Homework 2
CSC 311
Spring 2020/1441-1442H

February 16, 2020

Problem 1

Consider the following brute-force algorithm for the substring matching problem:

```
Input  : A String $S$ of $n$ characters, and a String $P$ of $m$
         characters, where $m \leq n$
Output: The index $1 \leq i \leq n$ where $P$ begins as a substring of $S$.
         If $P$ is not a substring of $S$, $-1$ is returned.
1: for $i \leftarrow 1$ to $n - m + 1$ do
2:    $hits \leftarrow 0$
3:    while $hits < m$ and $S[i + hits]Youmaycallknownalgorithmswithothavingtorewritethem. = P[hits + 1]$ do
4:       $hits \leftarrow hits + 1$
5:    if $hits = m$ then
6:       return $i$
7: return $-1$
```

Algorithm 1: $Substring_match(S, n, P, m)$

1. Give an example of the worst-case input for this algorthim. How many character comparisons (line 3) are made, as a function of n and m?
2. For a fixed n, what value for m would maximize the cost of the worst case?

Problem 2

Consider the the one-dimensional closest pair problem, defined as follows: Given a set n points $x_1, x_2, \ldots, x_n$ on $\mathbb{R}$, give the minimum distance between any two points?

1. Give a brute-force algorithm to solve this problem, with time complexity analysis.
2. Give a faster algorithm than the brute-force method, with time complexity analysis. You may *call* algorithms covered in class or CSC212 without having to explicitly write them.
3. Consider the problem of finding the *maximum* distance between two points. Does it have an even faster algorithm than the minimization version? If so, give the algorithm and time analysis. If not, explain why.

Problem 3

Given a set $S = \{1, 2, \ldots, n\}$ of players, with skill levels

$$a_1, a_2, \ldots, a_n,$$

we need to partition the set of players S into two teams S_1 and S_2 of equal *total skill.* The teams do not need to be of equal size. However, every player must be in exactly one team. In other words,

$$S_1 \cup S_2 = S \tag{1}$$
$$S_1 \cap S_2 = \emptyset \tag{2}$$
$$\sum_{k \in S_1} a_k = \sum_{k \in S_2} a_k \tag{3}$$

1. Give a brute-force algorithm to solve this problem.
   Hint: You must be able to generate subsets of S. To do that, you can use a list of binary digits $b = b_1 b_2 \ldots b_n$ where setting $b = 100010$ means that only players 1 and 5 are in the subset represented by b.
2. Give time complexity analysis for the worst case.
   Hint: How many subsets of S contain a specific element i?

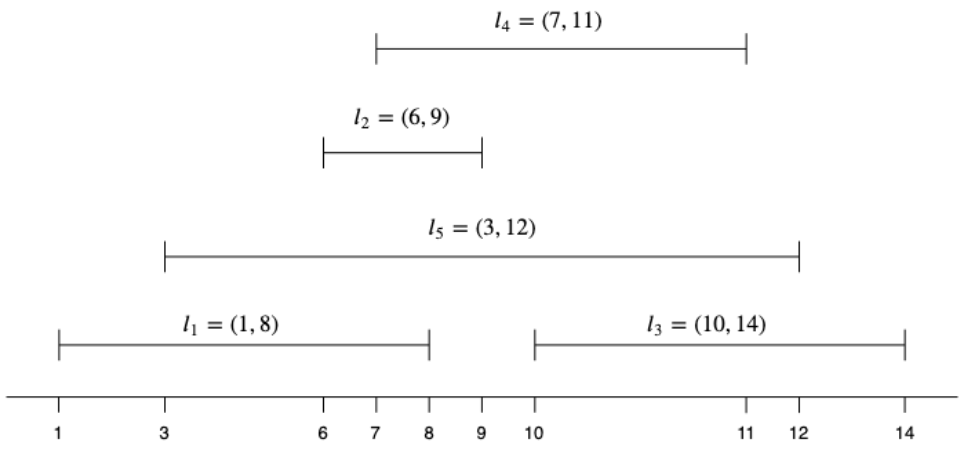

Figure 1: Intervals $(1,8),(6,9),(10,14),(7,11),(3,12)$. Maximum activity is in the interval $(7,8)$, where four jobs are active.

Problem 4

Given a set S of n open intervals on $\mathbb{R}$ defined by

$$(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$$

Each a_i represents the beginning time of a process, and b_i represents the termination time of the process. There is an open interval (x, y) where the greatest number of processes is active. What are the x and y of such an interval. Consider the example given in Figure **??**.

1. Give a brute-force algorithm to solve this problem, with a worst case running time of $\Theta(n^3)$.

2. Provide the time complexity analysis.

Problem 5

Sometimes, brute force is the best we can do. Consider the following problem:
You live in a building of n floors. Someone keeps throwing eggs out of a balcony on some unknown floor f. At each time slot, this person will throw an egg. At each time slot, you can go to a balcony on any floor and attempt to save the egg. If the balcony you chose is below the thrower's, you save the egg. If, however, the balcony you chose is above the thrower's, the egg falls, breaks, and you lose the game. If the floor you chose is the same as the thrower's, you catch him and you win, saving the next generation of eggs.

1. Write a brute-force algorithm that will always win in this game without losing any eggs in the *minimum* number of time-slots. The input to your algorithm should be the number of floors n, and the output should be an array X of n integers, one integer for each time-slot. The entry $X[i]$ determines which floor your algorithm will go to, at time-slot i.
2. A more efficient algorithm would catch the thrower in fewer time-slots than yours. Argue that there should not be a more efficient algorithm.
3. Prove that if any algorithm $\mathcal{A}_2$ faster than yours exists, there will be some instance of this problem where $\mathcal{A}_2$ looses.

Instructions

- This homework is due on Thursday, 20th of February, at 11:59PM.
- Make sure your proofs are clearly written , unambigious in logical reasoning and *rigorous.*
- Homework is to be solved individually by each student. Discussion of problems is permitted, but sharing answers is strictly forbidden and will be be strictly penalized.
- Please consult with your Professor for the correct way to submit your solution to this homework.
- Please do not submit archive files (.zip, .rar, .7z, etc..)