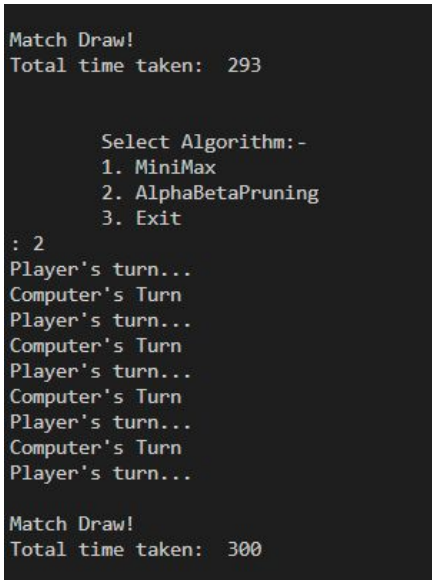
# Programming Assignment 2

**Report**
**Kaustav Vats (2016048)**

**Question 1:**
User Vs Computer Tic-Tac-Toe, using Minimax & Alpha-beta Pruning
Both return same answer. Minimax make calculation for every node in the tree, whereas Alpha beta prevent calculating node value which are 100% guaranteed to NOT be an Optimal State for MAX. In worst case both explore same number of nodes. Inshort Alpha beta ignore branches on the game tree that don't contribute to an optimal solution.
Sometimes we get an equal node value in game tree. Equal values doesn’t give better result and there’s no point in exploring further.
Minimax explores every node on the game tree.

Performance for both the algorithm were compared based on the **Time taken** by the algorithm and no of **recursive calls** made by the algorithm.
Difference is very minimal since the matrix size is very less.
Theoretically **Alpha beta Pruning** perform better than **Minimax**. Sometime result showed that Alpha beta is better than Minimax, but the difference is minimal for both ways. Image attached below



**Question 2:**

A) For TimeTable Scheduling Problem:

- **No of Courses(M): 32**
  Each Course Lecture happens once a day.
  Courses further divided into
  - **16 Courses 3** lecture per week
  - **16 Courses 2** lecture per week

- **No of Lecture Halls(N): 2**
  Classes happen parallelly for a particular time slot

- **No of Professors(P): 16**
  Each professor teaches 2 Courses.
  Professor cannot teach two courses at the same time.

- **Each Lecture class duration is 1 hour**

Number of Iteration after which we can expect an optimal solution - 500
Population size taken 100
Best selected from the population 10
No of new chromosomes created from best selected parent 9

B) **Gene** represent
<Course, Professor, RoomNo, Day, TimeSlot>
Where Room no can be [R1, R2]
Days can be [Monday, Tuesday, Wednesday, Thursday, Friday]
Timeslot [8-9, 9-10, …, 3-4]

**Chromosome** consist of Array of Genes
<A, P1, R1, Monday, 8-9>
<B, P2, R2, Monday, 8-9>
<C, P3, R1, Monday, 10-11>
…so on.

**Mutation -** Randomly select any one of the underlined attribute from gene and change it.

**Crossover -**

- Take two chromosomes
- Select genes which have same **Course and Professor**
- Then switch other underlined attributes to create two new chromosomes.

**Evaluation function:**
Evaluation function is strictly based on no of conflicts. Lesser the conflicts much better is the chromosome.

