**1.** Consider an instance of the single-source shortest path problem with $n$ vertices, $m$ edges, a source vertex $s$, and no negative cycles. Which of the following is true?

1. For every vertex $v$ reachable from source $s$, there is a shortest $s - v$ path with at most $n - 1$ edges.
2. For every vertex $v$ reachable from the source $s$, there is a shortest $s - v$ path with at most $n$ edges.
3. For every vertex $v$ reachable from the source $s$, there is a shortest $s - v$ path with at most $m$ edges.
4. There is no finite upper bound on the fewest number of edges in a shortest $s - v$ path.

(1) is the correct answer. If we had a shortest $s - v$ path with more than $n - 1$ edges then we have a non-negative cycle, removing this cycle would produce a $s - v$ path of at most $n - 1$ edges that is better than the previous solution.

**2** How many candiates are there for an optimal solution to a sub-problem with the destination $v$?

1. 2
2. $1 + the\, in - degree\, of\, v$
3. $1 + the\, out - degree\, of\, v$
4. $n$

(2) is the correct answer. The first case of the recurrence relation gives us a one candidate and the second case gives uses all neighbors of $v$ that have $v$ as a head node, which is the $in - degree\, of\, v$. Hence we have $1 + the\, in - degree\, of\; - v$.

**3** What's the running time of the Bellman-Ford algorithm, as a function of $m$ and $n$?

1. $O(n^2)$
2. $O(m \cdot n)$
3. $O(n^3)$
4. $O(m \cdot n^2)$

(2) is the correct answer. Given an adjanceny list representation we have $V - 1$ iterations which is the number of sub-problems, then we choose all $E$ edges to determine if we could relax it. Thus we have $O(V \cdot E)$. Note that when $E = O(V^2)$ this becomes $O(V^3)$ showing that $O(V \cdot E)$ is the tightest correct bound.

**4** How many invocatiosn of a single-source shortest path subroutine are needed to solve the all-pairs shortest path problem?

1. 1
2. $n - 1$
3. $n$
4. $n^2$

(3) is the correct answer. We iterate through all $n$ vertices and compute the single-shortest path using bellman-ford algorithm.