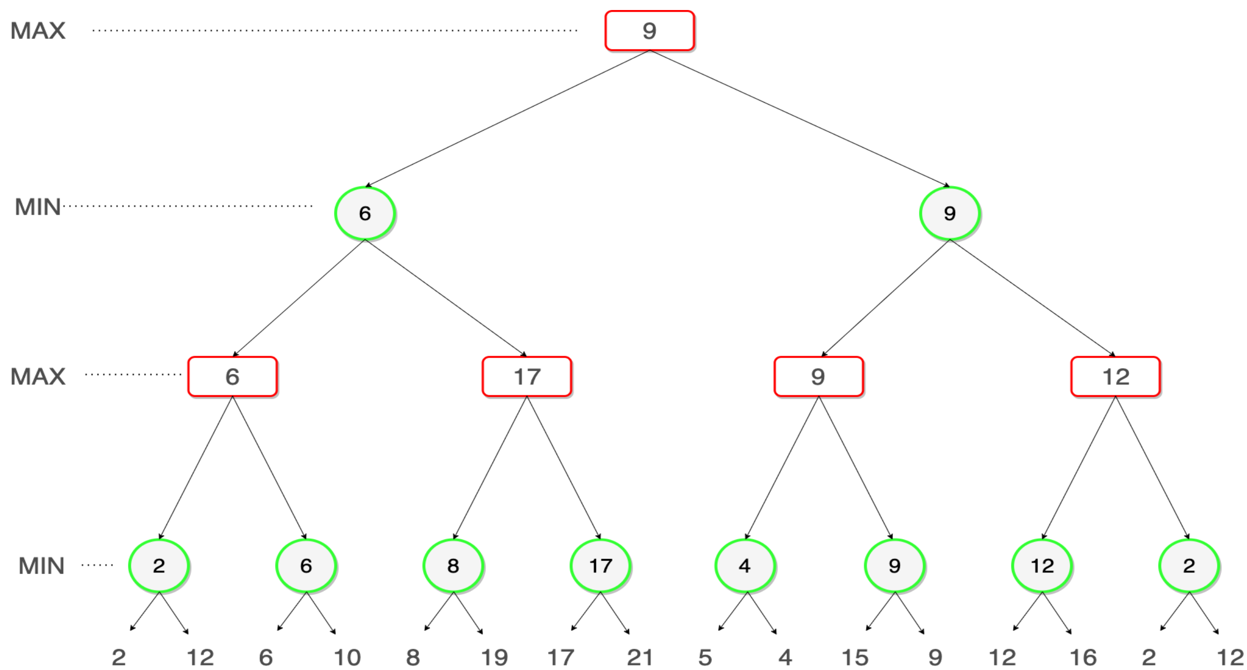
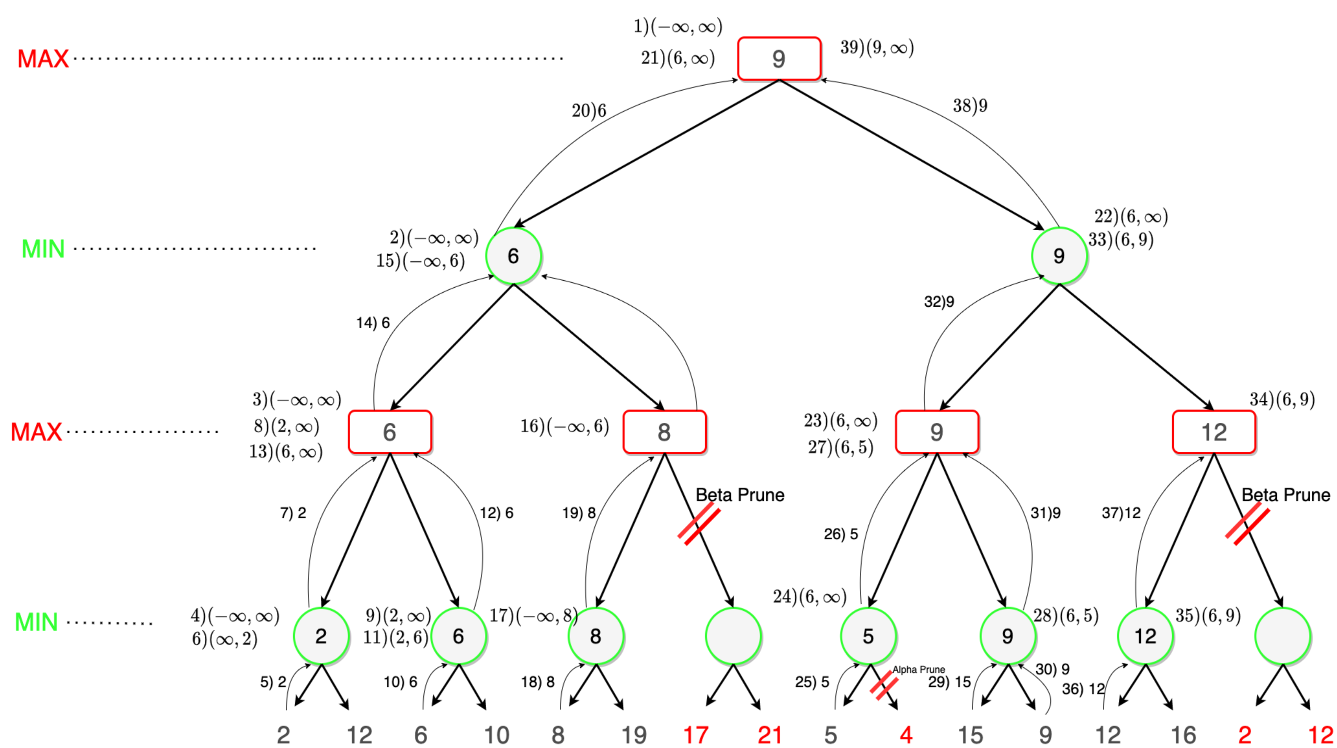
<u>AI Assignment 2</u>

