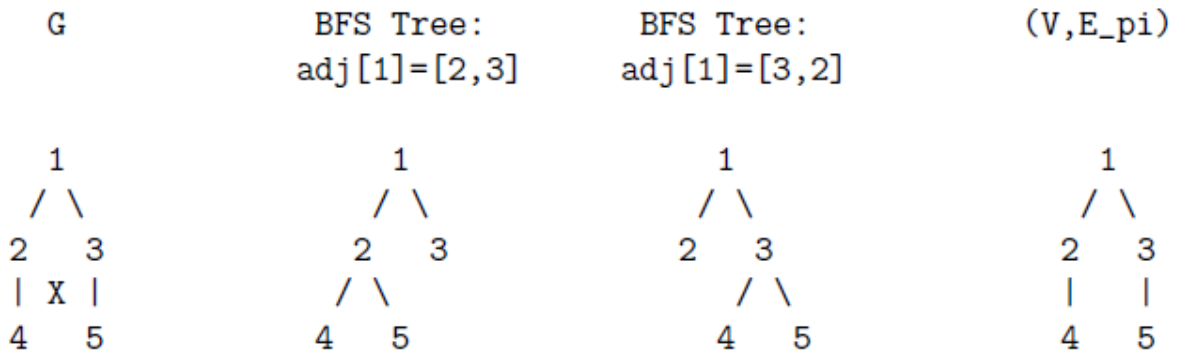
1.

Let $G = (V, E)$ with $V = \{1, 2, 3, 4, 5\}$, $E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}\}$ and $s = 1$.

Then there are two possible breadth–first search trees produced by BFS$((V, E), s)$, depending on the first vertex in the adjacency list of vertex 1.



2.

Answer: Let the weight function for H be w. Let e^* be the edge in H of minimum weight and denote $w^* = w(e^*)$. We are assuming that $w^* < 0$. We define a new weight function w' for H: for every edge $e \in E$, let $w'(e) = w(e) - w^*$. Then $w'(e) \geq 0$ for all e.

Now compute $\mathbf{A}(H, w')$ (i.e., run $\mathbf{A}$ on the graph H with weight function w'). The output of $\mathbf{A}$ is an integer t', which is the weight of the minimum spanning tree of H with weight function w'. Let $t = t' + (n-1)w^*$, where $n = |V|$. We claim that t is the weight of a minimum spanning tree for (H, w).

Justification: Let T' be an MST for (H, w') (the algorithm $\mathbf{A}$ does not find T', but T' exists nevertheless). Using weight function w', the weight of T' is t'. Also, let t_0 be the weight of an MST for (H, w). The weight of T' using weight function w is $t' + (n-1)w^*$, since T' contains $n-1$ edges and $w(e) = w'(e) + w^*$ for every $e \in E$. Hence $t_0 \leq t' + (n-1)w^*$. Conversely, let T be an MST for (H, w); then the weight of T is t_0. The weight of T using weight function w' is $t_0 - (n-1)w^*$ since T contains $n-1$ edges and $w'(e) = w(e) - w^*$ for every $e \in E$. Hence $t' \leq t_0 - (n-1)w^*$. Finally, since $t_0 \leq t' + (n-1)w^*$ and $t' \leq t_0 - (n-1)w^*$, we have $t_0 = t' + (n-1)w^*$. Therefore $t_0 = t$, as desired.

3a)

Answer: After doing a topological sort, assume the vertices are ordered $x_1, \ldots, x_n$ and the source vertex $u_0 = x_1$. Then we count the paths using the following algorithm:

```
N[x_1] = 1
for i from 2 to n do N[x_i] = 0
for i from 1 to n do
  for j in Adj[i] do
    N[x_j] = N[x_j] + N[x_i]
```

main idea: Whenever we process an edge $x_i x_j$, this creates $N[x_i]$ "additional" (x_1, x_j) paths, because we can take any existing (x_1, x_i) path and extend it to an (x_1, x_j) path by adding the edge $x_i x_j$. So we update $N[x_j]$ by adding the vale of $N[x_i]$ to it.

complexity: The topological sort has complexity $O(n + m)$. In the second part of the algorithm, we loop through every vertex and every edge, spending $O(1)$ time at each. So the complexity of this step is also $O(n + m)$.

