Freshmen Programming Contest 2021

Solutions presentation

May 9, 2021

A: Alleys Construction

Problem Author: Cristian - Alexandru Botocan

Problem:

- Given a number *n* for each query, you have to compute the number of possible ways in which alleys can be built for *n* houses

Statistics: 10 submissions, 0 accepted, 6 unknown

A: Alleys Construction

Problem Author: Cristian - Alexandru Botocan

Problem:

- Given a number n for each query, you have to compute the number of possible ways in which alleys can be built for n houses

Solution:

- The first observation is that for each number n you have to calculate the $C_{n/2}$ (Catalan number of (n/2))
    - The formula for Catalan number of n is $C(n) = \frac{1}{n+1} \cdot \binom{2n}{n} = \frac{1}{n+1} \cdot \frac{2n!}{n!(2n-n)!}$
    - Since all the n numbers will be even, we will not have any issues to compute $n/2$
    - Thus, we have just to compute the expresion: $C(n/2) = \frac{1}{(n/2)+1} \cdot \binom{n}{n/2}$
- The second observation is that we can precompute all the factorials until 313109.

A: Alleys Construction

Problem Author: Cristian - Alexandru Botocan

Problem:

- Given a number n for each query, you have to compute the number of possible ways in which alleys can be built for n houses

Solution:

- We have to compute the following expression and the result should be modulo 313109. How we will compute the Catalan number then?
  - For computing the binomial coefficient, we will use the Lucas' theorem:
    - $\binom{m}{n} = \prod_{i=0}^{k} \binom{m_i}{n_i}$, where:
      $m = m_k p^k + m_{k-1} p^{k-1} + \cdots + m_1 p + m_0$ and
      $n = n_k p^k + n_{k-1} p^{k-1} + \cdots + n_1 p + n_0$
      are the base p expansions of m and n respectively.
    - This uses the convention that $\binom{m}{n} = 0$ if $m < n$.

A: Alleys Construction

Problem Author: Cristian - Alexandru Botocan

Problem:

- Given a number n for each query, you have to compute the number of possible ways in which alleys can be built for n houses

Solution:

- For computing the denominators of an expression, we will use the modular inverse, which will have the time complexity $\mathcal{O}(\log p)$, where p represents the prime number 313109
- Thus, for each query we will have the maximum time complexity $\mathcal{O}(\log_p n)$
- Overall, the entire program will have the maximum time complexity $\mathcal{O}(p + q \cdot \log_p n)$, where p represents the prime number 313109

A: Alleys Construction

Problem Author: Cristian - Alexandru Botocan

Problem:

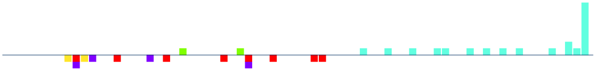
- Given a number *n* for each query, you have to compute the number of possible ways in which alleys can be built for *n* houses

Pitfalls:

- Forgetting to use the modular inverse for computing the value of the denominators.
- Forgetting to use the Lucas' theorem for computing the binomial coefficient.

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



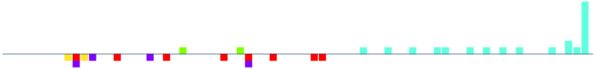
Problem:

- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Statistics: 37 submissions, 2 accepted, 21 unknown

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

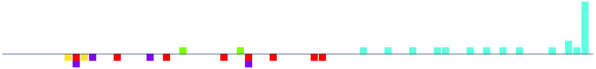
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

First idea

- Pick two indices
- Loop over it and find smallest price and count number of days included
- Multiply smallest price with the found width (in days)
- This is too slow! :(
- Around $(25 \cdot 10^8)$ operations

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

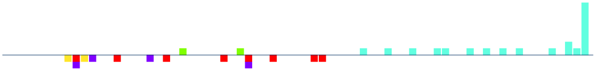
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Stack (recursion or iterative)

- Keep a stack of possible ”barriers”
    - if current price is bigger than top of the stack, then top is barrier
    - if not, then pop, because what’s on top will never be a barrier again
    - pop until find smaller price (possible barrier)
    - careful: in stack, keep indices not prices, otherwise impossible to calculate the width of the bubble

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

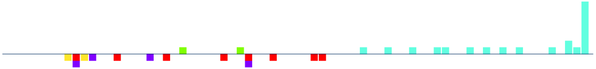
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Let's take an example (left)

- sequence : 3 9 5 7
- stack : empty

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

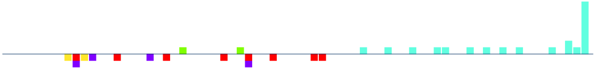
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Let's take an example (left)

