1. For each graph, show the “DFS forest” resulting from an execution of DFS. Whenever there is a choice of vertices, choose the one that is alphabetically first. Identify the cross, forward, and back edges, and label each vertex with its pre and post number.

![Graph](image)

2. Run the DFS-based topological sort algorithm on the following graph. Whenever there is a choice of vertices, choose the one that is alphabetically first. Show the “DFS forest”, including pre and post numbers. Give the resulting topological ordering of the vertices.

![Graph](image)

3. In class and in the text, we discussed another topological sorting algorithm. On a high level, for a graph $G$ with vertices $V$ and edges $E$, it works like this:

```java
while (V not empty) {
    find a source vertex $v \in V$
    output $v$
    remove $v$ from $V$ and all edges of the form $v \rightarrow w$ from $E$
}
```

(a) Illustrate the execution of this algorithm on the graph in the previous exercise — use the usual alphabetical rule to break ties.

(b) Show how to implement this algorithm in linear time, i.e., time $O(|V| + |E|)$, assuming a sparse graph representation for $G$. Give a detailed description of your algorithm and its data structures. What does your algorithm do if $G$ contains a cycle?

4. Consider the sparse representation of a graph $G$ with vertices $V$ and edges $E$. Assume the vertices are numbered $1 \ldots n$. Then the graph is represented as an array $A[1 \ldots n]$ of lists, where $A[i]$ is a list of all successors of $i$ in $G$, i.e., $A[i]$ is a list of all vertices $j$ such that $i \rightarrow j$ is an edge in $G$. Normally, we don’t care how the elements in the successor lists are ordered, but suppose we insist that each successor list is ordered in ascending order. Let is call this representation the sorted sparse representation of $G$.

Give a linear time algorithm (i.e., one that runs in time $O(|V| + |E|)$) that takes as input a graph $G$ in sorted sparse representation, and produces as output the reverse graph $G^R$ in sorted sparse representation.

Hint: think about some of the sorting algorithms we discussed in class before the midterm.
5. Most profitable path. The input is a directed acyclic graph $G$, with vertices $V$ and edges $E$. The input also includes an array $P$ indexed by $v \in V$, where each $P[v]$ is a number (positive, negative, or zero), which represents the “profit” or “benefit” associated with visiting vertex $v$. For a path $v_0 \to v_1 \to \cdots \to v_k$ in $G$, we define its profit to be $\sum_{i=0}^{k} P[v_i]$, i.e., the profit associated with visiting each vertex in the path. Note that since $G$ is assumed acyclic, it is not possible to visit the same vertex twice.

Design and analyze an algorithm that finds a most profitable path in $G$. Your algorithm should output not just the profit of this path, but also the path itself. It should run in linear time (i.e., $O(|V| + |E|)$).

The following is for honors students only.

H1. An undirected graph is called 2-colorable if we can assign each vertex one of two colors, say red and blue, such that no two adjacent vertex are assigned the same color.

Give a linear time algorithm for this problem: your algorithm should determine if a valid assignment exists, and if so, output that assignment.

Hint: modify DFS.