Acknowledgments. Acknowledge all the sources you used to complete the assignment. DO NOT COPY! Please read http://www.student.cs.uwaterloo.ca/~cs341/ for general instructions and policies.

For all algorithm design questions, you must give the algorithm, argue the correctness, and analyze time complexity.

1 Graph algorithms [10 marks]

(i) Give an example of a connected, weighted, undirected graph $G$ and a start vertex $s$ such that neither the DFS spanning tree nor the BFS spanning tree of $G$ from $s$ is a minimum weight spanning tree of $G$, regardless of how the adjacency lists are ordered. Justify the correctness of your counter-example.

(ii) Prove that if the BFS and DFS spanning trees of a connected undirected graph $G$ from the same start vertex $s$ are equal to each other, then $G$ is a tree.

2 Minimum spanning trees [10 marks]

Prove or disprove each of the following statements, where in each case $G = (V, E)$ is a connected undirected weighted graph with $n$ vertices and $m$ edges.

(i) If $G$ has $m \geq n$ edges and a unique heaviest edge $e$, then $e$ is not part of any minimum spanning tree of $G$.

(ii) If $G$ has $m \geq n$ edges and a unique lightest edge $e$, then $e$ is part of every minimum spanning tree of $G$.

(iii) If $e$ is a maximal weight edge of a cycle of $G$, then there is a minimum spanning tree of $G$ that does not include $e$.

(iv) Prim’s algorithm returns a minimum spanning tree of $G$ even when the edge weights can be either positive or negative.
3 Shortest paths [10 marks]

A vertex-and-edge-weighted graph is a directed graph $G = (V, E)$ where each vertex $v \in V$ has a cost $c(v)$ and every edge $e \in E$ has a weight $w(e)$. The length of a path in $G$ is the sum of the weights of the edges in the path and the cost of the vertices in the path. In the Single-Source Shortest Path (SSSP) problem on vertex-and-edge-weighted graphs, we are given $G$ and a source vertex $s$ and we want to determine the minimum length of a path from $s$ to $v$ for every $v \in V$. (Note that the minimum length of a path from $s$ to $s$ is $c(s)$.)

Solve the SSSP problem on vertex-and-edge-weighted graphs in the special case where all the weights and the costs are positive integers.

4 Water puzzles [10 marks]

You are given three water jugs. The first one can contain 4L of water and is initially full, the second one can contain 7L and is also initially full, and the third one contains up to 10L of water but is initially empty. You can move the water between the jugs by pouring from one to the other until either (a) the jug being filled becomes completely full, or (b) the jug you are pouring from becomes empty. Can you pour the water between the containers in a way that you end up with exactly 2L of water in the first container?

(i) Model the question above as a graph problem. Your answer should include a specific directed graph, a description of what the vertices and the edges of your graph represent, and the problem that you want to solve on this graph to get the answer to the original question, as well as the answer itself.

(ii) What graph algorithm lets you determine not just whether it is possible to end up with 2L of water in the first container, but the minimum number of pours needed to end up in this state (if it is reachable)?

(iii) Let’s say each jug pours 1L of water per minute and we now want to know how quickly we can end up with 2L of water in the first container (if that is possible), how do we need to modify our graph representation and what algorithm do we want to run on the graph to solve this problem?

Note. All the answers in this question should involve reductions to algorithms that we saw in class, so you don’t need to include proofs of correctness or time complexity analyses in your answer. (But you do need to explain why the reduction is the right one.)
5 Programming question [20 marks]

Alice and Bob plan to run a Dynamic Relay Race together. In this race, they must go from the startpoint 0 to the finish line \( n \) by travelling between checkpoints \( 0 \rightarrow 1 \rightarrow 2 \rightarrow \cdots \rightarrow n \) in order. Each segment \( i \rightarrow i + 1 \) in the race is run by either Alice or Bob. The teammate who is not running is brought to the next checkpoint to wait for the runner to finish the segment. The rules of the race state that both Alice and Bob need to run exactly \( n/2 \) of the segments. (The organizers guarantee that \( n \) is always even.)

Alice and Bob know exactly how fast they can run each segment: Alice takes \( a_i \) minutes to cover the segment \( i \rightarrow i + 1 \), Bob takes \( b_i \) minutes to run the same segment.\(^1\) And for every checkpoint \( 1 \leq i < n \), there is a fixed penalty of \( p_i \) minutes incurred if they switch places (i.e., if Alice ran the segment \( i - 1 \rightarrow i \) and Bob runs segment \( i \rightarrow i + 1 \) or vice-versa). Their goal is to determine who should run each segment to finish the Dynamic Relay Race as quickly as possible.

(i) Design a dynamic programming algorithm with time complexity \( O(n^2) \) that finds the minimum time required for Alice and Bob to run the Dynamic Relay Race. As part of your algorithm description, you should include a precise definition of the subproblems solved by the algorithm as well as the recurrence relation and base case(s) used to solve the subproblems. (As usual, you also need to provide a proof of correctness and time complexity analysis.)

(ii) Submit an implementation of your solution. The input consists of \( 3n \) nonnegative integers separated by spaces or linebreaks:

\[
n \ a_0 \ a_1 \ a_2 \ \cdots \ \ a_{n-1} \ b_0 \ b_1 \ b_2 \ \cdots \ \ b_{n-1} \ p_1 \ p_2 \ \cdots \ p_{n-1}
\]

The output consists of two lines. The first line gives the fastest race time that Alice and Bob can achieve. The next line is a single string of length \( n \) consisting of the characters A and B, where the \( i \)th character indicates whether Alice or Bob runs the segment \( i \rightarrow i + 1 \) of the race for each \( 0 \leq i \leq n - 1 \) in an assignment of the segments that lets them achieve their fastest possible time.

For example, on input

\[
4 \ 35 \ 12 \ 46 \ 110 \ 53 \ 21 \ 36 \ 153 \ 10 \ 45 \ 2
\]

the valid solution is the output

\[
214
\]

\[
ABBA
\]

\(^1\)Alice and Bob are in fantastic shape, so that their time to run a segment is the same no matter how many segments they have already run previously.