- sequence : 3 9 5 7
- stack : 3

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

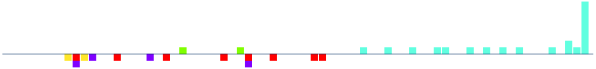
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Let’s take an example (left)

- sequence : 3 9 5 7
- stack : 3 9

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

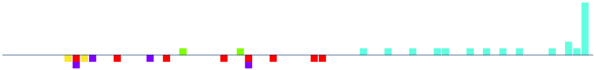
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Let’s take an example (left)

- sequence : 3 9 5 7
- stack : 3 5

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

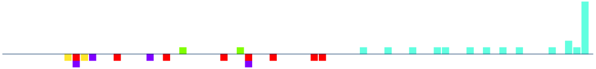
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Let's take an example (left)

- sequence : 3 9 5 7
- stack : 3 5 7

B: Bitcoin Bubble

Problem Author: Dragos Vecerdea



Problem:

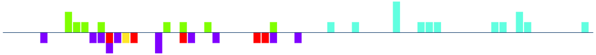
- Given a sequence of number, for every position, compute how many consecutive numbers in a row are smaller than the number on the selected position

Pitfalls:

- Only one price information
- Integers are not enough (use long long)

C: Coatis and Owls

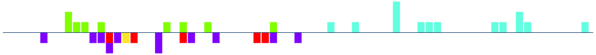
Problem Author: Maarten Sijm



- Problem: calculate the winner of a battlefield with squads of pikemen.

C: Coatis and Owls

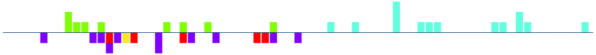
Problem Author: Maarten Sijm



- Problem: calculate the winner of a battlefield with squads of pikemen.
- Solution: simulate the game in $\mathcal{O}(n)$ time.
  - In other words: do not remove elements from the list in $\mathcal{O}(n)$ time!

C: Coatis and Owls

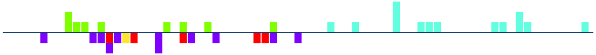
Problem Author: Maarten Sijm



- Problem: calculate the winner of a battlefield with squads of pikemen.
- Solution: simulate the game in $\mathcal{O}(n)$ time.
  - In other words: do not remove elements from the list in $\mathcal{O}(n)$ time!
- Pitfalls:
  - Using `float` instead of `double` for division/ceiling
  - Java: `Scanner` is too slow

C: Coatis and Owls

Problem Author: Maarten Sijm



- Problem: calculate the winner of a battlefield with squads of pikemen.
- Solution: simulate the game in $\mathcal{O}(n)$ time.
  - In other words: do not remove elements from the list in $\mathcal{O}(n)$ time!
- Pitfalls:
  - Using `float` instead of `double` for division/ceiling
  - Java: `Scanner` is too slow

Statistics: 40 submissions, 9 accepted, 14 unknown

D: Distribution Center

Problem Author: Alin Dondera

Problem:

- Find all squares in the grid from which it is impossible to move a crate to any destination.

Statistics: 13 submissions, 2 accepted, 9 unknown

D: Distribution Center

Problem Author: Alin Dondera

Problem:

- Find all squares in the grid from which it is impossible to move a crate to any destination.

Solution: do a modified BFS from the destinations

- Add all destinations to the queue and mark all other squares as dead squares
- Everytime we pop a position from the queue:
  - If already visited, we skip it
  - Else we add neighbouring *non-dead* squares in the queue
- A neighbouring square is *non-dead* if a crate can be pushed from that square to the current square
- To check that a crate can be pushed from a square in one of the four directions, we check that the square in the opposite direction is empty
- Lastly, all squares but the visited ones will be *dead*

D: Distribution Center

Problem Author: Alin Dondera

Problem:

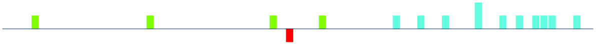
- Find all squares in the grid from which it is impossible to move a crate to any destination.

Pitfalls:

- Starting a BFS from each destination/square takes too much time
- Stack overflows

E: Efficient Grading

Problem Author: Alin Dondera



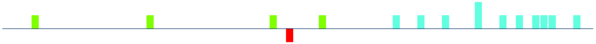
Problem:

- Given a number of exams, find the minimum amount of time needed to grade them. Also give the minimum amount of TAs needed for this time to be achieved.

Statistics: 16 submissions, 4 accepted, 11 unknown

E: Efficient Grading

Problem Author: Alin Dondera



Problem:

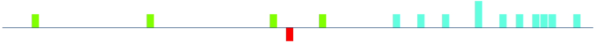
- Given a number of exams, find the minimum amount of time needed to grade them. Also give the minimum amount of TAs needed for this time to be achieved.

