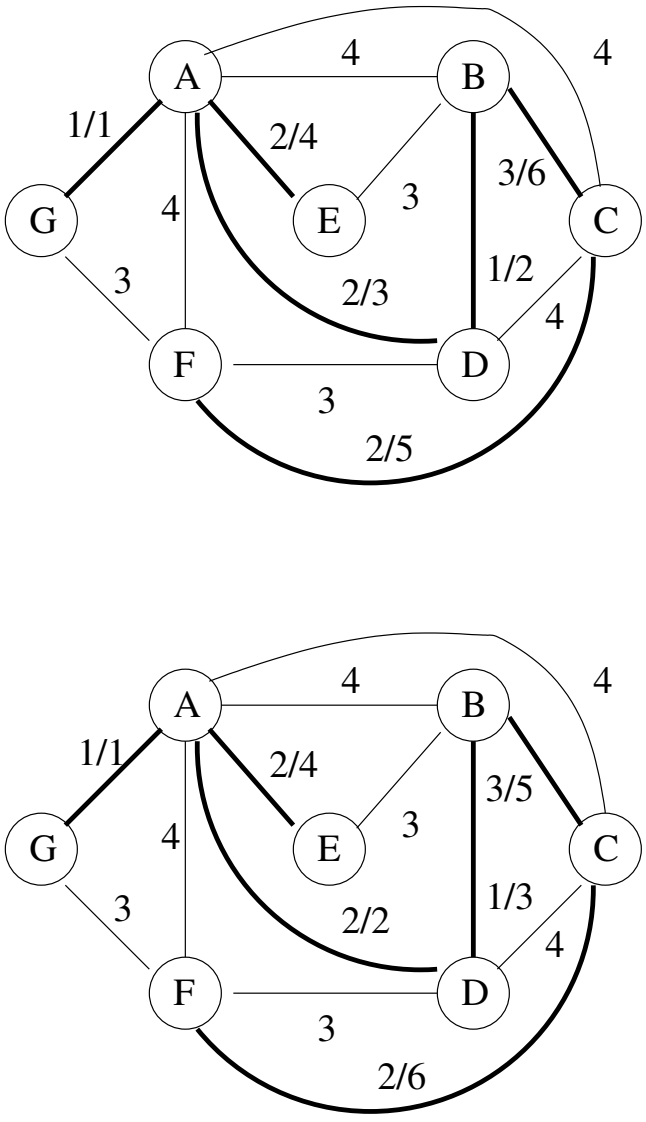


Figure 1: Answers for Question #3(a-b). Edges that are part of the minimum spanning tree are in bold face, and the order in which each tree-edge was added is noted as a number after its weight.

b) Redo part (a) above with the BFS algorithm starting at vertex Q.

Answer: See part (b) of Figure 2.

c) Show how depth-first search works on this graph. Give the graph at the end of the execution of the DFS algorithm on page 541 of the textbook, with the d-, f-, and π-values of all vertices as well as the types of all edges clearly marked.

Answer: See Figure 3.

