Assignment 4 (due March 14th Tuesday noon)

Please first read the instructions at https://www.student.cs.uwaterloo.ca/~cs341/#assignments.

1. [20 marks]

Given a set of numbers \( a = \{a_1, \ldots, a_n\} \) with positive sum \( s = \sum_{i=1}^{n} a_i > 0 \), we want to partition the numbers into two groups of equal size (assuming \( n \) is even) such that the sum within each group is (strictly) positive. Note that for this job to be feasible we at least need \( s \geq 2 \) (necessary but not sufficient). Give a dynamic programming algorithm to decide if such a partition exists, and if it does, use backtracking to find such a partition. Describe your base case, recursion, and pseudo-code, and analyze the running time.

**Example.** Suppose we have \( n = 4 \) and \( a = \{10, -14, 20, -6\} \). Then \( s = 10 \), and we can indeed find such a partition of equal size: \( \{10, -6\} \) and \( \{-14, 20\} \), with positive sum 4 and 6, respectively. On the other hand, if \( a = \{5, -14, 20, -6\} \), then \( s = 5 \) but it is impossible to find such a partition of equal size.

[Hint: Equivalently, you can reformulate the problem as choosing \( n/2 \) elements from the set of \( n \) numbers, such that their sum is exactly a number between 1 and \( s - 1 \) (why?). Consider using a three dimensional array to represent your problem, where each cell of your array stores a boolean, and one dimension of your array runs from 0 to \( p \), where \( p = \sum_i |a_i| \).]

2. [30 marks]

(a) [5 marks] Run the breadth-first search on the undirected graph \( G_1 \), starting from vertex 1, and show its final output. When you have to choose which vertex to process next (and that choice is not otherwise specified by the BFS algorithm), use the one with the smallest label. Draw the tree edges with solid lines and the non-tree edges with dashed lines. Indicate the layers of the BFS tree.
(b) [5 marks] Is $G_1$ bipartite? Justify your answer with reference to the BFS tree.

(c) [5 marks] Repeat part (a) on the directed graph $G_2$.

(d) [5 marks] Run the depth-first search algorithm on the undirected graph $G_1$, starting from vertex 1, and show its final output. When you have to choose which vertex to process next (and that choice is not otherwise specified by the DFS algorithm), use the one with the smallest label.

(e) [5 marks] Repeat (d) on the directed graph $G_2$.

(f) [5 marks] Is $G_2$ strongly connected? Justify your answer with reference to a search tree (or trees).

3. [10 marks]
Given a directed graph $G = (V, E)$ with $m$ edges and $n$ vertices, we say an edge $e$ is circular if there exists a cycle that contains $e$. Give an $O(m + n)$-time algorithm to identify all circular edges in $G$.

4. [20 marks]
Assume that you need to divide $n$ people into two teams (say team 0 and team 1). The participants have requests of the form “I want to be in the same team as X” or “I do not want to be in the same team as Y”. We would like to divide the participants into teams while trying to satisfy their requests. Let the total number of requests be $k$. The resulting teams do not have to be equal size. In an extreme case we might put all the participants into the same team. Design an efficient algorithm to divide the participants into two teams while satisfying all the requests, assuming that is possible.

[Hint: convert the problem input into an appropriate graph, find the strongly components of this graph. Then construct an appropriate “meta-graph” consisting of the strongly connected components and solve the problem on this “meta-graph”.

5. [20 marks]
Suppose that you have found a collection of historical records indicating the relative order in which various people lived and died. Each record tells you one of the following:

- Person A and person B were alive at the same time.
- Person A died before person B was born.

Your task is to determine whether the historical records are consistent; that is, whether it was actually possible for all of these people to have lived and died in such a way that all of your records are accurate. Suppose that your records involve $n$ total people and you have $m$ records relating their lifespans. Design an $O(m + n)$-time algorithm that determines whether or not the records are consistent with one another. As usual, analyze the runtime of your algorithm and briefly argue why your algorithm is correct.

[Hint: Consider having two vertices for each person.]