1. Suppose that we are doing Dijkstra’s Algorithm on vertex set $V = \{1, \ldots, 500\}$ with source vertex $s = 1$ and at some time we have $S = \{1, \ldots, 100\}$. What is the interpretation of $\pi[v], d[v]$ for $v \in S$?

**Solution:**
$d[v]$ is the minimal cost of a path from $s$ to $v$ and $\pi[v]$ will be the vertex just before $v$ on that path.

What is the interpretation of $\pi[v], d[v]$ for $v \not\in S$?

**Solution:**
$d[v]$ is the minimal cost of a path $s, v_1, \ldots, v_j, v$ where all the $v_1, \ldots, v_j \in S$. $\pi[v]$ will be the vertex just before $v$ in this path, here $v_j$.

Which $v$ will have $\pi[v] = NIL$ at this time.

**Solution:** Those $v$ for which there is no directed edge from any vertex in $S$ to $v$.

For those $v$ what will be $d[v]$?

**Solution:** Infinity

2. Suppose, as with Dijkstra’s Algorithm, the input is a directed graph, $G$, a source vertex $s$, and a weight function $w$. But now further assume that the weight function only takes on the values one and two. Modify Dijkstra’s algorithm – replacing the MIN-HEAP with a more suitable data structure – so that the total time is $O(E + V)$.

**Solution:** There are a number of approaches here. Start with $S = \{s\}$ and sets ONE (those $v$ adjacent to $s$ via an edge of weight one), TWO (those $v$ adjacent to $s$ via an edge of weight two), and INFTY (those not adjacent to $s$). Now rather than going one vertex at a time $S$ will be all points at weighted distance $d$ or less from $s$ and ONE, TWO will be those $v$ adjacent to a $v \in S$ be an edge of weight one or two (if both, one). Suppose, first, ONE is empty. Add all points $v \in TWO$ to $S$. Each new (not in $S$) neighbor of each such $v$ is put in ONE or TWO depending on its weight. Suppose, otherwise, ONE is not empty. Add all points $v \in ONE$ to $S$. All points of TWO move to ONE. Each new (not in $S$) neighbor of each such $v$ is put in ONE or TWO depending on its weight. **Alternate Approach:** Whenever $w(x, y) = 2$ create a new vertex $z$, delete edge $(x, y)$ and add edges $(x, z), (z, y)$, each of weight one. Now all the weights are one so that BFS will give the distances.

3. Let $G$ be a DAG on vertices $1, \ldots, n$ and suppose we are given that the ordering $1 \cdots n$ is a Topological Sort. Let $COUNT[i,j]$ denote the
number of paths from $i$ to $j$. Let $s$, a “source vertex” be given. Give an efficient algorithm (appropriately modifying the methods of §24.1) to find $\text{COUNT}[s,j]$ for all $j$.

**Solution:** Let's assume $s = 1$ (we can ignore the earlier vertices, if any) and write $\text{COUNT}[j]$ for $\text{COUNT}[1,j]$. We set $\text{COUNT}[1] = 1$.

The key is that $\text{COUNT}[1,j]$ is the sum, over all $i < j$ with $i,j$ a directed edge, of $\text{COUNT}[1,i]$. Why? Well, every path from 1 to $j$ will have a unique penultimate point $i < j$ and given $i$ there will be precisely $\text{COUNT}[i]$ such paths. One way to implement this is to make a reverse adjacency list, create for every $j$ a list $\text{Adjrev}[j]$ of those $i$ with a directed edge from $i$ to $j$. This can be done in time $O(E)$ by going through the original adjacency lists and when $j \in \text{Adj}[i]$ adding $i$ to $\text{Adjrev}[j]$. Then we can implement this sum. The total time (assuming addition takes unit time) is $O(E)$. 