**Tasks for Completion**

# Console Applications

## Task 01 – Max Counters (Counting Elements)

You are given N counters, initially set to 0, and you have two possible operations on them:

increase(X) − counter X is increased by 1,

max counter − all counters are set to the maximum value of any counter.

A non-empty array A of M integers is given. This array represents consecutive operations:

if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),

if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A such that:

A[0] = 3

A[1] = 4

A[2] = 4

A[3] = 6

A[4] = 1

A[5] = 4

A[6] = 4

the values of the counters after each consecutive operation will be:

(0, 0, 1, 0, 0)

(0, 0, 1, 1, 0)

(0, 0, 1, 2, 0)

(2, 2, 2, 2, 2)

(3, 2, 2, 2, 2)

(3, 2, 2, 3, 2)

(3, 2, 2, 4, 2)

The goal is to calculate the value of every counter after all operations.

Write a function:

class Solution { public int[] solution(int N, int[] A); }

that, given an integer N and a non-empty array A consisting of M integers, returns a sequence of integers representing the values of the counters.

Result array should be returned as an array of integers.

For example, given:

A[0] = 3

A[1] = 4

A[2] = 4

A[3] = 6

A[4] = 1

A[5] = 4

A[6] = 4

the function should return [3, 2, 2, 4, 2], as explained above.

Write an efficient algorithm for the following assumptions:

N and M are integers within the range [1..100,000];

each element of array A is an integer within the range [1..N + 1].

## Task 02 – Genomic Range Query (Prefix Sums)

A DNA sequence can be represented as a string consisting of the letters A, C, G and T, which correspond to the types of successive nucleotides in the sequence. Each nucleotide has an impact factor, which is an integer. Nucleotides of types A, C, G and T have impact factors of 1, 2, 3 and 4, respectively. You are going to answer several queries of the form: What is the minimal impact factor of nucleotides contained in a particular part of the given DNA sequence?

The DNA sequence is given as a non-empty string S = S[0]S[1]...S[N-1] consisting of N characters. There are M queries, which are given in non-empty arrays P and Q, each consisting of M integers. The K-th query (0 ≤ K < M) requires you to find the minimal impact factor of nucleotides contained in the DNA sequence between positions P[K] and Q[K] (inclusive).

For example, consider string S = CAGCCTA and arrays P, Q such that:

P[0] = 2 Q[0] = 4

P[1] = 5 Q[1] = 5

P[2] = 0 Q[2] = 6

The answers to these M = 3 queries are as follows:

The part of the DNA between positions 2 and 4 contains nucleotides G and C (twice), whose impact factors are 3 and 2 respectively, so the answer is 2.

The part between positions 5 and 5 contains a single nucleotide T, whose impact factor is 4, so the answer is 4.

The part between positions 0 and 6 (the whole string) contains all nucleotides, in particular nucleotide A whose impact factor is 1, so the answer is 1.

Write a function:

class Solution { public int[] solution(string S, int[] P, int[] Q); }

that, given a non-empty string S consisting of N characters and two non-empty arrays P and Q consisting of M integers, returns an array consisting of M integers specifying the consecutive answers to all queries.

Result array should be returned as an array of integers.

For example, given the string S = CAGCCTA and arrays P, Q such that:

P[0] = 2 Q[0] = 4

P[1] = 5 Q[1] = 5

P[2] = 0 Q[2] = 6

the function should return the values [2, 4, 1], as explained above.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [1..100,000];

M is an integer within the range [1..50,000];

each element of arrays P, Q is an integer within the range [0..N − 1];

P[K] ≤ Q[K], where 0 ≤ K < M;

string S consists only of upper-case English letters A, C, G, T.

## Task 03 – Triangle (Sorting)

An array A consisting of N integers is given. A triplet (P, Q, R) is triangular if 0 ≤ P < Q < R < N and:

A[P] + A[Q] > A[R],

A[Q] + A[R] > A[P],

A[R] + A[P] > A[Q].

For example, consider array A such that:

A[0] = 10 A[1] = 2 A[2] = 5

A[3] = 1 A[4] = 8 A[5] = 20

Triplet (0, 2, 4) is triangular.

Write a function:

class Solution { public int solution(int[] A); }

that, given an array A consisting of N integers, returns 1 if there exists a triangular triplet for this array and returns 0 otherwise.

