ASSIGNMENT 1

Goal: The goal of this assignment is to take a complex new problem and formulate and solve it as a 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 Conference Schedules
Assume that a conference has n papers accepted. Our job is to organize them in the best possible schedule. Let the
schedule have **p** parallel sessions at any given time, where each session has **k** papers, and there are a total of **t** time slots. We can safely assume that **n = (t * p * k)**. For example, as per the figure below t = 3, p = 2 and k = 4.

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

Now a good schedule is one where most people would feel no conflict about which session to attend => (1) all papers in a session should be related to a single theme and (2) all papers in parallel sessions should be semantically as far away as possible to avoid conflicts.

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 session) + C.Sum(distances of all pairs within a single time slot in the parallel session).

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*** tradeoff between the importance of semantic coherence of one session versus reducing conflict across parallel sessions.

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

Input:
Line 1: k: the number of papers per session
Line 2: p: number of parallel sessions
Line 3: t: number of time slots
Line 4: C: trade-off constant
Starting on the fifth line, we have space-separated lists of distances between every paper.
[Note: dist(x, y) = dist(y, x), and dist(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 as found within the desired time limit. The output format is a space-separated list of paper ids, where time slots are separated by bars, and parallel sessions are on different lines.

For the problem shown above, the optimal solution is clearly p1 and p2 in one session; and p3 and p4 in another. This will be represented as:
0 1
2 3
Other equivalent ways to represent this same solution are:
1 0
2 3 or
3 2
0 1 or
2 3
1 0 (etc)
All of these alternatives are equally valid.
Verify that for this problem the total goodness is 4.4.

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. Your code will be automatically evaluated. You get a zero if your output is not automatically parsable.
4. You are allowed to use the Python programming language.
5. 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.

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. 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.