3b)

Answer: Let $\mathcal{P}(i, k)$ denote the set of (x_1, x_k) paths having vertices in $\{x_1, \ldots, x_i\} \cup \{x_k\}$. The loop invariant is the following:

At the end of iteration i of the main `for` loop, $N[x_k] = |\mathcal{P}(i, k)|$.

We prove this statement by induction on i.

base case: $i = 1$ is the base case. At the end of iteration 1, $N[x_1] = 1$, which correct. For $k > 1$, we have $N[x_k] = 1$ if $x_1 x_k$ is an edge, and $N[x_k] = 0$, otherwise.

induction: Assume the loop invariant holds at the end of iteration $i - 1$, and consider the situation at the end of iteration i. At the end of iteration $i - 1$, we have $N[x_k] = |\mathcal{P}(i - 1, k)|$, by induction. We now consider a path $P \in \mathcal{P}(i, k)$. We identify two cases:

case 1: $x_i \notin P$. Then $P \in \mathcal{P}(i-1,k)$ and we counted P in the value of $N[x_k]$ at the end of iteration $i-1$.

case 2: $x_i \in P$. Then, because the vertices are topologically ordered, x_i must be the vertex preceding x_k in P, and $x_i x_k$ must be an edge. The number of (x_1, x_k) paths having vertices in $\{x_1, \ldots, x_i\} \cup \{x_k\}$, such that x_i is one of the vertices in the path, is equal to $N[x_i]$, by induction (note that the value of $N[x_i]$ does not change after iteration $i-1$, because we have a topological ordering of the vertices).

The new value of $N[x_k]$ at the end of iteration i is equal to sum of the old value of $N[x_k]$ and $N[x_i]$, which equals $|\mathcal{P}(i,k)|$.

By induction, the loop invariant holds for $i = 1, \ldots, n$. Therefore, at the end of the algorithm, we have that $N[x_k] = |\mathcal{P}(n,k)|$ for all k, and thus $N[x_k]$ is equal to the total number of (x_1, x_k)-paths, as desired.

4.

If $u = v$ the problem is trivial, so assume that $u \neq v$. Let $V = \{v_1, v_2, \ldots, v_n\}$ with $u = v_1$ and $v = v_2$. We will construct a directed graph $G' = (V', E')$ on $24n+1$ vertices and $48|E| + 24n + 24$ edges such that a fastest route from city u to v corresponds to a shortest (weight) path from vertex $v_{1,0}$ to v_Ω in G', and vice versa. In what follows, indices i and h should be assumed to run over their full range: $i \in [1, n]$ and $h \in [0, 23]$.

Let V' consist of $24n$ vertices of the form $v_{i,h}$ (corresponding to city i at hour h) plus one sink vertex v_Ω. For each $e = \{v_a, v_b\} \in E$ include edges in E' of the form $(v_{a,h}, v_{b,(h+D[e,h]) \bmod 24})$ and $(v_{b,h}, v_{a,(h+D[e,h]) \bmod 24})$ with weight $D[e,h]$, corresponding to the possibility of travelling between city a and b, starting at one of the cities at hour h and arriving at the other at hour $(h + D[e,h]) \bmod 24$. Also include in E' edges of the form $(v_{i,h}, v_{i,(h+1) \bmod 24})$ with weight one, corresponding to waiting at city i from hour h to $(h+1) \bmod 24$. Finally, include 24 edges in E' with weight zero of the form $(v_{2,h}, v_\Omega)$.

5.

Here is the dynamic programming table:

	1	2	3	4	5	6
1	0	0	0	0	5	5
2	0	4	4	4	5	5
3	0	4	4	7	7	7
4	0	4	4	7	7	9
5	1	4	4	7	7	9

Column headings are the weight level. Row headings are the item number.

Note they may have an extra row of 0's at the top and an extra column of 0's to the left if they find that helpful they don't lose any marks for the extra 0's

To read off the answer:

- (5,6) and (4,6) are the same value 9 so item 5 was not selected. Go up one row.
- (4,6) and (3,6) are different so item 4 was selected. Go up one row and over 2 (the weight of item 4).
- (3,4) and (2,4) are different so item 3 was selected. Go up one row and over 2 (the weight of item 3).
- (2,2) and (1,2) are different so item 2 was selected. Go up one row and over 2 (the weight of item 2).
- You are at zero weight so item 1 was not selected.