For example, given array A such that:

A[0] = 10 A[1] = 2 A[2] = 5

A[3] = 1 A[4] = 8 A[5] = 20

the function should return 1, as explained above. Given array A such that:

A[0] = 10 A[1] = 50 A[2] = 5

A[3] = 1

the function should return 0.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [0..100,000];

each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

## Task 04 – Ladder (Fibonacci numbers)

You have to climb up a ladder. The ladder has exactly N rungs, numbered from 1 to N. With each step, you can ascend by one or two rungs. More precisely:

with your first step you can stand on rung 1 or 2,

if you are on rung K, you can move to rungs K + 1 or K + 2,

finally you have to stand on rung N.

Your task is to count the number of different ways of climbing to the top of the ladder.

For example, given N = 4, you have five different ways of climbing, ascending by:

1, 1, 1 and 1 rung,

1, 1 and 2 rungs,

1, 2 and 1 rung,

2, 1 and 1 rungs, and

2 and 2 rungs.

Given N = 5, you have eight different ways of climbing, ascending by:

1, 1, 1, 1 and 1 rung,

1, 1, 1 and 2 rungs,

1, 1, 2 and 1 rung,

1, 2, 1 and 1 rung,

1, 2 and 2 rungs,

2, 1, 1 and 1 rungs,

2, 1 and 2 rungs, and

2, 2 and 1 rung.

The number of different ways can be very large, so it is sufficient to return the result modulo 2P, for a given integer P.

Write a function:

class Solution { public int[] solution(int[] A, int[] B); }

that, given two non-empty arrays A and B of L integers, returns an array consisting of L integers specifying the consecutive answers; position I should contain the number of different ways of climbing the ladder with A[I] rungs modulo 2B[I].

For example, given L = 5 and:

A[0] = 4 B[0] = 3

A[1] = 4 B[1] = 2

A[2] = 5 B[2] = 4

A[3] = 5 B[3] = 3

A[4] = 1 B[4] = 1

the function should return the sequence [5, 1, 8, 0, 1], as explained above.

Write an efficient algorithm for the following assumptions:

L is an integer within the range [1..50,000];

each element of array A is an integer within the range [1..L];

each element of array B is an integer within the range [1..30].

## Task 05 – Min Abs Sum (Dynamic programming)

For a given array A of N integers and a sequence S of N integers from the set {−1, 1}, we define val(A, S) as follows:

val(A, S) = |sum{ A[i]\*S[i] for i = 0..N−1 }|

(Assume that the sum of zero elements equals zero.)

For a given array A, we are looking for such a sequence S that minimizes val(A,S).

Write a function:

class Solution { public int solution(int[] A); }

that, given an array A of N integers, computes the minimum value of val(A,S) from all possible values of val(A,S) for all possible sequences S of N integers from the set {−1, 1}.

For example, given array:

A[0] = 1

A[1] = 5

A[2] = 2

A[3] = -2

your function should return 0, since for S = [−1, 1, −1, 1], val(A, S) = 0, which is the minimum possible value.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [0..20,000];

each element of array A is an integer within the range [−100..100].

## Task 06 – Min Max Division (Binary search algorithm)

You are given integers K, M and a non-empty array A consisting of N integers. Every element of the array is not greater than M.

You should divide this array into K blocks of consecutive elements. The size of the block is any integer between 0 and N. Every element of the array should belong to some block.

The sum of the block from X to Y equals A[X] + A[X + 1] + ... + A[Y]. The sum of empty block equals 0.

The large sum is the maximal sum of any block.

For example, you are given integers K = 3, M = 5 and array A such that:

A[0] = 2

A[1] = 1

A[2] = 5

A[3] = 1

A[4] = 2

A[5] = 2

A[6] = 2

The array can be divided, for example, into the following blocks:

[2, 1, 5, 1, 2, 2, 2], [], [] with a large sum of 15;

[2], [1, 5, 1, 2], [2, 2] with a large sum of 9;

[2, 1, 5], [], [1, 2, 2, 2] with a large sum of 8;

[2, 1], [5, 1], [2, 2, 2] with a large sum of 6.

The goal is to minimize the large sum. In the above example, 6 is the minimal large sum.

