CSOR W4231.001
Feb 29, 2020
Eleni Drinea

Hackathon session (max score: 140)

Moderators: Amaury Sudrie, Jake Stamell, Kevin Wibisono, Ruizhe Li, Tian Wang, Yanda Chen, Yiyang Sun, Yonghe Zhao

Please turn off the Wi-Fi on your laptop at this point. You're encouraged to solve the problems on paper first. This will allow us to give you partial credit for partially correct algorithms and/or code.

If you solved a problem and your output matches the output for our test cases, you should ask a TA to review your solution and code so they can give you credit for it. If your output does not match our output for some or all of our test cases, and you still do not know why this is the case after thinking about it for a while, you should ask a TA to review your written solution or your code so they can assign you partial credit, if applicable.

At the end of the 3 hours, you should create a tgz file with your code called `hackathon-<myUNI>.tgz` *using* `tar -czvf`*. Then turn on the Wi-Fi on your laptop, and submit the tgz file on courseworks under assignment Hackathon.*

If you did not write code for some problems but came up with some solutions on paper, take pictures of your solutions (or scan them use CamScan), tar them together with any code you wrote and upload the tgz file on courseworks under assignment Hackathon.

1. (30 points) `Max sum of monotonically increasing subsequence`

You're given a sequence of n positive integers $\langle a_1, \ldots, a_n \rangle$. You want to compute the maximum sum of any monotonically increasing subsequence of integers.

A *subsequence* is any subset of the numbers in the original sequence taken in order, of the form $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_k} \rangle$, where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. A *monotonically increasing subsequence* is a subsequence in which the numbers are getting strictly larger.

Examples:

i. Input: $\langle 1, 101, 2, 3, 100, 4, 5 \rangle$
Output: 106 *(corresponding to the sum of the numbers in the subsequence* $\langle 1, 2, 3, 100 \rangle$*)*

ii. Input: $\langle 173, 48, 118, 193, 68, 196 \rangle$
Output: 562

iii. Input: $\langle 38, 141, 73, 138, 134, 80, 193 \rangle$
Output: 442

iv. Input: $\langle 169, 16 \rangle$
Output: 169

v. Input: $\langle 100, 190, 119, 145, 74 \rangle$
Output: 364

vi. Input: $\langle 176, 197, 26, 68, 152, 104, 93, 186, 143 \rangle$
Output: 432

vii. Input: $\langle 127, 93, 127, 183, 57, 151, 126, 66 \rangle$
Output: 403

viii. Input: $\langle 139, 107, 110, 80 \rangle$
Output: 217

ix. Input: $\langle 196, 85, 106, 134, 137 \rangle$
Output: 462

x. Input: $\langle 123, 175, 184, 198 \rangle$
Output: 680

Hint: If you find this problem hard, see next page for a somewhat easier version of the problem and its solution.

Warm-up exercise with solution—you do not need to solve it or code it up!

`Length of longest monotonically increasing subsequence`

You're given an array of n positive integers $a_1, \ldots, a_n$. You want to compute the length of a longest monotonically increasing subsequence of integers in the array.

In Example i. in the previous page, the output should be 5, corresponding to the subsequence $\langle 1, 2, 3, 4, 5 \rangle$.

Solution:

Let

$$OPT(i) = \text{max length of monotonically increasing subsequence } \textbf{ending at } a_i.$$

We want

$$\max_{1 \leq i \leq n} OPT(i)$$

Recurrence:

$$OPT(i) = 1 + \max_{\substack{1 \leq j < i \\ a_j < a_i}} OPT(j)$$

Boundary conditions: $OPT(0) = 0$, $OPT(1) = a_1$.

Running time: $O(n^2)$.

2. (30 points) `Longest palindromic substring`

Given a nonempty string s, give a Dynamic Programming algorithm to find the length of a longest **palindromic substring** in s. You may assume that the maximum length of s is 1000.

A **substring** of a string is a contiguous sequence of characters from the string. For example, on input $s = abbbaa$, abb and bba are substrings of s but aba is not (although it is a subsequence of s).

