Assignment

Instructions

1. Write your functions according to the signature provided below in a python(.py) file and submit them to the autograder for evaluation.
2. The autograder has a limited number of attempts. Hence, test it thoroughly before submitting it for evaluation.
3. Save your python file as text(.txt) file and submit it to the canvas.
4. If the assignment has theoretical questions, upload those answers as a pdf file on the canvas.
5. Submission on the canvas is mandatory and the canvas submitted text file should be the same as that of the final submission of the autograder. If not, marks will be deducted.
6. No submission on canvas or no submission on autograder then marks will be deducted.
7. Access the auto grader at https://c200.luddy.indiana.edu.

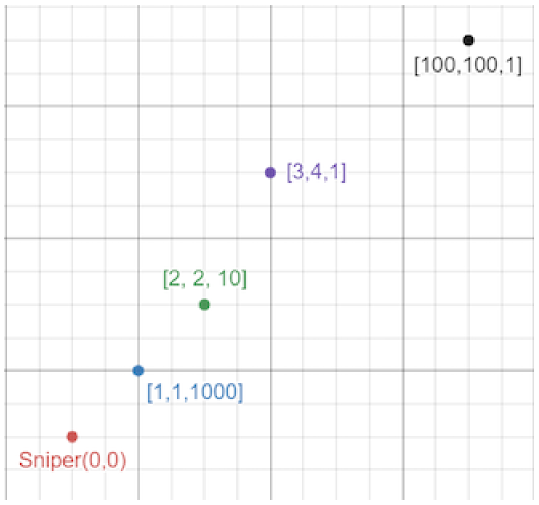
Question 1

Private Ryan, a highly skilled sniper in the American army, finds himself encircled by enemy forces. Despite being outnumbered and surrounded, Ryan's expertise in concealment enables him to remain undetected on a small hill. He faces the daunting task of eliminating the adversaries strategically, ensuring his survival amidst the hostile environment. Ryan's tactical calculations dictate that the ease of neutralizing an enemy is contingent upon two factors: the distance between him and the target and the square root of the enemy's ammunition supply. He aims to eliminate the easiest targets first before confronting the more formidable adversaries, thereby maximizing his chances of success and minimizing potential risks. To facilitate Ryan's mission, a function named `helpRyan(points, sniperPoint)` is devised. This function returns the sequence in which the targets are to be eliminated, adhering to Ryan's strategic principles of engagement.

Examples

```
Input: points = [[1, 1, 1000], [2, 2, 10], [3, 4, 1], [100, 100, 1]] sniperPoint=[0,0]
Output: [[2, 2, 10], [3, 4, 1], [1, 1, 1000], [100, 100, 1]]
```

Explanation: As per ryan's calculation the target at point `[2,2]` is the easiest to eliminate and the one at `[100,100]` is the hardest



```
Input: points = [[1, 1, 1], [1, -1, 1]] sniperPoint=[0,0]
Output: [[1, 1, 1], [1, -1, 1]]
```

Explanation: If there is a tie we can kill the one that appears first in the list first.

Constraints

- Solve the question $O(n \cdot log(n))$.
- Do not use any inbuilt sorting functions.

Function

```
def helpRyan(points, sniperPoint):
    return []
```

Question 2

Jack is a blind porter who is searching for clay objects he spilled on the floor. Fortunately, Jack has a machine capable of identifying the size of these clay objects. Unfortunately, due to a power failure, the program aiding Jack got erased from memory. Whenever Jack needs a particular-sized object, he describes it as the "5th largest" among those on the floor, relying on the machine to retrieve it. Your task is to help Jack identify the object he seeks. Implement the function `jacksMachine(objects,query)` to handle Jack's query and return the size of the requested object. If retrieval isn't possible, return -1. By completing this function, you'll enable Jack to efficiently locate the objects he needs amidst the clay clutter on the floor.

Examples

```
Input: objects=[3,4,5,6] query=2
Output: 5
```

Explanation: The second largest object on the floor is of value 5.

```
Input: objects=[3,4,5,6] query=6
Output: -1
```

Explanation: There are only 4 objects on the ground and none of them could be the sixth largest.

Constraints

- Solve the question using the concept quickselect.
- Do not use any sorting functions.

Function

```
def jacksMachine(objects,query) -> int:
    pass
```

Question 3

In a serene village lived Ethan, a young mathematician renowned for his intellect. One day, an elder named Alden sought Ethan's help with a farming dilemma. They had an array representing their harvest and needed to divide it into three contiguous subarrays while minimizing the cost, with the cost defined as the value of the first element in each subarray. Now, drawing inspiration from Ethan's ingenuity, it's your turn to tackle the same problem.

Examples

Example 1:

```
Input: harvestArray = [1, 3, 4, 2, 1, 10, 8]
Output: 4
```

Explanation: Three subarrays with minimum costs are [1, 3, 4], [2], and [1, 10, 8]. The first element represents the cost of the subarray; here, the total cost is 4.