Solution: Calculate the time needed to grade all exams, assuming that at the end there will be exacly k TAs. Do this for all $1 \leq k \leq n$ and select the best result.

- The main observations here is that the best strategy for training k TAs is a greedy one. If we want to train a TA, it's best to do it as early as possible
- For the first part of the grading session we will train k TAs
- For the second part we will grade the exams

E: Efficient Grading

Problem Author: Alin Dondera



Problem:

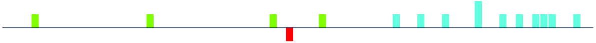
- Given a number of exams, find the minimum amount of time needed to grade them. Also give the minimum amount of TAs needed for this time to be achieved.

Solution:

- The amount of time-steps p needed to train k TAs is $\lceil \log_2(k) \rceil$
  - In the last time step we need to make sure that TAs who don't need to train, will instead grade exams
  - For example, when we go from 8 to 12 TAs, during this time-step, 4 will grade, while the other 4 will train
- Let q be the number of exams left to be graded after the training phase is done
- The total time needed to grade all exams will be $\left(p + \left\lceil \frac{q}{k} \right\rceil\right) \cdot t$

E: Efficient Grading

Problem Author: Alin Dondera



Problem:

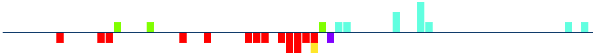
- Given a number of exams, find the minimum amount of time needed to grade them. Also give the minimum amount of TAs needed for this time to be achieved.

Pitfalls:

- Integer overflow
- Having TAs on ”idle” mode

F: Fraud Checking

Problem Author: Maarten Sijm



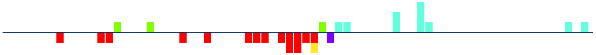
Problem:

- Test whether two pieces of code are *similar*, and if so, give the list of replacements.

Statistics: 30 submissions, 3 accepted, 10 unknown

F: Fraud Checking

Problem Author: Maarten Sijm



Problem:

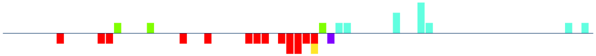
- Test whether two pieces of code are *similar*, and if so, give the list of replacements.

Solution:

- Split the lines of code into lists of words
  - If some lists have different lengths, exit
- Iterate over the words of both pieces of code
- Remember which word in code 1 maps to which word in code 2, and vice versa
  - If the same word later maps to something else, exit
- Print the sorted list of word replacements

F: Fraud Checking

Problem Author: Maarten Sijm



Problem:

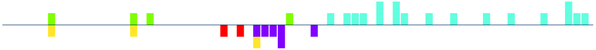
- Test whether two pieces of code are *similar*, and if so, give the list of replacements.

Pitfalls:

- Forgetting to sort
- Make sure that splitting a string on spaces results in empty words
- Forgetting to check whether two words map to the same word

G: Gardening

Problem Author: Dragos Vecerdea



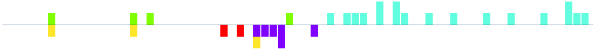
Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

Statistics: 33 submissions, 4 accepted, 18 unknown

G: Gardening

Problem Author: Dragos Vecerdea



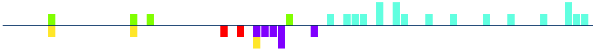
Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

Parsing

- Recursive function
  - keep a global index (current position)
  - read character
  - create node
  - move to next character
  - if character is '(', parse nodes until ')', otherwise return

G: Gardening



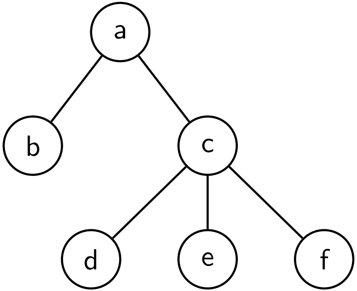
Problem Author: Dragos Vecerdea

Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

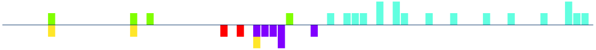
Removing leaves

- Key observation: post-order traversal is the order we are looking for
- Answer :



G: Gardening

Problem Author: Dragos Vecerdea

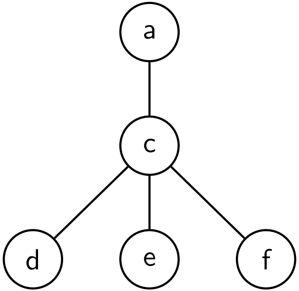


Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

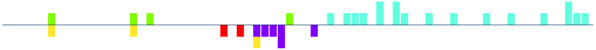
Removing leaves

- Key observation: post-order traversal is the order we are looking for
- Answer : b