A **palindrome** is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are `01210`, `racecar` and $\nu\iota\psi o\nu\alpha\nu o\mu\eta\mu\alpha\tau\alpha\mu\eta\mu o\nu\alpha\nu o\psi\iota\nu$, a famous palindromic phrase inscribed on a fountain in the city of Constantinople in the time of the Byzantine Empire.

Examples:

i. Input: babad
   Output: 3 (corresponding to bab or aba)
ii. Input: fox
   Output: 1 (corresponding to f, o or x)
iii. Input: amaury
   Output: 3 (corresponding to ama)
iv. Input: hatee
   Output: 2 (corresponding to ee)
v. Input: wibisono
   Output: 3 (corresponding to ibi or ono)
vi. Input: neveroddoreven
   Output: 14 (corresponding to neveroddoreven)
vii. Input: derekchen
   Output: 3 (corresponding to ere)
viii. Input: programming
   Output: 2 (corresponding to mm)
ix. Input: akusukarajawalibilawajarakusuka
   Output: 31 (corresponding to akusukarajawalibilawajarakusuka)
x. Input: wapapapapapapow
   Output: 11 (corresponding to papapapapap or apapapapapa)

3. (40 points) `Burst balloons`

You are given n balloons, indexed from 0 to $n-1$. Balloon i is labelled with an integer number $nums[i]$.

You are asked to burst all the balloons. If you burst balloon i you will get $nums[left] \cdot nums[i] \cdot nums[right]$ coins, where $left = i-1$ and $right = i+1$. After the burst, the balloons at positions $left$ and $right$ become adjacent.

Give a dynamic programming algorithm to compute the maximum number of coins you can collect by bursting all the balloons. Assume $nums[-1] = nums[n] = 1$ (the balloons at these positions are not real therefore you can not burst them) and $0 \leq n \leq 500, 0 \leq nums[i] \leq 100$.

Examples:

i. Input: $\langle 3, 1, 5, 8 \rangle$
Output: 167
Explanation:
nums = [3,1,5,8], burst 1, get coins 3*1*5 = 15
nums = [3,5,8], burst 5, get coins 3*5*8 = 120
nums = [3,8], burst 3, get coins 1*3*8 = 24
nums = [8], burst 8, get coins 1*8*1 = 8
Total coins = 15 + 120 + 24 + 8 = 167

ii. Input: $\langle 1, 5 \rangle$
Output: 10

iii. Input: $\langle 46, 22, 60, 12, 27 \rangle$
Output: 155968

iv. Input: $\langle 2, 8, 2 \rangle$
Output: 40

v. Input: $\langle 1, 2, 3, 3 \rangle$
Output: 30

vi. Input: $\langle \rangle$ $(n = 0)$
Output: 0

vii. Input: $\langle 7, 2, 7, 4, 9, 6 \rangle$
Output: 1218

viii. Input: $\langle 7, 9, 8, 0, 7, 1, 3, 5, 5, 2 \rangle$
Output: 1582

ix. Input: $\langle 21, 12, 0, 47, 33, 49, 1, 14, 17, 16, 18, 37, 7, 1, 25, 46, 24, 9 \rangle$
Output: 479123

x. Input: $\langle 15, 40, 9, 0, 45, 20, 0, 67, 0, 0, 82, 0 \rangle$
Output: 521842

4. (40 points) `Army Arrangement`

You have an army of n_1 footmen and n_2 horsemen. In an *arrangement* of the soldiers, all $n_1 + n_2$ soldiers must be present. All footmen are considered indistinguishable among themselves. Similarly, all horsemen are considered indistinguishable among themselves.

An *arrangement* is considered *not beautiful* if somewhere in the line there are

- strictly more than k_1 footmen standing successively one after another; *or*
- strictly more than k_2 horsemen standing successively one after another.

Your task is to find the number of *beautiful* arrangements of the soldiers. (As usual, a solution that is more efficient in terms of time and/or space will get more points.)

`Input:` The only line contains four space-separated integers n_1, n_2, k_1, k_2, which represent how many footmen and horsemen there are, and the largest acceptable number of footmen and horsemen standing in succession, respectively. You may assume $1 \leq n_1, n_2 \leq 100$, and $1 \leq k_1, k_2 \leq 10$.