1. Adversarial Models

1. Min-Max Search

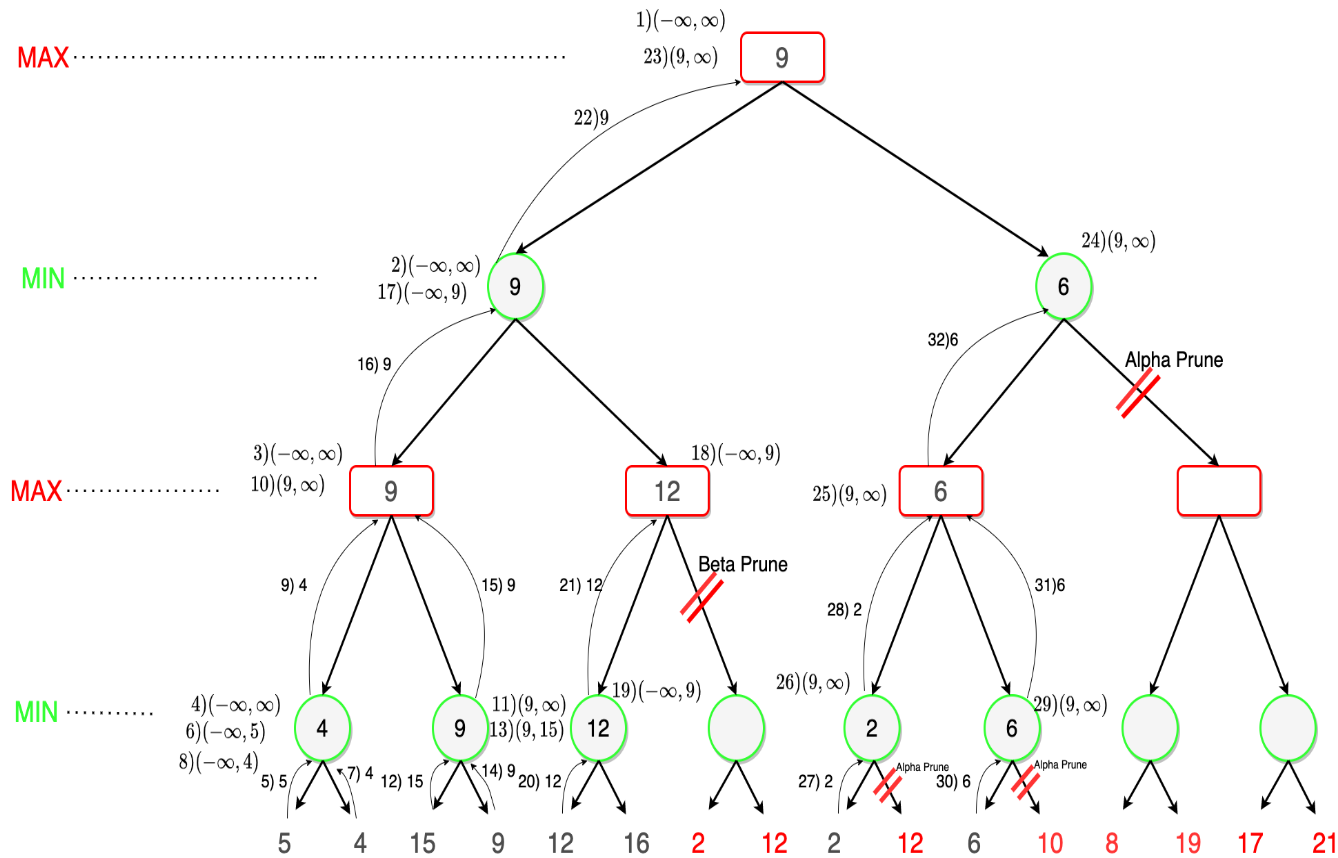


2. Min-Max Alpha/Beta Pruning



Each and every step indicates an update in either a node or its corresponding alpha/beta values. And the total number of pruned nodes is **5.**

3. Yes if we shuffle the move order we can increase the number of items pruned. Here is an example:



From the above diagram, it is evident that upon changing the move order we can increase the number of items pruned. We got **8** elements pruned this time which is greater than the previous one.

The effectiveness of alpha-beta pruning highly depends on the move order in which each node is visited. The ordering can be of the below types:

- **Worst case ordering**: There are cases where the alpha-beta pruning algorithm does not prune any of the leaf nodes and works exactly as a min-max algorithm. Such cases have the worst ordering and the time complexity for such ordering is O(b^n). In such cases, the best moves occur on the right side of the tree.

- **Ideal case ordering:** Case in which a lot of pruning happens in the tree, and the best moves that occur to the left side of the tree is the ideal case ordering. We apply a Depth-first search algorithm to search in the left-most part of the tree and go deep twice as the min-max algorithm at the same time. The time complexity of such cases is O($b^{n/2}$) which makes a huge difference.

Rules to find good ordering:

