Algorithms Laboratory (CS29203)
Lab Test - I
Department of CSE, IIT Kharagpur

5th October 2023

Question-1 (40 points)

The FITJEE coaching institute of Kharagpur is arranging for some prize distribution among their students who have secured good rank in JEE Advanced this year. Before the JEE Advanced results came out, they had already purchased the prizes of different types. Now they realized that the number of students who have cleared the mains, are actually very low than what they had expected, and they have purchased way more prizes than the qualified students. So now they don't have any option but distribute all the prizes among the few number of qualified students. However they just don't want to give the prize items randomly to any student, rather follow some rules as below:

- One student can be given at most one type of prize (but can be given any quantity of it).
- After distributing the prizes, each student will have some quantity of prizes (p) of one particular type. They want p to be as small as possible. Basically they want to minimize the maximum number of prizes given to any student.

Let us consider an example. Let there are 6 qualified students and there are two types of prizes having quantities 11 and 6 respectively (we can denote this as an array $[11, 6]$). Now the optimal way of distributing the prizes is the following. The 11 prizes of first type are given to the first four students in these quantities: $2, 3, 3, 3$. The 6 prizes of second type are given to the other two students in these quantities: $3, 3$. Then the maximum number of prizes given to any student is, $\max(2, 3, 3, 3, 3, 3) = 3$.

Your task is to develop and implement an algorithm to solve the above problem. For getting the full credit, your algorithm should have worst case complexity of $O(n \log m)$ where n is the number of students and m is the maximum number of available quantities. (*Hint: think in the idea of simple binary search*).

Example:

```
(Input) n = 7, prizes = [15,10,10]
(Output) 5

Explanation: One optimal way of distribution:
- The 15 quantities of first type are given to the first three students in these quantities: 5, 5, 5
- The 10 quantities of second type are given to the next two students in these quantities: 5, 5
- The 10 quantities of third type are given to the last two students in these quantities: 5, 5

So the maximum number of prizes given to any student is, max(5, 5, 5, 5, 5, 5, 5) = 5.
```

Question-2 (30 points)

Let there are n cities connected by road networks. The cost of travelling between i-th and j-th city is denoted as $d(i,j)$. Assume that $d(i,j) = d(j,i)$ for all i and j. Also, $d(i,i) = 0$ for all i. Our goal is to find a closed route (cycle) through the cities such that each city is visited only once, and the total travel cost is minimized. Assume that the tour starts (and ends) at city 0.

Actually this is a very difficult computational problem where no polynomial-time algorithms exist for solving this. In this exercise you will be implementing a greedy solution which runs fast but the solution is not guaranteed to be optimal. Before discussing the greedy approach, let us look at how to prepare the data for n cities. First, read n from the user, and generate each city as a point (x, y) in the plane. You may take a random value between 0 to 999 as each of the coordinates. After cities are generated, store the distances $d(i,j)$ between pairs of cities in a $n \times n$ 2D matrix.

Now the greedy approach works as follows. A possible greedy approach to construct a suboptimal solution is to iteratively add one city at a time to the tour. This is an approximate solution to the given problem. Initially choose city #0 as the starting point. While there are unvisited cities, select the nearest unvisited ciy from the current city and add it to the path. Update the current city to the newly visited city. Once all the cities have been visited, return to the starting city by adding the distance from the last visited city to the starting city. At the end, output the final path and the total cost.

Your task is to implement the greedy algorithm described above. Note that the following example is just for reference for the input and output format, you might get a different solution on the same data presented in the example.

Example:

```
n = 12
City coordinates: (21,993), (36,490), (59,101), (87,740), (193,997), (286, 19), (561,131),
(737,100), (821,614), (856,246), (958,136), (962,967)

Final path:   0->1->2->5->6->7->10->9->8->11->3->4->0
Cost: 4082
```

Question-3 (30 points)

Suppose you are given a sequence of positive numbers separated by + and * signs, for example: $6*3+2*5$. You can change the value of this expression by adding parentheses in different places. For example, $(6*3)+(2*5) = 28$, $6*(3+(2*5)) = 78$, $(6*(3+2))*5 = 150$. Hence, for this expression, the optimal ordering is to add the middle numbers first, then perform the multiplications. Similarly, for the expression $0.1*0.1+0.1$, the optimal ordering is to perform the multiplication first, then the addition: $(0.1*0.1)+0.1 = 0.11$. Write a Dynamic Programing algorithm to compute the maximum possible value of a given expression of the form $a_1\ op_1\ a_2\ op_2 \cdots op_{n-1}\ a_n$ by adding parentheses. $a_1, a_2, \cdots, a_n$ are n numbers and $op_1, op_2, \cdots, op_{n-1}$ are the $n-1$ operators in between. Assume all the numbers in the input are positive and only + and * are the valid operators.

Hint:

1. Since you are asked to solve this using Dynamic Programing, think about a table where each cell contains the solution of the subproblems and the solution to the bigger subproblems uses the solution to the smaller ones. The order in which the table can be filled up has similarity with the Matrix Chain Multplication problem we discussed in the class.
2. What are the base cases?
3. To solve the subexpression involving $a_i, \cdots, a_j$, can you split it into two subproblems at the k^{th} operator, and recursively solve the subexpressions involving $a_i, \cdots, a_k$ and $a_{k+1}, \cdots, a_j$?

Some note about input/output:

1. Take the number a_i's as float array and operators op_i's as char array. As the operators are characters, you need to be clever to use them as operators. You can take any way to resolve this.
2. You don't need to print the paranthesis positions in the answer. The maximum value is to be printed though.
3. You can take the maximum number of numbers in the expression as 10.
4. If you need to take maximum value of the expression, you can take it to be 10000. For both of these, a good coding practice is dfine proper macros.

Example 1:

```
Number of positive numbers (max value 10): 4
Enter 4 numbers from left to right:
6
3
2
5
Enter 3 operator from left to right (+, *):
*
+
*
Max value of expression is: 150.000000
```

Example 2:

```
Number of positive numbers (max 10): 3
Enter 3 numbers from left to right:
0.1
0.1
0.1
Enter 2 operator from left to right (+, *):
*
+
Max value of expression is: 0.110000
```