Example 2:

```
Input: harvestArray = [5, 1, 3, 2, 6, 8]
Output: 8
```

Explanation: Three subarrays with minimum costs are [5], [1, 3], and [2, 6, 8]. The first element represents the cost of the subarray; here, the total cost is 8.

Constraints

- harvestArray.length >= 3
- Solve the question using the heap data structure concept.
- Solve the question in O(n) time complexity.

Function

```python
def subarrayWithMinimumCost(harvestArray: list[int]) -> int:
    pass
```

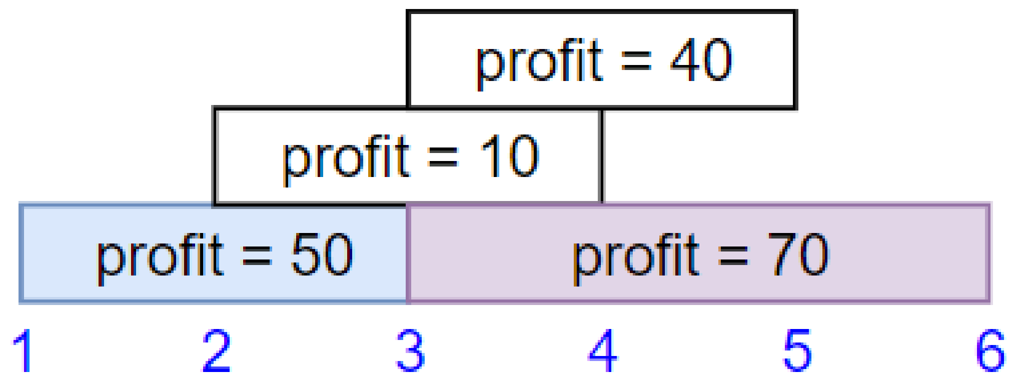
Question 4

Rahul Jaykar, a talented singer faced a plethora of opportunities, with invitations pouring in for him to showcase his musical prowess at various events and gigs across the town. Rahul was presented with an exciting challenge: how to schedule his performances in a way that would maximize his earnings without missing any events. We have n events, where every event is scheduled to be done from startTime[i] to endTime[i], obtaining a profit of profit[i].

As Rahul sifted through the invitations, he imagined a puzzle where he had to fit in all the pieces perfectly. he wanted to make sure that no event overlapped with another, allowing him to sing his heart out at each one. Now, how can Rahul ensure he earns the most money without any event clashes?

Examples

Example 1:



```
Input: startTime = [1,2,3,3], endTime = [3,4,5,6], profit = [50,10,40,70]
Output: 120
```

Explanation: The subset chosen is the first and fourth event. Time range [1-3]+[3-6] , we get profit of 120 = 50 + 70.

Constraints

- Time complexity should not exceed $O(NlogN)$
- $startTime[i] < endTime[i]$

Function

```
def maxEarnings(start_time, end_time, profit):
  return 0
```

Question 5

You've stumbled upon a magical garden filled with unique flowers. You're given a 1-indexed array called "blossomCosts", where blossomCosts[i] represents the cost to acquire the ith flower.

The garden has a special offer: when you acquire the ith flower at blossomCosts[i], you unlock the opportunity to pick the next 'i' flowers for free.

Note that even if a flower 'i' is available for free, you still have the option to acquire it at blossomCosts[i] to activate an offer of the next 'i' flower for free.

What's the minimum cost required to collect all the flowers from this garden?

Examples

Example 1:

```
Input: blossomCosts = [4, 1, 3, 5, 6, 3]
Output: 7
```

Explanation: Purchase the first and third flowers to collect all the flowers using the offer.

Example 2:

```
Input: blossomCosts = [1, 2, 5, 3, 2, 6, 1, 1, 10]
Output: 5
```

Explanation: Purchase the first, second, and fifth flowers to collect all the flowers using the offer.

Constraints

- Solve the question using the dynamic programming concept.

Function

```
def minimumCost(blossomCosts: list[int]) -> int:
    pass
```

Question 6

You are given n balloons, indexed from 0 to n - 1. Each balloon is painted with a number on it represented by an array nums. You are asked to burst all the balloons.

If you burst the ith balloon, you will get `nums[i - 1] * nums[i] * nums[i + 1]` coins. If i - 1 or i + 1 goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it.

Return the maximum coins you can collect by bursting the balloons wisely.

Examples

```
Input: [3,1,5,8]
Output: 167
Explanation:
       [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []
coins =  3*1*5    +   3*5*8   +  1*3*8  + 1*8*1 = 167
```

Constraints

- Time complexity should not exceed $O(N^3)$

Function

```
def maxCoins(nums):
  return 0
```

Question 7

Santa aimed to transmit an encoded message to Banta. Now, Santa has visited the GHC and intends to share the details with Banta. This time around, they've opted for Huffman coding to encode their text. Your assistance is required to help them implement this plan.