- The best moves happen from the lowest node.
- Order the nodes in the tree in such a way, that the best nodes are picked first.
- Use domain knowledge while finding the best move.

2. **Constraint Problems**

A Golumb Ruler is a set of marks at integer at integer positions along a number line such that no two pairs of marks are at the same distance apart. The number of marks on the ruler is its order 'o', and the largest distance between two of its marks is its length 'l'.

1. **Variables** - The number of marks on the ruler are 'o' points. These 'o' points are considered as the variables ranging {P1, P2, P3, P4……………..Po}.

2. **Domain** - Each variable has a non-empty domain D, i.e each variable is within a domain of integers, X_i is in range of [X_0, X_0+ l], where p ∈ Z (Z is the set of all the integers, both positive and negative including zero),X_0 is the starting number in the interval and 'l' is the largest distance between two of its marks.

3. Constraints - For example all the below set of integers satisfy the Golumb Ruler Constraints.

[0,1,4,10,18,23,25]
[0,1,4,9,15,22,32,34]
[0,1,5,12,25,27,35,41,44]
[0,1,6,10,23,26,34,41,53,55]
[0,1,4,13,28,33,47,54,64,70,72]
[0,2,6,24,29,40,43,55,68,75,76,85]

On the number line all the integers are placed in the proper increasing order.
So from the above examples, as no pair of points has the same distance, considering quaternary constraint with four different variables.

