Assignment 06

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

There is a unique game arranged in the town where different people shout with all of their strength. There are multiple people on ground and you have to choose single person who is going to shout. You are provided with the coordinate position of all the people participating along with the radius their shout can be audible. Upon one person shouting, if another individual is within the audible radius, that person becomes deaf and subsequently starts shouting, causing others within their radius to go deaf and trigger a chain reaction of shouting. Also, the person who starts the shouting will also go deaf. choose a person which makes maximum number of people deaf and return that maximum number.

Examples

Example 1:

```
Input: PeopleShout = [[1,2,1],[10,10,2]]
Output: 1
```

Explanation: First person is present on coordinate (1,2) with audible radius of 1 unit. Second person is present on coordinate (10,10) with audible radius of 2 unit. No shout from either of the person will make other person deaf because their shout won't be heard by the other person.

So, only the person shouting will go deaf and the answer will be 1

Example 2:

```
Input: PeopleShout = [[2,1,3],[6,1,4]]
Output: 2
```

Explanation: If first person shouts, second person can't hear their shout. So if first person shouts, only they will get deaf resulting in only 1 deaf person.

But, if second person starts shouting, then their shout can be heard by the first person as they are inside the radius of second person. Here 2 people will go deaf.

So the maximum 2 people can go deaf if second person shouts and hence the answer is 2

Constraints:

- Solve the problem using DFS approach

Function

```
def MaximumDeafPeople(PeopleShout):
  #return int
  return maximumDeafCount
```

Question 2

Charlie and Devon are college students who are planning to take a train trip over winter break, but are unsure of where to go. They have obtained a brochure that contains the costs of trips from `n` different train stations. The costs are organized in an `n x n` matrix called `costs`, where the element at position `(x,y)` represents the cost of a train ticket from city `x` to city `y`. Write a function that computes and returns a list of the cheapest prices to travel from the `start` city to every other city in the list. This information will help Charlie and Devon to decide on the best travel destination for their winter trip.

Example:

Example 1:

```
Input: n = 2, start = 0, costs = [[0,50],[20,0]]
output = [0, 50]
```

Explanation: Since youare starting from index 0, the price to get to destination 0 is 0 as you are already there. The cheapest trip to destination 1 is a direct path which costs 50.

Example 2:

```
Input: n = 3, start = 2, costs = [[0,50,40],[20,0,10],[60,5,0]]
output = [25, 5, 0]
```

Explanation: Since youare starting from index 2, the price to get to destination 2 is 0 as you are already there. The cheapest trip to destination 1 is a direct path which costs 5. The cheapest trip to destination 0 is to go first to index 1, then go from index 1 to 0. This costs 5 + 20 = 25.

Contraints:

- Solve this problem using methods taught in class.
- Note that if `costs[x,y] = 0`, there is no path from index `x` to index `y`.

Function:

```
def cheapest_path(n, costs, start):
  #return list of ints
  return prices
```

Question 3

While taking a leisurely stroll in the enchanting forest, you stumble upon a concealed chest tucked away beneath a dense canopy of trees. To your intrigue, the chest is securely locked and adorned with inscriptions of multiple unique strings of letters. Accompanying these inscriptions, you discover a mysterious note bearing the message, "To unveil the secrets within, seek the smallest possible super-string where every word etched on this chest finds its refuge."

To solve this cryptic puzzle, you are tasked with crafting a code that can autonomously generate this elusive superstring. You'll be provided with a list of sub-strings, denoted as 'words', and your mission is to create the shortest possible superstring that encompasses every word found on the chest's inscriptions.

Example:

Example 1:

```
Input: words = ['XYY', 'YYX']
output = 'XYYX'
```

explanation: The strings from arr are represented in the output as [XYY]X and X[YYX]

Example 2:

```
Input: words = ['XYYZXY', 'ZXYXXXY', 'XYZZX']
output = 'XYYZXYXXXYZZX'
```

explanation: The strings from arr are represented in the output as... [XYYZXY]XXXYZZX, XYY[ZXYXXXY]ZZX, AND XYYZXYXX[XYZZX]

Contraints:

- Solve this problem using a greedy algorithm.
- You may assume that all test cases given will have one unique solution.