`Output:` Print the number of beautiful arrangements of the army modulo 100000000 (10^8).

Examples:

i. Input: 2 1 1 10 Output: 1
ii. Input: 2 4 1 1 Output: 0
iii. Input: 10 10 5 7 Output: 173349
iv. Input: 12 15 7 2 Output: 171106
v. Input: 20 8 4 8 Output: 162585
vi. Input: 15 8 2 6 Output: 156
vii. Input: 100 100 10 10 Output: 950492
viii. Input: 99 100 10 10 Output: 65210983
ix. Input: 1 3 10 10 Output: 4
x. Input: 2 2 10 10 Output: 6

5. *(Extra credit, if you still have time! Hard!)* `Box removal`

There are several boxes of different colors in a row. Each color is represented by a different positive integer.

Your goal is to remove all the boxes while maximizing your profit. The rules of the game are that (a) you may use several rounds to do so; and (b) in each round, you may choose any **contiguous boxes of the same color** to remove; if you remove, say, k boxes of the same color for some $k \geq 1$, you earn $k \cdot k$ points.

Find the maximum number of points you can get.

Examples:

i. Input: [1, 3, 2, 2, 2, 3, 4, 3, 1]   Output: 23
   Explanation: [1, 3, 2, 2, 2, 3, 4, 3, 1]
   $\longrightarrow$ [1, 3, 3, 4, 3, 1] (remove 2, $3 \times 3 = 9$ points)
   $\longrightarrow$ [1, 3, 3, 3, 1] (remove 4, $1 \times 1 = 1$ points)
   $\longrightarrow$ [1, 1] (remove 3, $3 \times 3 = 9$ points)
   $\longrightarrow$ [] (remove 1, $2 \times 2 = 4$ points)

ii. Input: [1, 1, 1]   Output: 9

iii. Input: [1, 1, 2]   Output: 5

iv. Input: [1, 2, 1, 2, 1, 2, 1]   Output: 19

v. Input: [1, 2, 3, 4, 5, 3, 2, 1]   Output: 14

vi. Input: [3, 5, 10, 7, 7, 4, 9, 10, 1, 9]   Output: 14

vii. Input:
   [3, 8, 8, 5, 5, 3, 9, 2, 4, 4, 6, 5, 8, 4, 8, 6, 9, 6, 2, 8, 6, 4, 1, 9, 5, 3, 10, 5, 3, 3, 9, 8, 8, 6, 5, 3, 7, 4, 9, 6, 3, 9, 4, 3, 5, 10, 7, 6, 10, 7]
   Output: 136

viii. Input:
   [86, 26, 80, 27, 1, 16, 78, 71, 36, 52, 65, 76, 58, 77, 45, 17, 100, 37, 37, 75, 49, 2, 37, 42, 19, 99, 14, 33, 34, 58, 4, 30, 100, 88, 74, 47, 80, 77, 85, 32, 80, 35, 80, 25, 60, 91, 99, 27, 47, 66, 13, 20, 15, 10, 26, 39, 60, 9, 63, 24, 66, 32, 29, 79, 67, 19, 88, 35, 44, 67, 22, 99, 27, 27, 40, 78, 2, 21, 40, 69, 88, 26, 57, 23, 15, 70, 1, 100, 37, 20, 26, 18, 27, 86, 88, 33, 28, 40, 92, 15]
   Output: 144

ix. Input:
   [12, 32, 47, 58, 5, 27, 13, 20, 49, 26, 36, 34, 17, 47, 32, 4, 1, 32, 51, 77, 58, 69, 87, 96, 53, 16, 66, 32, 41, 72, 78, 9, 52, 96, 51, 78, 69, 90, 2, 96, 38, 10, 63, 66, 21, 47, 75, 29, 52, 68, 30, 38, 94, 91, 89, 8, 36, 36, 33, 42, 32, 75, 23, 25, 64, 7, 80, 44, 15, 41, 46, 92, 24, 78, 91, 89, 28, 14, 44, 24, 31, 91, 81, 5, 50, 41, 41, 48, 61, 31, 39, 80, 41, 6, 86, 80, 73, 85, 58, 44]
   Output: 144

x. Input: []   Output: 0