Conflicts are added based on

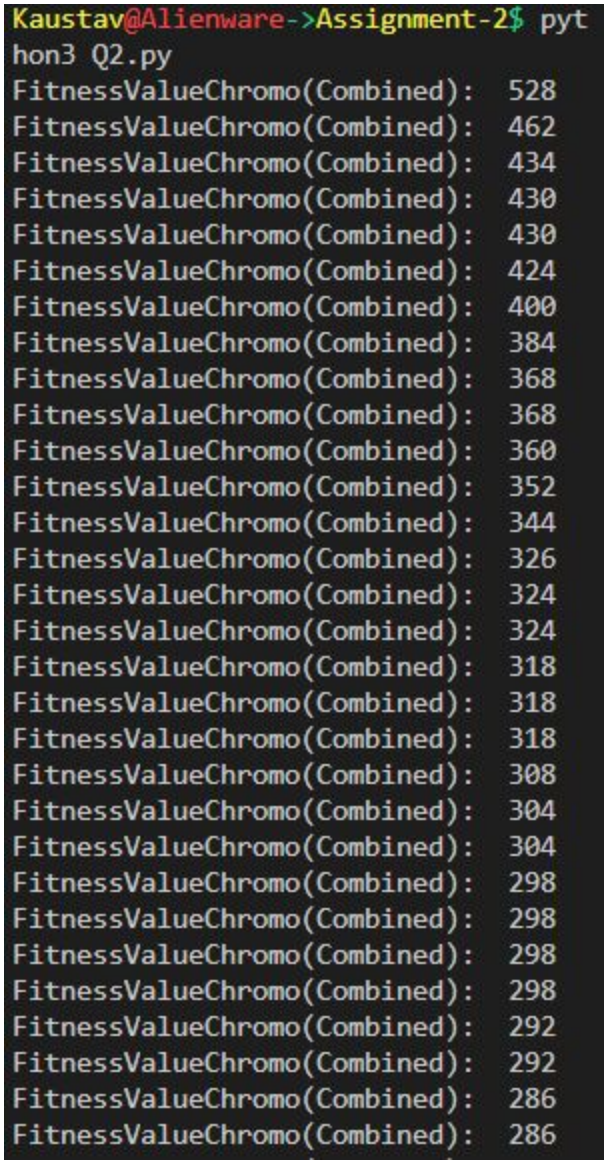
1) If any course occur more than once for a particular day. Increase the total number of conflicts by 1.
2) If Any professor have two classes scheduled at the same time slot on same day. Increase the conflict value of chromosome by 1.
3) Check the number of courses scheduled for same slot on same day and same room. Increase the number of conflict by number of clashes.

**Local Optimisation-**
Removing conflict by randomly picking new value for the particular attribute.
For eg- There are two courses which clash with same time slot, same day and same room number. So in local optimisation i will randomly pick any underlined attribute and change it to some new value.
But how will you check for constraints?
By iteratively storing First occurring course details in dictionary and later comparing it with same course details. Similar technique for other constraints.
*Check localOptimisation function in MemeticAlgorithm class in Q2.py*

MA Approaches solution much faster because of local optimisation whereas GA takes time. MA gradually decreases conflicts then again slowly reduces.
Whereas GA slowly reduces conflicts.

```
Kaustav@Alienware->Assignment-2$ pyt
hon3 Q2.py
FitnessValueChromo(Combined):  306
FitnessValueChromo(Combined):  252
FitnessValueChromo(Combined):  218
FitnessValueChromo(Combined):  218
FitnessValueChromo(Combined):  224
FitnessValueChromo(Combined):  220
FitnessValueChromo(Combined):  212
FitnessValueChromo(Combined):  206
FitnessValueChromo(Combined):  200
FitnessValueChromo(Combined):  184
FitnessValueChromo(Combined):  184
FitnessValueChromo(Combined):  186
FitnessValueChromo(Combined):  184
FitnessValueChromo(Combined):  182
FitnessValueChromo(Combined):  178
FitnessValueChromo(Combined):  178
FitnessValueChromo(Combined):  176
FitnessValueChromo(Combined):  170
FitnessValueChromo(Combined):  164
FitnessValueChromo(Combined):  164
FitnessValueChromo(Combined):  164
FitnessValueChromo(Combined):  164
```

Result for MA. No of Conflicts are reducing much faster than the below GA.




These are the result for GA.

You can see difference the Conflict value is decreasing after each iteration.
And also the rate of decrease for MA is high as compared to GA. But after some time the slowly reduces.

CSP can be solved using Backtracking.
Firstly i created a 3D matrix. Where 1 D represent no of days in a week, 2D represent no of time slots available and 3D represent no of rooms available.
File the matrix by first filling rooms then for each day.