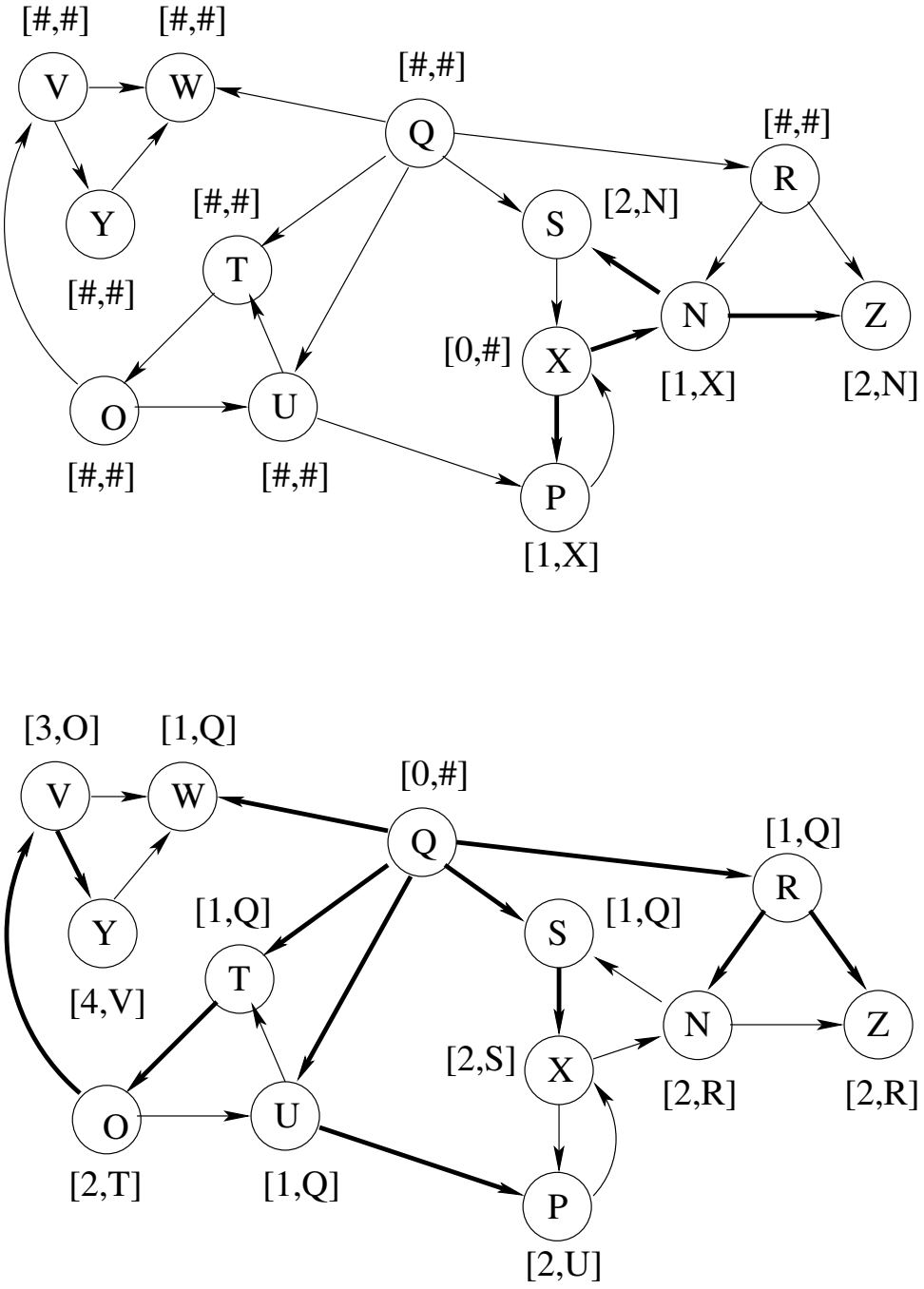


Figure 2: Answers for Question #4(a-b). For each vertex, the required d- and π-values x and y are indicated in the diagram using string "$[x, y]$". Note that any vertex that is not reachable from the start vertex has the associated string "$[\#, \#]$" (∞ and **nil**, respectively).

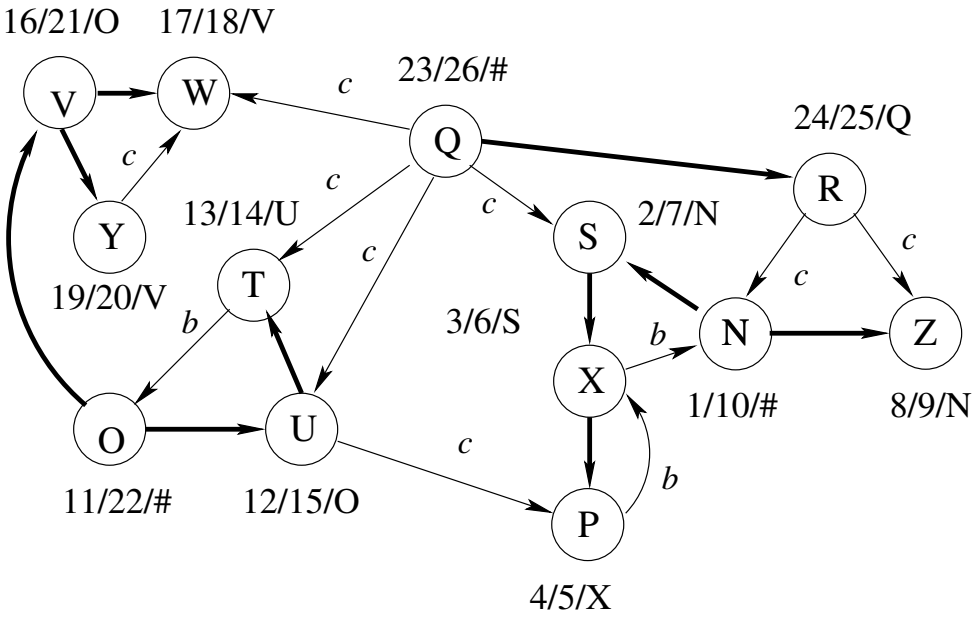


Figure 3: Answer for Question #4(c). For each vertex, the required d-, f-, and π-values x, y. and z are indicated in the diagram using string "$x/y/z$". Note that a π-value of **nil** is indicated by a hash-mark ("#"). All tree edges are in bold, and back, forward, and cross edges are annotated with the characters "b", "f", and "c", respectively.