Home Work № 7

CSCI B505 Applied Algorithms 10/31/2022

Instructions

1. Under any circumstances the deadline will NOT be extended.
2. Code should be written in python language in a Jupyter notebook .
3. Unless mentioned in the questions, feel free to use built-in python functions
4. Upload Jupyter notebook and html file of the Jupyter notebook in canvas. No other forms submissions is considered as valid submission. Make sure to have the output of the required cells of the jupyter notebook and its html version before making submission.
5. Plagiarism is unacceptable and we have ways to find it. So do not do it.
6. There are four problems. Each problem is worth 25 points.
7. The code should readable with variables named meaningfully.
8. Write test cases wherever required so that they cover all scenarios.
9. students are expected to use the function signature given in the question or they'll get a penalty.

Problem 1

Given a vehicle of capacity k travels in a straight line(starts from point 0 and travels in positive direction). Given a list h of n values(each a list of size 2) where h[i][0] indicates the pickup point where the ith person intends to get on the vehicle and h[i][1] indicates the destination point where the ith person intends to get off the vehicle. Once the vehicle's capacity is reached, the driver doesn't allow more people to board the vehicle. Return the minimum number of seats the vehicle should have had in addition to what it already does so as to accommodate all the people along the way.

```
Example 1:
Input: 2, [[0,2],[1,2],[0,3],[2,3]]
Output: 1

Example 2:
Input: 3, [[8,10],[2,4],[7,11]]
```

```
Output: 0
```

Following the below given method signature

```
1
2 def additional_seats(k,h):
3     #Your code goes here
4     return ans
```

Problem 2

An XYZ shipping company, At the conclusion of each quarter, releases a list of the routes that received the most deliveries during that quarter. At the end of the year, we'll have four lists, one for each quarter. The shipping company wants to use these lists to show the board of directors the smallest range that includes at least one element from each of the four quarters so that they can use it to market their company with the fewest amount of resources possible and try to open a new warehouse in that range. For simplicity all lists would be in sorted order. Expected time complexity is O(nlogM) where n is the total number of elements present in M lists.

```
Ex-1:
 Input :[[ 3, 6, 8, 10, 15],[ 1, 5, 12 ],[ 4, 8, 15, 16 ],[ 2, 6 ]]
 Output: The minimum range is (4,6)

Ex-2:
 Input:  [[ 2, 3, 4, 8, 10, 15 ],[ 1, 5, 12 ],[ 7, 8, 15, 16 ],[ 3, 6 ]]
 Output: The minimum range is (4,7)
```

```
1 def minimum_range(lists):
2     ###### logic #####
3     #### logic ######
4     return tuple (a, b)
```

Problem 3

Given a string s, write a function 'encode' that returns the Huffman encoding of that string as a string of 0s and 1s and the Huffman dictionary as a tuple. Write another function 'decode' which

takes the encoded string and a Huffman dictionary and returns the original string. Note that the composition of encode and decode should be the identity function, i.e decode(*encode(s)) = s. You can use that fact to test your code.

```
Example:
>>> s = 'aabc'
>>> encode(s)
    ('001011', {'a': '0', b: '10', c: '11'})
>>> decode('001011', {'a': '0', b: '10', c: '11'})
    'Aabc'
```

```
1  def encode(s):
2      pass
3
4  def decode(s, d):
5      pass
```

Problem 4

You are given a list of 3 positive integers. You are allowed to make a decrement to any two of the three (non zero) integers in a single move. Your objective is to find the maximum number of moves you can play before you run out of valid moves (i.e there are less than two non-zero integers left).

```
Example 1:
    input = [2,4,6]
    output: 6
    Explanation:
    1st move: decrement 1st and the 3rd integer: list = (1, 4, 5)
    2nd move: decrement 1st and the 3rd integer: list = (0, 4, 4)
    3rd move: decrement 2nd and the 3rd integer: list = (0, 3, 3)
    4th move: decrement 2nd and the 3rd integer: list = (0, 2, 2)
    5th move: decrement 2nd and the 3rd integer: list = (0, 1, 1)
    6th move: decrement 2nd and the 3rd integer: list = (0, 0, 0)

Example 2:
    input = [4,4,6]
    output: 7
    Explanation:
```

```
1st move: decrement 1st and the 2nd integer: list = (3, 3, 6)
2nd move: decrement 1st and the 3rd integer: list = (2, 3, 5)
3rd move: decrement 2st and the 3rd integer: list = (1, 3, 4)
4th move: decrement 2nd and the 3rd integer: list = (0, 3, 3)
5th move: decrement 2nd and the 3rd integer: list = (0, 2, 2)
6th move: decrement 2nd and the 3rd integer: list = (0, 1, 1)
7th move: decrement 2nd and the 3rd integer: list = (0, 0, 0)
```

```python
def max_moves(nums: list) -> int:
    ####logic####

    #############
    return maxmoves
```