TOBB ETÜ, BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİL 331/531 Design and Analysis of Algorithms

HOMEWORK 2 (100 Points)

Due Date: April 7, 2024

The specifics of implementation and submission will be announced in piazza soon.

1 [25 POINTS] MR. BIDIK'S NEVER-ENDING ADVENTURES

Your nephew Mr. Bıdık is interested in computer science. He stopped by the computer science department of a reputable private university, located at a central point of Ankara, to take part in a research project. The professors in the department and very kind people but at the same time they are demanding. In order to see the potential of Mr. Bıdık, professors decided to test his algorithmic skills. At that time, the department was establishing a department library and one needed to put the books on shelves. So, the professors decided to kill two birds with one stone. They gave the following task to Mr. Bıdık.

Consider placing n books on shelves in a library. The order of the books is fixed by the catalog numbers and cannot be changed. Each book b_i has a thickness t_i. The length of each bookshelf is L. The heights of the books can be ignored since any book fits on any shelf. Let k_j denote the total thickness of the books placed on shelf j, and assume a total of m shelves were used. We would like to minimize $\max_{1\leq j\leq m}\{L-k_j\}$ (the maximum unused length over m shelves). You can assume that there are enough shelves to place the books.

This problem was a piece of cake for Mr. Bıdık and he solved the problem quite efficiently. What about BİL 331 students? Is this an easy task for them? Let's see.
In this question, you are asked to write a Java method solve this problem. Your method should return the value of $\max_{1\leq j\leq m}\{L-k_j\}$. The signature of your method should be as follows:

public static double bookPlacement (double [] thickness, double L)

2 [25 POINTS] MOST PROFITABLE TRADES

You are looking at n consecutive days of a given stock, at some point in the past. The days are numbered $1,2,\ldots,n$. You are given $p[1..n]$ where $p[i]$ denotes the price per share for the stock on the i-th day. For a given integer k you want to know what is the maximum profit of a so-called *k-buy-sell strategy*. A k-buy-sell strategy is a collection of m pairs of days $(b_1,s_1),(b_2,s_2),\ldots,(b_m,s_m)$ for some $1 \le m \le k$ and $b_1 < s_1 < b_2 < s_2 < \ldots < b_m < s_m$. This can be viewed as a set of at most k non-overlapping intervals, during each of which you buy 1,000 shares of the stock (on day b_i) and then sell it (on day s_i). The profit of such a strategy is simply the profit of the m buy-sell transactions,

$$1000 \sum_{i=1}^{m} (p[s_i] - p[b_i]).$$

In this question, you are asked to write a Java method that takes p and k as input, and returns the maximum profit obtained by a k-buy-sell strategy throughout days 1 to n. The signature of your method should be as follows:
public static double letsGetRich (double [] p, int k)

3 [25 POINTS] NUMBER OF TIMES LIGHTS GET SWITCHED ON

There are n people entering and exiting a room. For each $i \in \{1,\ldots,n\}$, person i enters the room at time a_i and exits the room at time b_i (assume $b_i > a_i$ for all i), and all the a_i, b_i are distinct. At the beginning of the day, the lights in the room are switched off, and the first person who enters the room switches them on. In order to conserve electricity, when person i leaves the room at time b_i, if there is no one else present in the room at time b_i, then person i will switch the lights off. The next person to enter will then switch them on again. Given the values $(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)$, we want to find the number of times the lights get switched on. Design an $O(n \lg n)$-algorithm that computes the number of times the lights get switched on.
Hint: The nice part of designing $O(n \lg n)$-time algorithms is that you are allowed to sort the input.
Write a method with the following signature:
public int switchCount(int [ ] a, int [] b)

4 [25 POINTS] FINDING INDEPENDENT SET

In this question, you will be given an undirected acyclic and connected graph $G = (V,E)$ with nonnegative weights on the vertices. Your algorithm should return the total weight of the maximum weight independent set of G. Recall from the class that a subset of vertices S is called an independent set if no two vertices in S are the endpoints of the same edge.