G: Gardening

Problem Author: Dragos Vecerdea

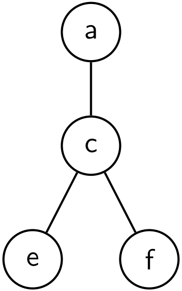


Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

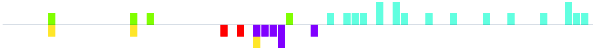
Removing leaves

- Key observation: post-order traversal is the order we are looking for
- Answer : b d



G: Gardening

Problem Author: Dragos Vecerdea

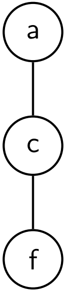


Problem:

- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

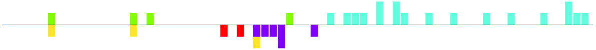
Removing leaves

- Key observation: post-order traversal is the order we are looking for
- Answer : b d e



G: Gardening

Problem Author: Dragos Vecerdea



Problem:

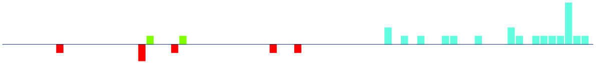
- Given a tree (encoded as string) parse it and remove leaves until tree is empty.

Pitfalls:

- Not considering only one node case eg. of tree: 'a'
- Slow parsing (operations with strings are not needed)

H: Heraldic Prediction

Problem Author: Angel Karchev



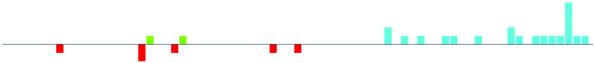
Problem:

- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

Statistics: 29 submissions, 2 accepted, 21 unknown

H: Heraldic Prediction

Problem Author: Angel Karchev



Problem:

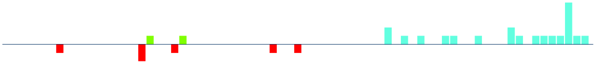
- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

Solution: Spotting the Pattern.

- The case where $p \neq 3$:
  - Every prime number besides 3 can be represented in the form $3k + 1$ or $3k + 2$.
  - $p^2 = (3k+1)^2 = 9k^2 + 6k + 1 = 3 \cdot (3k^2 + 2k) + 1$, or
    $p^2 = (3k+2)^2 = 9k^2 + 12k + 4 = 3 \cdot (3k^2 + 4k + 1) + 1$
  - To make $m + p^2$ divisible by 3, we need to pick an m such that $m \bmod 3 = 2$

H: Heraldic Prediction

Problem Author: Angel Karchev



Problem:

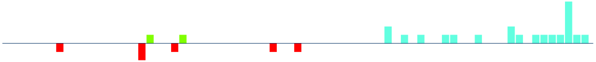
- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

Solution: Spotting the Pattern.

- The case where $p \neq 3$:
  - Every prime number besides 3 can be represented in the form $3k + 1$ or $3k + 2$.
  - $p^2 = (3k + 1)^2 = 9k^2 + 6k + 1 = 3 \cdot (3k^2 + 2k) + 1$, or
    $p^2 = (3k + 2)^2 = 9k^2 + 12k + 4 = 3 \cdot (3k^2 + 4k + 1) + 1$
  - To make $m + p^2$ divisible by 3, we need to pick an m such that $m \bmod 3 = 2$
- The case where $p = 3$:
  - $p^2 = 9$ and $9 \bmod 5 = 4$.
  - To make $m + p^2$ divisible by 5, we can pick an m such that $m \bmod 5 = 1$

H: Heraldic Prediction

Problem Author: Angel Karchev



Problem:

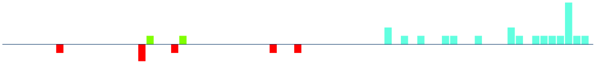
- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

Solution: Spotting the Pattern.