Write a function:

class Solution { public int solution(int K, int M, int[] A); }

that, given integers K, M and a non-empty array A consisting of N integers, returns the minimal large sum.

For example, given K = 3, M = 5 and array A such that:

A[0] = 2

A[1] = 1

A[2] = 5

A[3] = 1

A[4] = 2

A[5] = 2

A[6] = 2

the function should return 6, as explained above.

Write an efficient algorithm for the following assumptions:

N and K are integers within the range [1..100,000];

M is an integer within the range [0..10,000];

each element of array A is an integer within the range [0..M].

## Task 07 – Count Non Divisible (Sieve of Eratosthenes)

You are given an array A consisting of N integers.

For each number A[i] such that 0 ≤ i < N, we want to count the number of elements of the array that are not the divisors of A[i]. We say that these elements are non-divisors.

For example, consider integer N = 5 and array A such that:

A[0] = 3

A[1] = 1

A[2] = 2

A[3] = 3

A[4] = 6

For the following elements:

A[0] = 3, the non-divisors are: 2, 6,

A[1] = 1, the non-divisors are: 3, 2, 3, 6,

A[2] = 2, the non-divisors are: 3, 3, 6,

A[3] = 3, the non-divisors are: 2, 6,

A[4] = 6, there aren't any non-divisors.

Write a function:

class Solution { public int[] solution(int[] A); }

that, given an array A consisting of N integers, returns a sequence of integers representing the amount of non-divisors.

Result array should be returned as an array of integers.

For example, given:

A[0] = 3

A[1] = 1

A[2] = 2

A[3] = 3

A[4] = 6

the function should return [2, 4, 3, 2, 0], as explained above.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [1..50,000];

each element of array A is an integer within the range [1..2 \* N].

# Web Projects

## Web Trading App

Create a web trading project using ASP.NET Core.

* Database – depends on your choice (MS SQL, **Kinvey**, Oracle, MySQL,…)
  + Use Entity Framework to connect to the database and extract the requested information from the client.
* Items in the trading platform
  + Title
  + Category
  + Start price
  + Min bet price (shows error message if user selects lower price than the **current price + min bet price**)
  + Start date (The date of item creation)
  + End date (The date when bidding ends)
  + Specifics
  + Information
  + Item pictures
  + Option to bid or buy
    - If item is set to **bid** we need to wait to the end date to select a winner
    - If item is set to **buy** the first who bids is the winner
* User roles
  + Administrator
    - Ban Users
    - Make/Unmake Administrators
    - Set/Unset **Item makers** from the **Item requestors** list
    - Administrator is not able to change his role
    - Can add to favorites items
    - Can add comment on items in the trading platform
    - Update their settings
      * **Can not change username**
      * Change password
      * Update information
      * Update avatar
  + User Item Maker
    - Can view the dashboard page – Web page which contains information about:
      * the total count of all items in the trading platform
      * the sum of the money income from the successful items
      * graphics of the money income about every day in the last 30 days
    - Can view the items in the trading platform
    - Can add to favorites items
    - Can add comment on items in the trading platform
    - Create new items in the trading platform
    - Restart unsuccessful items (items that have reached the end date and no one bid on them)
    - Delete items from the trading platform
    - Update their settings
      * **Can not change username**
      * Change password
      * Update information
      * Update avatar
  + User Item Requestor
    - Can view the items in the trading platform
    - Can add to favorites items
    - Can bid on the trading items
    - Can add comment on items in the trading platform
    - Update their settings
      * **Can not change username**
      * Change password
      * Update information
      * Update avatar
  + Not logged
    - Can view the items in the trading platform
* Front-end
  + Bootstrap 4
  + Native JavaScript (**jQuery not preferable**)

## Web Blog

Create web blog in ASP.NET Core

* Database – depends on your choice (MS SQL, Oracle, MySQL)
  + The project logic must be in db stored procedures or db functions
  + Use the server to call the different stored procedures or functions used in the project
* User roles
  + Logged User
    - Can view all blog items (ordered by date)
    - Create new blog item
    - Comment on other blog items
    - Can like blog items
    - Update their settings
      * **Can not change username**
      * Change password
      * Update information
  + Not logged
    - Can view all blog items (ordered by date)
* Front-end
  + Semantic-UI
  + jQuery