German University in Cairo
Department of Computer Science
Dr. Nourhan Ehab

CSEN 703 Analysis and Design of Algorithms, Winter Term 2024
Assignment 1

Deadline: 19-11-2024

Exercise 1-1

The Carnival Ring Toss Challenge

Welcome to the bustling carnival! The air is filled with excitement, the sounds of laughter, and the delightful aroma of popcorn and cotton candy. As you wander through the maze of colorful tents, you spot the **Ring Toss Booth** — a classic carnival game, and the perfect opportunity to win a prize.

The game is simple: a row of shiny, colorful pegs stands before you, and your goal is to toss rings onto the pegs. But this is not just any ordinary ring toss game — there is a *twist*!

- **The Pegs**: In front of you is a row of pegs, the target heights of the pegs are predetermined and listed on a nearby scoreboard. The heights represent the number of rings that need to be on each peg.
- **The Rings**: You have an unlimited number of rings, but there is a catch: Your cousin Andy is with you and he is a master ring tosser!! He can toss multiple rings and hit multiple pegs at the same time, however, Andy's skill is limited a bit. Even though he can hit multiple pegs at the same time, they have to be a *contiguous* group of pegs. In other words, in **one** toss Andy can use any number of rings denoting the size of the contigious group of pegs that he has chosen, but only one ring will land on each peg. However, being a master ring tosser, it is guaranteed that all the pegs in the group will be hit.
- **The Goal** is to toss the rings so that the number of rings on each peg match exactly with the target heights on the scoreboard. But here's the challenge: You want to minimize the number of ring tosses to get everything aligned. The fewer the tosses Andy makes, the better the prize you get and you really want that Grogu figurine. You are the mind and Andy is the muscles. Plan Andy's tosses and trust he will hit whatever group of pegs you tell him to, so you need to figure out the minimum number of tosses Andy can make.

Walkthrough This section provides an example, where 5 pegs are used, to enhance your understanding of the problem only, but this does **not** mean that the number of pegs will be fixed. The heights of the pegs will be nonnegative integers and the height of one peg is unconstrained by the heights of the others. You will always begin with an array of pegs without rings: `[0,0,0,0,0]`.

Imagine the scoreboard shows the following target heights for the pegs.

`[2,3,4,2,1]`

The optimal solution for this example is to make **4 tosses** where Andy could have done the following.

1. Toss three rings on the first three pegs (index 0, 1 and 2) by adding 1 to each.
   After this toss, the array looks like this: `[1, 1, 1, 0, 0]`.
2. Toss three rings on the first three pegs again (index 0, 1, and 2) by adding 1 to each.
   After this toss, the array looks like this: `[2, 2, 2, 0, 0]`.

3. Toss four rings on the last four pegs (index 1, 2, 3, and 4) by adding 1 to each.
   After this toss, the array looks like this: `[2, 3, 3, 1, 1]`.
4. Toss two rings starting from the third peg (index 2 and 3) by adding 1 to each.
   After this toss, the array looks like this: `[2, 3, 4, 2, 1]`.

There *can* be several ways in which you reach the optimal solution but **This is a way ;)**

Requirements

- Your task is to solve the ring toss challenge using a greedy algorithm that runs in $O(n)$.

Deliverables

You are required to submit one **Java** file titled `RingToss` containing the following methods. The class name must be `RingToss` and the class must have a package name `csen703.main.assignment1`.

a) `public static int RingTossGreedy(int [] pegs)` that implements a greedy approach to finding the minimum number of tosses needed to match the scoreboard.

b) Notice that using static variables might mess up the test cases, therefore we advise against using static variables. There are other workarounds that you can do. **Failing to abide with class name, package name, or function signature will lead to an automatic zero.**

Regulations

1. A team can include up to 3 members.
2. You should write your solutions in Java.
3. You will submit your solutions through `https://forms.gle/kpqLZvmNqFmPZU7g6`.
4. The deadline for this assignment is on Tuesday, 19^{th} of November, 2024.

Plagiarism, including the use of GenAI tools such as GPT, Copilot, and similar assistants, is not tolerated. You are also advised against discussing elements of your solutions with other teams.