Both Santa and Banta have agreed upon a common class structure for this task.

```
class Huffman():
    def __init__(self):
        self.huffman_codes = {}
        self.source_string = ""
```

```python
    def set_source_string(self, src_str):
        self.source_string = src_str

    def generate_codes(self):
        huffman_codes = {}

        # Write your code here

        self.huffman_codes =  huffman_codes

    def encode_message(self, message_to_encode):

        encoded_msg = ""

        # Write you code here

        return encoded_msg

    def decode_message(self, encoded_msg):

        decoded_msg = ""
        return decoded_msg
```

First, they will compute the frequencies of characters in the source_string, which is stored in the `source_string` attribute (Note: The autograder sets this attribute). Subsequently, they will invoke the `generate_codes` function to create Huffman codes, which will be stored in the `huffman_codes` attribute within the Huffman class. To encode or decode any message, they will utilize the `encode_message` and `decode_message` functions, respectively, leveraging the `huffman_codes` attribute.

Your help is pivotal in making their Huffman coding implementation a success.

Example

```
 if source_string = "cbadeeee"
 generate_codes() will create a huffman codes.
 the huffman tree will look like
                                    [abcde: 8]
                                    /        \
                                [abcd: 4]    [e: 4]
                                 /      \
                            [ab: 2]     [cd: 2]
                           /     \        /    \
                        [a: 1] [b: 1]   [c: 1] [d: 1]


As the rule says left childs append 0s and right child append 1s
the huffman_codes attribute will be
```

```
{
    a: '000',
    b: '001',
    c: '010',
    d: '011',
    e '1'
}

now if encode_message('ae') is called it will return '0001'
and if decode_message('0100011011') is called it will return 'cbed'
```

Constraints

- All the strings will only contain lowercase English alphabets
- `encoded_msg` will only contain `'0'`s and `'1'`s

Note

- While generating the Huffman codes, if two or more symbols have same frequency choose the one that is lexicographically smaller. (This is to make sure you have same codes as that of autograder)
- When you move to a left child, append '0' to the current Huffman code. (This is to make sure you have same codes as that of autograder)
- When you move to a right child, append '1' to the current Huffman code. (This is to make sure you have same codes as that of autograder)
- Do not touch the `__init__` and `set_source_string` functions
- `generate_codes` function will generate huffman codes for all the alphabets and store them in the `huffman_codes` attribute
- `encode_message` will encode the input strring using the codes from the `huffman_codes` attribute which was set by `generate_codes`
- `decode_message` will decode the input strring using the codes from the `huffman_codes` attribute which was set by `generate_codes`

Question 8

Let A =[a1, a2, a3, ...an] be a given sequence of integers. Implement the wavelet tree data structure to support efficient rank and select queries. Rank returns the count of the desired element in the given range.

Reference

1. Refer Lecture 14 - Page 20
2. You can also refer the following paper to understand how to implement https://arxiv.org/pdf/0903.4726.pdf

Note

1. The `get_wavelet_level_order` function returns all the levels using level order traversal of the Wavelet tree as list of lists.

2. If the node is a leaf node, return ‘X’ as the value, based on the count of the element. Look at the examples for better understanding.
3. The rank query returns the count of the given character in the given range of the array. If such a character is not present in the given range, return 0.

Example

Example 1:

```
wv_tree = Wavelet_Tree([6, 2, 0, 7, 9, 3, 1, 8, 5, 4])
```

```
wv_tree.get_wavelet_level_order()
```

```
[['1001100110'], ['00101', '00110'], ['100', '01', '010', '10'],
    ['01', 'X', 'X', 'X', '10', 'X', 'X', 'X'],['X', 'X', 'X', 'X']]
```

```
wv_tree.rank(7, 3) # 0
```

Explanation: There is no 7 in the first 3 elements.

Example 2:

```
wv_tree = Wavelet_Tree([6, 2, 0, 7, 7, 9, 3, 1, 8, 5, 4])
```

```
wv_tree.get_wavelet_level_order()
```

```
[['10011100110'], ['00101', '000110'], ['100', '01', '0110', '10'],
    ['01', 'X', 'X', 'X', '10', 'XX', 'X', 'X'], ['X', 'X', 'X', 'X']]
```

```
wv_tree.rank(7, 5) # 2
```

Explanation - There are two 7’s in the first 5 elements. The two 7’s are correctly represented by two X’s in the leaf node over 4th level.

Constraints

1. Each integer is between 0 and 9.
2. 1 <= position <= number of elements in the array
3. The rank query should only be implemented using the built wavelet tree in log(n) time complexity.
4. The leaf nodes should represent the correct number of X’s denoting the count of the element.

```
# Feel free to change the class Node as per your requirement
class Node:
    def __init__(self, data, left=None, right=None):
        self.data = data
        self.left = left
        self.right = right

class Wavelet_Tree:
    def __init__(self, A:list[int]=[]):
        pass
```

```
def get_wavelet_level_order(self):
    # Return level order traversal of the tree.
    pass

def rank(self, character, position):
    #Return the rank of the given character in the given position range.
    pass
```