- The case where $p \neq 3$:
  - Every prime number besides 3 can be represented in the form $3k + 1$ or $3k + 2$.
  - $p^2 = (3k+1)^2 = 9k^2 + 6k + 1 = 3 \cdot (3k^2 + 2k) + 1$, or
    $p^2 = (3k+2)^2 = 9k^2 + 12k + 4 = 3 \cdot (3k^2 + 4k + 1) + 1$
  - To make $m + p^2$ divisible by 3, we need to pick an m such that $m \bmod 3 = 2$
- The case where $p = 3$:
  - $p^2 = 9$ and $9 \bmod 5 = 4$.
  - To make $m + p^2$ divisible by 5, we can pick an m such that $m \bmod 5 = 1$
- Remember, m has to be even, so $m \bmod 2 = 0$

H: Heraldic Prediction

Problem Author: Angel Karchev



Problem:

- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

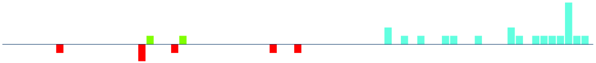
In conclusion, we can pick any number m where:

- $m \bmod 2 = 0$
- $m \bmod 3 = 2$
- $m \bmod 5 = 1$

So every number within the interval m, where $m = 26 + 30 \cdot l$, is a valid answer

H: Heraldic Prediction

Problem Author: Angel Karchev



Problem:

- Find an even number m within the given interval, for which $m + p^2$ is composite for every prime p.

Pitfalls:

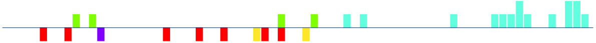
- Brute-forcing for a finite number of prime numbers *might* be possible within the time limit

  but those of you who tried, failed

- Making tests is hard, so a very well optimized/lucky solution could be accepted

I: Icarus' Rebirth

Problem Author: Cristian - Alexandru Botocan



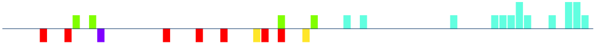
Problem:

- Given a string, we have to compute the minimum steps which we have to do get to the last character of the word if we start from the first character of the word.

Statistics: 29 submissions, 4 accepted, 15 unknown

I: Icarus' Rebirth

Problem Author: Cristian - Alexandru Botocan



Problem:

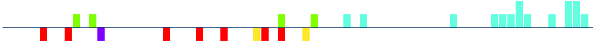
- Given a string, we have to compute the minimum steps which we have to do get to the last character of the word if we start from the first character of the word.

Solution:

- We can model this problem as a bidirectional graph traverse problem
  - We encode the characters of the string as nodes
  - For each character, we will have an edge with the left and right character
  - Moreover, for each character, we will have an edge with the first left character which is the same as the actual character
  - We will apply the same idea for the first right character which is the same as the actual character
- After constructing the graph, the result is the distance from the first character of the word to the last one, which can be computed by using BFS.
- Overall, the entire program will have the maximum time complexity $\mathcal{O}(n+n) = \mathcal{O}(n)$, where n represents the numbers of the letters in the given string

I: Icarus' Rebirth

Problem Author: Cristian - Alexandru Botocan



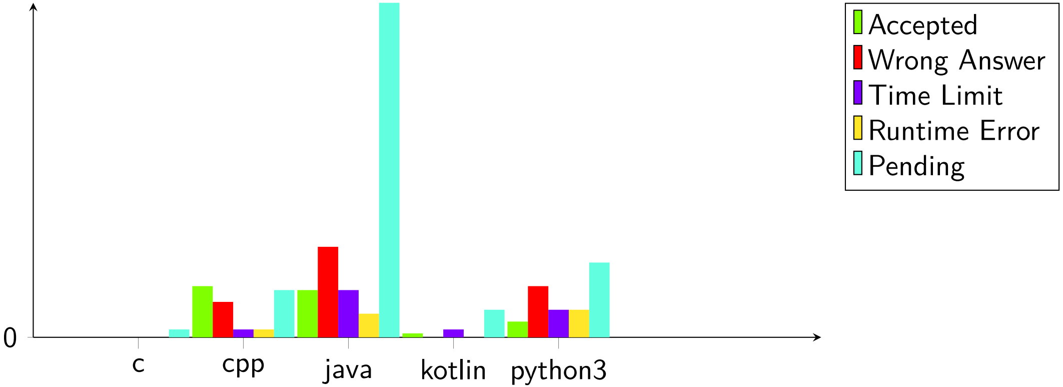
Problem:

- Given a string, we have to compute the minimum steps which we have to do get to the last character of the word if we start from the first character of the word.

Pitfalls:

- Applying DFS instead of BFS, if you are using a graph approach.

Language stats



Other stats

- 323 commits
- 219 secret testcases
- 44 accepted jury solutions, 21 WA and 8 TLE
- The minimum number of lines the jury needed to solve all problems is

$$23 + 11 + 17 + 14 + 10 + 18 + 10 + 1 + 21 = 125$$

(average: 13.9 lines per problem)

Thanks to:

The Proofreaders

- Arnoud van der Leer
- Davina van Meer
- Joey Haas
- Tim Huisman

The Jury

- Alin Dondera
- Angel Karchev
- Cristian - Alexandru Botocan
- Dragos Vecerdea
- Maarten Sijm