Problem 1. *Suppose $G = (V,E)$ is a directed acyclic graph represented by adjacency lists. Devise a linear time algorithm that, given such a G, returns the length of the longest path in G. Prove your algorithm runs in $O(|V|+|E|)$-time.*

SOLUTION: Note that the longest path from vertex u is $\max(\{1+\text{the length of the longest path from } v : (u,v) \in E\})$ where, by convention, $\max(\emptyset) = 0$. Now, to solve the problem, first do a topological sort of G and number G's vertices 1 through n where $i < j$ implies that vertex i comes before vertex j in the topological sort. Next compute:

```
1.  for i = n, n−1, ..., 1 do
2.      longest[i] ← 0
3.      for each j adjacent to i do     // so (i, j) ∈ E
4.          longest[i] ← max(longest[i], 1 + longest[j])
5.  return max({longest[i] : i = 1, ..., n})
```

Since we are considering vertices in reverse-top-sort order, we know that in line 4, each $longest[j]$ has its correct value, hence when we are done with the for-loop of lines 3–4, $longest[i]$ has its correct value.
Runtime analysis: Line 2 is done once for each $i \in V$, line 4 is done once for the $(i,j) \in E$ and line 5 takes $\Theta(n)$ time. Hence, the entirety takes $\Theta(|V|+|E|)$-time.

Problem 2. *Suppose $G = (V,E)$ is a directed graph represented by adjacency lists. Devise a linear time algorithm that, given such a G, returns a list of all the source vertices of G. Prove your algorithm runs in $O(|V|+|E|)$-time.*
SOLUTION: The *in-degree* of a vertex v = the number of edges with end-point v. So, v is a source iff $in\text{-}degree(v) = 0$.
Now, we can compute in-degrees and identify sources as follows

```
1.  for each u ∈ V do inDegree[u] ← 0;
2.  for each (u, v) ∈ E do
3.      inDegree[v] ← inDegree[v] + 1;
4.  return [ v : v ∈ V, inDegree[v] = 0].
```

Clearly, this is $\Theta(|V|+|E|)$.

Problem 3. DPV Problem 3.18
Starting at the root vertex do a DFS with pre and post numbering. Then: u is an ancestor of v in the tree $\iff$ Corrected

$$(pre[u], post[u]) \supset (pre[v], post[v]) \qquad (1)$$

and we can test (1) in constant time.

Problem 4. DPV Problem 3.22
In the case where we know that G is a directed acyclic graph, there is such a vertex in G if and only if G has exactly one source vertex. Problem 2 showed how to determine the source vertice of a dag in linear time.
In the case where G is not necessarily a dag, compute the strongly connected component dag of G (which takes linear time) and then test if that dag has just one source.

Problem 5. DPV Problem 3.23
This is a variation of what we did for Problem 1. First do a topological sort of G. Remove all vertices that are either before s or after t in the topological-sort (because they cannot be in any path from s to t). Number the remaining vertices 1 through n in topological sort order. (So $s = 1$ and $t = n$.) Next compute:

```
1.  paths[n] ← 1
2.  for i = n − 1, ..., 1 do
3.      paths[i] ← 0
4.      for each j adjacent to i do     // so (i, j) ∈ E
5.          paths[i] ← paths[i] + paths[j]
6.  return paths[1]
```

Problem 6. DPV Problem 4.1

(a) Here is the table. Corrected

Step	A	B	C	D	E	F	G	H
init	0	∞	∞	∞	∞	∞	∞	∞
1	**0**	1	∞	∞	4	8	∞	∞
2	0	**1**	3	∞	4	7	7	∞
3	0	1	**3**	4	4	7	5	∞
4	0	1	3	**4**	4	7	5	8
5	0	1	3	4	**4**	7	5	8
6	0	1	3	4	4	6	**5**	6
7	0	1	3	4	4	**6**	5	6
8	0	1	3	4	4	6	5	**6**

(b) Here is the tree. Solid edges make up the tree; edges not in the tree are dashed.