MET CS 566 - Analysis of Algorithms
Assignment 5 - 20 Points

Please note the honor code policy regarding this assignment. You may not discuss particular questions or discuss or transmit answers from the assignment with other people except for the MET CS 566 teaching team.

Tasks

1. **Dynamic Programming Tables (10 points)**: An investor wants to invest 10 million dollars in real estate. He can select properties from a list of 4 houses in 4 different cities in Massachusetts. He wants to purchase as much as possible, and he wants to maximize the expected profits that he can get when he is going to sell the houses in the next year.

Table 1: 4 houses with their house prices and their expected home value growth in the next year.

Houses	1. Newton	2. Waltham	3. Watertown	4. Weston
House Values in Millions	3	2	4	3
Expected profit in the next year %	2	3	3	2

Tasks:

- Task 1.1. Describe what are the sub-problems here and what is the size/count of the subproblems
- Task 1.2. Describe which inputs to the knapsack problem (value and size) correspond to the inputs in the home purchasing problem. (2 points)
- Task 1.3. Fill up the following dynamic programming table and check the maximum possible profits. (3 points)
- Task 1.4. What is the optimal set of houses to buy and invest the money in? (3 points)

$V[i,w]$	0	1	2	3	4	5	6	7	8	9	10
$i=0$											
$i=1$											
$i=2$											
$i=3$											
$i=4$											

2. **Dynamic Programming - City Planning. (10 points)**: A city planner is asked to organize grocery shops in a new city. The city has a straight-line main street that goes throughout the city. The city planner is asked where the city should provide permits to build new grocery shops so that people of the new city can have the shortest distance to their grocery markets. The population density is not constant along both sides of the main street. A higher density is around multiple city centers or crossroads alongside the main street.
Your task is to develop an algorithm, given the positions of the city centers and the number of grocery shops compute the least possible sum of all distances between each city center and its nearest grocery shops.

Input to your Algorithm:

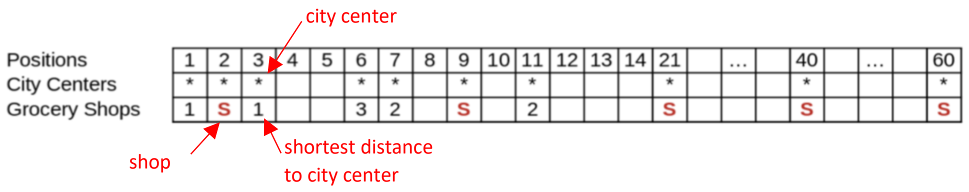
1. List of city centers coordinate positions along the main street (each an integer number between 1 and 1000). This is a sequence of numbers.

2. The number of city centers is an integer between 2 and 100.
3. The number of grocery shops (an integer number between 2 and 30).
4. The number of shops is smaller than the number of city centers.

The **output** of your algorithm: A single integer, which is the sum of all distances between each city center and its nearest grocery market.

Sample Input:
5
[1 2 3 6 7 9 11 21 40 50]

Sample Output:
9



A visualization of City Center Positions and Grocery Shops. (Numbers in grocery shop row is the distance to the nearest shop.)

Tasks:

- Task 2.1. What are the sub-problems in this case? What are the counts of sub-problems? Provide a brief description of your solution. (2 points)
- Task 2.2. Write up your algorithm in Pseudocode or python implementation. (6 points)
- Task 2.3. What is the run time complexity of your algorithm? (2 points)

The following is a python program template to help you with your DP algorithm. Some parts are replaced with three question marks "???".

```
def groceryShops(A, k):
    # Write your code here
    if not A:
        return 0

    n = len(A)  # n is the number of city centers

    if k >= n:  # we should not have more shops than city centers
        return 0

    A.sort()
    cost = [[0 for _ in range(n + 1)] for _ in range(n + 1)]

    """
    cost[i][j] is the minimum cost to build a shop between city centers i
and j
    This shop should be in the median of the city centers.
    """
```

```
for i in range(n):
    for j in range(i, n):
        mid = int((i+j)/2)

        for r in range(i, mid + 1):
            cost[i + 1][j + 1] += A[mid] - A[r]
        for r in range(mid + 1, j + 1):
            cost[i + 1][j + 1] += A[r] - A[mid]


"""
dp[i][j] is the minimum cost for the first j city centers with i shops.
"""
dp = [[float('inf') for _ in range(n + 1)] for _ in range(k + 1)]


dp[0][0] = 0

for i in range(1, k + 1):
    for j in range(1, n + 1):
        for r in range(j):
            dp[i][j] = min(dp[i][j], dp[i-1][r] + ???)

return dp[-1][-1]
```