$X_j - X_i = X_m - X_n$ ∀i<j; n<m and (i, j) != (m,n)

Considering auxiliary variables $D_{i,j} = X_j - X_i$ ∀i<j,

$D_{i,j} != D_{m,n}$ ∀i<j; n<m and (i, j) != (m,n)

3. Bandit Algorithms Report

Algorithm	**Exploration rate**	**Uniform Distribution Parameter**	**Decay rate**	**Reward function**	**Input Data File**	**Cummulative Reward**
STAT	0.5	0.5	0.5	0.2	Data1.csv	1899
STAT	0.5	0.5	0.5	0.4	Data1.csv	1998
STAT	0.5	0.5	0.5	0.6	Data1.csv	2160
STAT	0.5	0.5	0.5	0.8	Data1.csv	2193
STAT	0.5	0.5	0.5	1	Data1.csv	2345
STAT	0.5	0.5	0.2	1	Data2.csv	144322
STAT	0.5	0.5	0.4	1	Data2.csv	123010
STAT	0.5	0.5	0.6	1	Data2.csv	127421
STAT	0.5	0.5	0.8	1	Data2.csv	105015
STAT	0.5	0.5	1	1	Data2.csv	97473
STAT	0.5	0.2	1	1	Data1.csv	2274

STAT	0.5	0.4	1	1	Data1.csv	2237
STAT	0.5	0.6	1	1	Data1.csv	2231
STAT	0.5	0.8	1	1	Data1.csv	1983
STAT	0.5	1	1	1	Data1.csv	2441
STAT	0.2	1	1	1	Data2.csv	53664
STAT	0.4	1	1	1	Data2.csv	118001
STAT	0.6	1	1	1	Data2.csv	111443
STAT	0.8	1	1	1	Data2.csv	124090
STAT	1	1	1	1	Data2.csv	107469

Algorithm	**Exploration rate**	**Uniform Distribution Parameter**	**Input Data File**	**Cummulative Reward**
ROLL	0.5	0.5	Data1.csv	1982
ROLL	0.5	0.5	Data2.csv	124010
ROLL	0.5	0.2	Data1.csv	2234
ROLL	0.5	0.4	Data1.csv	2137
ROLL	0.5	0.6	Data1.csv	2031
ROLL	0.5	0.8	Data1.csv	1923
ROLL	0.5	1	Data1.csv	2445
ROLL	0.2	1	Data2.csv	79094

ROLL	0.4	1	Data2.csv	148932
ROLL	0.6	1	Data2.csv	121458
ROLL	0.8	1	Data2.csv	122821
ROLL	1	1	Data2.csv	88927

Algorithm	**Exploration rate**	**Uniform Distribution Parameter**	**Decay rate**	**Input Data File**	**Cummulative Reward**
REC	0.5	0.5	0.5	Data1.csv	2393
REC	0.5	0.5	0.2	Data2.csv	124322
REC	0.5	0.5	0.4	Data2.csv	133010
REC	0.5	0.5	0.6	Data2.csv	147421
REC	0.5	0.5	0.8	Data2.csv	125015
REC	0.5	0.5	1	Data2.csv	99473
REC	0.5	0.2	1	Data1.csv	2234
REC	0.5	0.4	1	Data1.csv	2347
REC	0.5	0.6	1	Data1.csv	2531
REC	0.5	0.8	1	Data1.csv	2003
REC	0.5	1	1	Data1.csv	2451
REC	0.2	1	1	Data2.csv	60064
REC	0.4	1	1	Data2.csv	119001
REC	0.6	1	1	Data2.csv	123443
REC	0.8	1	1	Data2.csv	153090
REC	1	1	1	Data2.csv	167469

Based on the above analysis, Yes, there is an impact in the cummulative reward based on the parameter values.

Exploration Rate - This increases the cumulative reward up to a specific point and may decrease then onwards turning out to be a exploitation rate.
Uniform Distribution - The cumulative reward may increase or decrease based on the changes in this parameter values.
Decay Rate - As this value increases on an average, cumulative reward decreases.
Reward Weight Function - As this value increases on an average, cumulative reward increases. This is only present in STAT algorithm.

ROLL bandit algorithm does not have any decay rate and reward weight function. Its performance solely depends on the feature scaling applied to its weights. And REC gives best performance as it keeps track of previous rewards with in a given window size and keeps learning from them at each step. All these algorithms fall under reinforcement learning and the performance order of the algorithms is **REC > ROLL > STAT**.