ASSIGNMENT 1: AUTOMATED MARKET OPENING SCHEDULER DURING COVID

Goal: The goal of this assignment is to take a complex new problem and formulate and solve it as search. Formulation as search is an integral skill of AI that will come in handy whenever you are faced with a new problem. Heuristic search will allow you to find optimal solutions. Local search may not find the optimal solution, but is usually able to find good solutions for really large problems.

Scenario: Optimization of Market Opening Schedule.

A city has ***n*** types of shops. The government wants to create an opening schedule for the markets ensuring the safety of maximum people. Due to the current COVID situation the government wants the people to make minimum movement out of their houses. They have approached you to take your help in order to organize the opening of shops in a best possible schedule. You need to use the power of AI and write a generalized search algorithm to find the best possible schedule.

The city has ***m*** market places which can be opened parallely. In a market place during each time slot the government is planning to open ***k*** types of shops. And in a day there are a total of ***T*** time slots available. We can assume that ***n = T.m.k***.

For example, in figure below m = 2, T = 3 and k = 4

Type: 1,2,3,4	Type: 5,6,7,8	Type: 9,10,11,12
Type: 13,14,15,16	Type: 17,18,19,20	Type: 21,22,23,24

We first define the characteristics of a good schedule. For any good schedule people should make minimum movement and most of the people should feel no conflict about which market they should go for purchasing.

That is:

1. all types of shops opening in one time slot in the same market should sell related items (items generally bought together)
2. all types of shops opening in parallel markets should be as far away as possible to avoid people's movement to all of the markets (selling items that are generally not bought together)

To operationalize this intuition let us assume we are given a function representing the distance between two types/categories: d(t1, t2), such that d is between 0 and 1. We can similarly define a similarity between two, s(t1, t2) = 1 - d(t1, t2).

Now we can define the goodness of a schedule as follows:
Sum(similarities of all pairs within a single time slot in the same market) + C.Sum(distances of all pairs within a single time slot in the parallel market).

In our example, the goodness will be computed as

G(Schedule) = s(1,2) + s(1,3) + s(1,4) + s(2,3) + s(2,4) + s(3,4) + s(5,6) + s(5,7) + s(5,8)
+ s(6,7) + s(6,8) + s(7,8) + ……. + s(13,14) + …. + s(21,22) + …..
+ C x [d(1,13) + d(1,14)… d(2,13) + d(2,14) + … + d(5,17) + d(5,18) + …]

The constant ***C*** trades off the importance of semantic coherence of one market versus reducing conflict across parallel markets.

Your goal is to find a schedule with the maximum goodness.

Input:
Line 1: k: total types of shops opening in one time slot in one market
Line 2: m: number of parallel markets
Line 3: T: number of time slots
Line 4: C: trade-off constant
Starting on the fifth line we have a space separated list of distances between a type of shop and rest others. Note that d(x,y) = d(y,x). Also, all d(x,x) = 0.

Sample Input:

```
2
2
1
1
0 0.4 0.8 1
0.4 0 0.6 0.7
0.8 0.6 0 0.3
1 0.7 0.3 0
```

Output: Your algorithm should return the max-goodness schedule.

Output Format:
Space separated list of shop ids (i.e, shop's type ids), where time slots are separated by bars and parallel markets are separated by line.

For the above problem the optimal solution is t1 and t2 in one market; and t3 and t4 in the other market. It will be represented as:

1 2
3 4

Other equivalent ways to represent this same solution:

4 3
2 1

OR

2 1
3 4

etc. All are valid and have the total goodness of 4.4 (Verify).

Another sample input is provided along with the assignment representing similar easy problems. We recommend you to experiment with other problems as well.

Important Instructions:

1. You may work in teams of maximum three or by yourself. If you are short of partners, our recommendation is that this assignment is quite straightforward and a partner should not be required.
2. You cannot use built-in libraries/implementations for search or scheduling algorithms.
3. Please do not search the Web for solutions to the problem. Your submission will be checked for plagiarism with the codes available on Web as well as the codes submitted by other teams. Any team found guilty will be awarded a suitable penalty as per IIT rules.
4. Your code will be automatically evaluated. You get a zero if your output is not automatically parsable.
5. You are allowed to use any of the two programming languages: C++, Python.

What to submit:

1. In google classroom submit a zip file named in the format <RollNo>.zip. If there are two members in your team it should be called <RollNo1_RollNo2>.zip and so on. Your zip file should contain followings:
    a. <RollNo>.py: Source code, your solution to the problem developed
    b. <RollNo>.txt: 1 page write-up containing details about the search strategy you followed. Details about heuristics (if designed any) etc.

2. Submit your code on the hackerrank platform (link will be shared soon). You need to sign up on hackerrank using your iitj e-mail id.

Note:

1. At both the places one submission per group is required.
2. Fill your group details in the following google sheet: AI-1 Assignment-1 Group Details

Evaluation:
Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Note that the performance of your method will not be disclosed until the late submission deadline.

Late submission deadline and penalty:
After the deadline, maximum achievable marks will be reduced by 20%. It means if you submit 5 days later to the deadline, zero marks will be awarded. This also applies to the re-submissions which are done past the Submission deadline.