# ASSIGNMENT 1: SPORTS COMPLEX PLANNING

**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 a new planned city

A new large sports complex is being built in Delhi. It will have all the highest end equipment and courts with lots of zones. A zone will house one type of court/equipment. Example, there will be different zones for basketball courts, table tennis rooms, cardio gym, weight training, yoga center, swimming pool and so on. Different sportspersons will have different typical schedules of when they do swimming and when they practice in a court and so on.

Based on typical interest of sports persons and children in Delhi (example, more people want to be cricketers than volleyball players), some zones will be used more and some will be used less. Moreover, because it is a large complex, the zones are far from each other, so there will be walking times involved from one zone to the other based on a user's personalized schedule. Our goal is to identify which zone should be at which part of the complex so that (on average) time spent walking can be reduced.

For that, we assume that we have Z: a set of z zones numbered {1,2,…,z} and L: a set of l locations numbered {1,2,..,l} such that (z<=l). We define two matrices T (lxl) and N (zxz). An entry $t_{ij}$ will represent the time taken to walk from location i to location j. On the other hand, the entry $n_{xy}$ will represent the number of daily walks from zone x to zone y. Note that $t_{ij} = t_{ji}$ , though this is not true for matrix N. Also, $t_{ij} = 0$ if i=j.

Your goal is to output a mapping from Z to L such that the total time walking in a day is minimized.

## Input:

The first line has total processing time available in minutes.

The second line has z: the number of zones

The third line has l: the number of locations

Starting fourth line we have the matrix N one row at a time. Assume all times are non-negative integers.

Starting line 4+z, we have the matrix T one row at a time. Assume all entries to be non-negative integers.

Here is a sample input

2

3

4

2 5 3

2 4 1

3 4 5

0 3 4 1

3 0 4 1

4 4 0 8

1 1 8 0

## Output:

Your algorithm should return the best allocation of zones to locations, such that one zone is in one location and one location has only one zone. The output format is a sequence of Z unique numbers, all less than equal to L. Example, if you decided that in the above problem zone 1 goes to location 2, zone 2 goes to location 4 and zone 3 goes to location 1, you should output:

2 4 1

The total time to walk for this permutation is 30.

## Basic Algorithms you can try:

1. Heuristic Search: Design a state space and transition function for this problem. Define a heuristic function for evaluating a state. Implement A* (or variants) and measure quality of solutions found (and scalability). If heuristic is admissible – quality is optimal but algorithm may be slower. Test a couple of heuristics if you can design them.

2. Branch and Bound: Design a state space and transition function for this problem. Optionally define a heuristic function for evaluating a partial walk. Also, optionally, define a ranking to pick the next successor. Implement Depth First Branch and Bound (or variants) and measure scalability.
3. Local search: Implement a neighbor function for this problem. Implement local search (or variants). Measure of quality of best solution found as a function of time or other model parameters.

Recommended: start with local search as your base algorithm.

## Sample Code:

You are being provided sample code that can take in the input and generate the output in C++. You may choose to not use this code. You may program the software in any of C++, Java or Python. The versions of the compilers/interpreters that will be used to test your code are

**JAVA:** java version "11.0.20" (OpenJDK)
**Python** 3.9.17
**g++** 9.4.0

The sample code (written in cpp) can be downloaded here. This code reads in an input file and computes the cost of the allocation. You need to write the additional logic for allocation of resources. Here, we have allocated the resources sequentially. The allocation is then written in the output file. You may or may not use the sample code. Note that you need to implement logic to compute allocation in given time. You can compile and run the sample code as follows:

make

./main <input_filename> <output_filename>

We have even supplied formatchecker.jar which checks the format of the output file generated and gives the cost (if the format is valid). To run the format checker, use the following command:

./format_checker <input_filename> <output_filename>

We have provided three input files representing easy problems. We recommend you experiment with other problems as well.

## What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" the following files are produced in the working directory:
   compile.sh
   run.sh
   writeup.txt

   You will be penalized for any submissions that do not conform to this requirement.

   Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 20.04. You are already provided information on the compilers on those VMs. They have RAM of 8 GB

   Your run.sh should take 2 inputs, someinputfile.txt and someoutputfile.txt. It should read the first input as input and output the answer in the outputfile.
   ./run.sh input.txt output.txt ( please note any name could be given to the two files).

2. The writeup.txt should have two lines as follows
   First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.
   Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.
   After these first two lines you are welcome to write something about your code, though this is not necessary.

**Code verification before submission:** 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. Failure to follow any instruction will incur significant penalty.

We shall be generating a log report for every submission within 12 hours of submission. This log will let you know if your submission followed the assignment instructions (format checker, scripts for compilation & execution, file naming conventions etc.). Hence, you will get an opportunity to resubmit the assignment within half a day of making an inappropriate submission. However, please note that the late penalty as specified on the course web page will still apply for resubmissions beyond the due date. Exact details of log report generation will be notified on Piazza soon.

Also, note that the log report is an additional utility in an experimental stage. In case the log report is not generated, or the sample cases fail to check for some other specification of the

assignment, appropriate penalty for not adhering to the input/output specifications of the assignment will still apply at the time of evaluation on real test cases.

**Evaluation Criteria:** Performance, i.e., quality of the allocations output by your method on a variety of test inputs. Extra credit may be given to standout performers.

**What is allowed? What is not?**

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Recall, that you can't repeat a team in COL333, and repeat a time only once in COL671. 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 or optimization algorithms.
3. While you are allowed one of three languages we recommend that you use C++ since it produces the most efficient code. This assignment requires you to produce the best solution within a given time constraint.
4. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
5. Please do not search the Web for solutions to the problem.
6. Please do not use ChatGPT or other large language models for solutions to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.
7. Your code will be automatically evaluated. You get a minimum of 20% penalty if your output is not automatically parsable.
8. We will run plagiarism detection software. Any team found guilty of either (1) sharing code with another team, (2) copying code from another team, (3) using code found on the Web, will be awarded a suitable strict penalty as per IIT and course rules.