Assignment 3 (2023)

be21b004@smail.iitm.ac.in Switch account

Instructions

Please note that resubmission of the form and the code files is not allowed.

Write Python code for the following questions using only basic Python data structures, loops, and conditional statements.

Allowed : list, set, tuple, dictionary, variables (int, float), control statements (if-else, switch-case), loops (for, while)

For Assignment 3, you are allowed to use built-in sort function for sorting. You cannot use any inbuilt functions relating to binary search.

IMPORTANT: **Use the same function name and parameters given in each question**

All files that DISREGARD the naming convention will NOT be corrected and 0 marks will be awarded for the respective questions.

Please avoid any unnecessary print and input statements while submitting the code.

Write explanations as **comments** in the python file. **Code files without comments will receive a penalty.**

Q1. Magic door

Statement:
In a unique world, there exists a mystical staircase with ***n*** steps, each imbued with its own magical properties. Inhabitants of this world can ascend these stairs by taking one, two, or three steps at a time. Roger White, a magician finds a door at the end of the staircase to another magical world whose password is found by count of the number of ways to reach the door through the staircase.

Function Definition:
def numways(n: int) -> int:
The function takes input n and returns the number of unique ways we can reach the end of the staircase.
Expected time complexity: O(n)

Examples:
numways(2) = 2
numways(3) = 4

Add file

Q2. Forming groups

Statement:
There are ***n*** students in a class. The goal is to form a study group with the maximum number of students possible. However, to ensure that the study group is productive, the group should be balanced. In this context, balanced means that the difference in skill levels between any pair of students in the group **should not exceed 5**.

Given the skill levels of ith student as ***a_i***, the task is to determine the **maximum** number of students you can include in a balanced study group while adhering to the skill difference constraint.

Function definition:
def maxstud(n: int, a: list) -> int:

Output should return one integer — the maximum possible number of students you can include in a balanced study group.

Expected time complexity: O(nlogn)

Examples:

maxstud(5, [1,2,3,4,6]) = 5

maxstud(4, [5,1,8,7]) = 3

Add file

Q3. Array Product

Statement:
You are given a list of ***n*** numbers denoted as ***a_i***.
An operation on the list is defined as follows: you can **choose one element and remove or add one to the value of the element**.
The goal is to make the **product of the entire array equal to one** using **minimum number of operations**.

Function definition :
def minop(n: int, a: list) -> int:
Given n and array a, output the minimum number of operations required to attain the goal.
Expected Time Complexity: O(n)

Examples:
minop(3, [1,2,1]) = 1

minop(5, [1,2,3,4,6]) = 11

minop(4, [-5,-1,8,7]) = 17

Hint: Make sure to cover corner cases like having 0 in the array.

Add file

Q4. InfixPostfix

Statement:
Write a Python code consisting of two functions "InfixToPostfix" and "EvaluatePostfix". "InfixToPostfix" should convert a given string representation of an infix expression to postfix (as a list of elements). "EvaluatePostfix" should compute and return the numerical value of a postfix expression. Please return the required output in both functions. Assume that the only possible operators are '+, -, /, *, ^' (^ is the exponent operator). NOTE: For simplicity, the numeric entities in the infix expression are positive integers and not negative or floating point numbers.
Refer: https://eecs.wsu.edu/~nroy/courses/cpts122/labs/Infix2Postfix.pdf
Function defintions:
def InfixToPostfix(infix: str) -> list:
InfixToPostfix takes the string and outputs the postfix (as a list of elements)

def EvaluatePostfix(postfix: list) -> int:
EvaluatePostfix takes the postfix as a list and returns the numerical value of the expression
The efficiency of the algorithm is not checked for this problem
Examples: see image

```
infix = "3^4/(5*6)+10"
postfix = InfixToPostfix(infix)
print("Postfix expression: ", postfix)
result = EvaluatePostfix(postfix)
print("Value of expression: ",result)
```

```
Postfix expression:  [3, 4, '^', 5, 6, '*', '/', 10, '+']
Value of expression:  12.7
```

Add file

Q5. Stacks & Queues

Statement:
In a school cafeteria, there are circular sandwiches (represented by 0) and square sandwiches (represented by 1) available during lunchtime. Students stand in a queue, each having a preference for either circular or square sandwiches. The number of sandwiches in the cafeteria is equal to the number of students. These sandwiches are stacked up, with the top sandwich being the first to be picked.

At each step, the following happens:

- If the student at the front of the queue prefers the type of sandwich on the top of the stack, they will take it and leave the queue.
- If the student does not prefer the sandwich on the top of the stack, they will leave it and return to the end of the queue.

This process continues until no student in the queue wants to take the top sandwich, which means they are unable to eat it. You are given two arrays: "students," which represents the preferences of the students (0 for circular, 1 for square), and "sandwiches," which represents the types of sandwiches in the stack (0 for circular, 1 for square).
Your task is to determine the number of students who are unable to eat based on these preferences and sandwich availability.

Function definition:
def numstud(students: list, sandwiches: list) -> int:
The function numstud takes two lists students and sandwiches and returns the number of students who are unable to eat.
The efficiency of the algorithm is not checked for this problem

Examples: see image

Example 1:

```
Input: students = [1,1,0,0], sandwiches = [0,1,0,1]
Output: 0
Explanation:
- Front student leaves the top sandwich and returns to the end of the line making students =
[1,0,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students =
[0,0,1,1].
- Front student takes the top sandwich and leaves the line making students = [0,1,1] and sandwiches =
[1,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [1,1,0].
- Front student takes the top sandwich and leaves the line making students = [1,0] and sandwiches =
[0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,1].
- Front student takes the top sandwich and leaves the line making students = [1] and sandwiches = [1].
- Front student takes the top sandwich and leaves the line making students = [] and sandwiches = [].
Hence all students are able to eat.
```

Example 2:

```
Input: students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]
Output: 3
```

Add file

Q6:Agent Vega

Statement:

In a futuristic world, Agent Vega is on a secret mission inside a high-tech facility. She finds herself in a perplexing situation within a secure chamber. The chamber is designed as an N x N grid, with certain cells designated as clear pathways (represented by '.') and others fortified with laser barriers (represented by '#'). She is initially positioned in the south of the chamber and her mission requires her to gain visual access to a critical console located on the eastern side of the chamber.

Agent Vega has an advanced tool at her disposal: a 45-degree reflective prism. She can strategically position this prism on any clear pathway cell within the grid to observe the information on the eastern console through reflection. Exiting on any row toward the east will give access to the console.

However, here's the catch: if a laser barrier obstructs the path of the prism's reflection, it will block her view entirely. Agent Vega needs to determine the number of available clear pathway cells where she can place the prism to successfully gain visual access to the console on the eastern side of the chamber.

Your mission is to assist Agent Vega in calculating the count of such viable prism placement locations.

Function Definition:
def agent_vega(chamber: List[List[int]]) -> int The function should not print anything. It should return an integer which corresponds to the count of possible prism locations.
Expected Time complexity = O(N^2)

Examples:
agent_vega([['#', '.', '.'], ['#','.', '.'], ['#','.', '.']) = 6
agent_vega([['#', '.', '#'], ['#', '.', '#'], ['#', '.', '#']) = 0

Add file

Back Submit Page 2 of 2 Clear form