Function:

```python
def generate_password(words):
  #return string
  return password
```

Question 4

You are visiting the museum with your friends. You reached to the entrance of mysterious room in the museum where the room is divided into different parts with different heights. Your group has members with different heights and you can only enter the different parts of room if your height is less than or equal to the height of that part of room because your group has arthritis and no one can bend. Also, only one person is allowed to stand in the one part of room. Furthermore, If the height of some part in the room is less than the height of a person, then that person and all other person behind it will be stopped before that part of room and they won't be able to move forward. Also, you can only enter the room from left end as it is the entrance. It might be the case that not everyone can enter the room given the above condition. As you already paid the price for ticket, your guide asks your group to discuss with each other and come up with the strategy so that maximum number of people can enter the room. You and your group can see the heights of different parts of the room before you enter into the room. Can you tell us how many maximum person from your group can visit the room.

Example 1:

```
Input: personHeight = [1,1,1], roomHeight = [2,2,2]
output = 3
```

explanation: All people have height 1 and all can enter the different parts of room. For example, 1st person will occupy the last part of room, 2nd person can occupy last second part of the room and 3rd person can occupy the 1st part of the room

Example 2:

```
Input: personHeight = [1,1,1,1], roomHeight = [2,2,2]
output = 3
```

explanation: All people have height 1 and all can enter the different parts of room, but note here that room only has 3 parts so only any of the 3 person can enter into the room each occupying each separate part as only one person can occupy one single part and not the multiple persons.

Example 2:

```
Input: personHeight = [4,1,8,7], roomHeight = [7,4,2,2,5]
output = 3
```

explanation: All people have different heights. Person with height 1 can enter first and occupy 3rd part with height = 2, person with height 4 can enter after that and occupy the 2nd part of room with height = 4, and person with height 7 will enter at the last and can occupy first part of room with height = 7. The person with height 8 can't occupy any part of the room and hence maximum 3 people can enter the room.

Constraints

- Solve it using greedy approach

Function:

```
def maximumPeople(personHeight, roomHeight):
  #return int
  return maximumNumberOfPersons
```

Question 5

In a quaint town nestled between rolling hills, there exists a mysterious palindrome, a string of lowercase English letters with an enigmatic secret. As the townsfolk delved into the code that governed their digital existence, they uncovered a programming puzzle veiled in the palindrome. The challenge was to meticulously replace just one character within the string, ensuring the resultant sequence was no longer a palindrome and stood as the lexicographically smallest variation possible. This town, known for its ingenious programmers, awaited the coder (you) who could crack this puzzle and unveil the hidden magic within the once-palindromic spell.

Examples

Example 1

```
Input: "abcddcba"
Output: "aacddcba"
```

Explanation: Out of all the ways in which the input string can be made a non-palindrome, like “bbcddcba” or “abcdacba”, the lexicographically smallest one is “aacddcba”.

Example 2

```
Input: "aaabaaa"
Output" "aaabaab"
```

Constraints

- Solve it using greedy approach
- Time complexity - O(n), where n is the length of the input string

Function

```
def smallestString(encoded: str) -> str:
  # Your code here
  return decoded
```

Question 6

Jeremy Clarkson, Richard Hammond, and James May embark on an adventure in Antarctica. Their goal is to drive their cars in a lattitudinal circle around the continent. They have been given a list of checkpoints where they can rest and halt during their trip. The list mentions which of the checkpoints are connected serially. In the interest of time, they do not want to retrace the path between two checkpoints. Can you help them determine if there exists a route using which they can visit each of the checkpoints exactly once and return to their starting point, thus completing the circuit?

Examples

Example 1

```
Input: [0, 1, 1, 1, 0],
       [1, 0, 1, 0, 1],
       [1, 1, 0, 1, 1],
       [1, 0, 1, 0, 0],
       [0, 1, 1, 0, 0]
Output: True
```

Explanation: They travel from Checkpoint 0 to Checkpoint 1, Checkpoint 1 to Checkpoint 4, Checkpoint 4 to Checkpoint 2, Checkpoint 2 to Checkpoint 3, and Checkpoint 3 to Checkpoint 0. Hence, they are able to visit each of the checkpoints exactly once and return to their starting position.

Example 2

```
Input: [0, 1, 1, 1, 0],
       [1, 0, 1, 0, 1],
       [1, 1, 0, 1, 0],
       [1, 0, 1, 0, 0],
       [0, 1, 0, 0, 0]
Output: False
```

Explanation: Checkpoint 4 can only be visited from Checkpoint 1. Once you visit Checkpoint 4, you cannot go to any other checkpoint without first travelling back to Checkpoint 1. Hence, their goal to visit all the checkpoints exactly once fails.

Constraints

- Solve it using graph

Function

```python
def grandTour(checkpoints):
  # return True or False
  return boolean
```