**Ajay Babu Gorantla**

# Artificial Intelligence

## Winter 2022 - Mid Term

## Solutions

**1)**

Historically, there are several reasons that led to the first AI Winter:

**Scientific standpoint**

From a scientific standpoint, one of the most important setbacks was after publication of "Perceptrons" by Minsky and Papert. They proved that the Rosenblatt's perceptron model can solve a problem only if it is linearly separable. One of the examples is the XOR problem. The Rosenblatt's perceptron model cannot solve this. However, they mentioned that this can be solved by using multiple layers of neural network. This big setback caused an AI winter until the "*backpropagation*" algorithm came into existence in 1985 (Only to realize that it was discovered way before "Perceptrons" was published).

**Computing Resources**

Implementing AI models requires a lot of computing resources. It was only from a decade or two ago, computing resources became more accessible. This led to a widespread development in the field of AI

**Money**

Developing the infrastructure needed to do research in AI is an expensive task. It requires lot of money. Often it needs to also be supported by Governments (particularly during the 1960s). For the governments to invest they needed a hope and promise that AI is feasible.

**2)**

**a)** Fully Observable

**b)** Single Agent

**c)** Deterministic

**d)** Discrete

**3)**

**(i)** False

**(ii)** True

**(iii)** True

**(iv)** False

**(v)** False

**(vi)** True

**(vii)** False

**(viii)** False

**(ix)** The cross point would be between 2nd and 3rd bit, this results in two individuals out of which one will have 6 bits of 1(s). This would work even when selection is made with crossover point between 3rd and 4th bit.

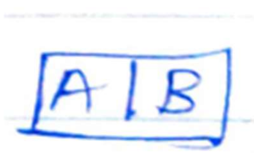
**4)**

**(i)** Yes, h(n) is admissible. h(n) < h*(n), where h*(n) is the true cost to reach the goal from node n (a state)

**(ii)** Yes, h(n) is consistent because for a node n, f(n) <= g(n')+h(n'), where n' are successor nodes of n.

**5)**

Let us consider the example of vacuum word with only two locations (A&B).




Let the task environment be:

- Performance measure awards one point for each clean square at each time step
- The geography is constant
- Actions available are left, right and suck
- Agent perceives its location and dirt presence precisely

In this task environment, every possible agent works rationally.

**6)**

Yes, the global coordination of simple agents is possible, so that the agents are coordinated to solve a problem with complexity beyond the capabilities of each individual agent.

The ‘Boids’ theory is a perfect testament to the above statement. For example, Birds swarm to form a pattern which has a lot of benefits for all the birds involved in the swarm. However, the problems they solve for themselves is far beyond the capabilities of each individual bird in those swarms

Another example is a group of robots working in tandem in the warehouses. They solve a larger problem even though each robot is very limited in capabilities.

**7)**

**(a)** $O(n^{2n})$ – (iii)

**(b)** $5^n$ – (iii)

**(c)** Although there are many possible admissible heuristics for this problem, one of them which we can use is the Manhattan Distance between the car’s current location and the goal.

$hi = | x_i - n | + |y_i - (n - i + 1)|$

**(d)**

$max\ \{h_1, \ldots .. h_n\} - (ii)$

$min\ \{h_1, \ldots . h_n\} - (iii)$

**8)**

**(i)** This holds true only when all the step costs are equal

**(ii)** This is true, because this just follows $f(n) = g(n)$ where as A* search uses $f(n) = g(n) + h(n)$. correlating it with Uniform Cost search it $h(n) = 0$.

**9)**

If we consider a state space such as a Linked List, where each node has at most one child.

With Depth First Search (DFS), to reach the goal state, it would take $O(n)$ time, assuming the goal state is at depth level ‘n’ .

With Iterative Deep Search (IDS), for first iteration it would traverse to depth 1, for second iteration till depth 2.. and so on. This would take $O(n^2)$ time.

Hence, in this search space, IDS performs much worse than DFS

**10)**

Since we already know that there is a solution whose true cost is K, we discard the solutions or do not expand nodes whose cost is greater than K. This algorithm indeed would maintain optimality because if a solution exists it gives us the best solution.

**11)**

**(i)**

The main idea behind Simulated Annealing Algorithm is that unlike a strict hill climbing algorithm where there is always a chance to get stuck in a local minimum, this algorithm aims to escape from that local minimum, thereby giving a chance to reach global minimum (if there is any).

For each iteration the temperature T is scheduled which controls the probability of downward steps

Next, instead of selecting a conventional stepwise successor of current (like in the case of strict hill climbing), a random successor of current is selected and assigned to next

Then the difference between next and current($\Delta E$) is evaluated to check if the objective function is going uphill and if true it keeps going. Else, next is assigned to current only with a probability of $e^{\Delta E/T}$

This process is iteratively done until the global minimum is reach/ the objective function is maximized

**(ii)**

In order to make it implement a strict version of hill-climbing, we need to avoid maintaining a temperature, and avoid selecting the successor randomly and instead select it the conventional way i.e. stepwise.

**(iii)**

The probabilities of accepting the mentioned moves are:

**Move 1**:

$\Delta E = -1, T = 20$

Probability = $e^{\Delta E/T} = 0.9512$

**Move 2**:

$\Delta E = -12, T = 25$

Probability = $e^{\Delta E/T} = 0.6187$

**Move 3**:

$\Delta E = -1, T = 276$

Probability = $e^{\Delta E/T} = 0.9963$

**12)**

**Initialization**

Populate with all possible different routes while ensuring that the route representation complies with the rules i.e. no city is visited twice which is nothing but no repetition of elements and no cities are to be skipped which is nothing but no omission of cities.

**Fitness Assignment**

In this case i.e. the Travelling Sales Man problem the fitness function would be how short the overall distance is

**Selection**

We can use the roulette wheel selection method while selecting the parents for creating the next possible sequence strings

**Crossover**

One issue that we face while dealing with crossover is ensuring valid representation. When we perform the usual crossover there is always a chance of repetition of elements which is not acceptable. Hence, we randomly select a subset of the first parent string and then fill the remainder of the route with the genes from the second parent in the order in which they appear, without duplicating any genes in the selected subset from the first parent.

**Mutation**

In this case as we cannot flip the elements (as in the case of strings with 0(s) and 1(s)), we essentially swap to elements of the string.

**13)**

Let Start node be ‘S’ and Goal node be ‘G’

**(i) Iterative deepening depth first search**

SSACDSABCEG

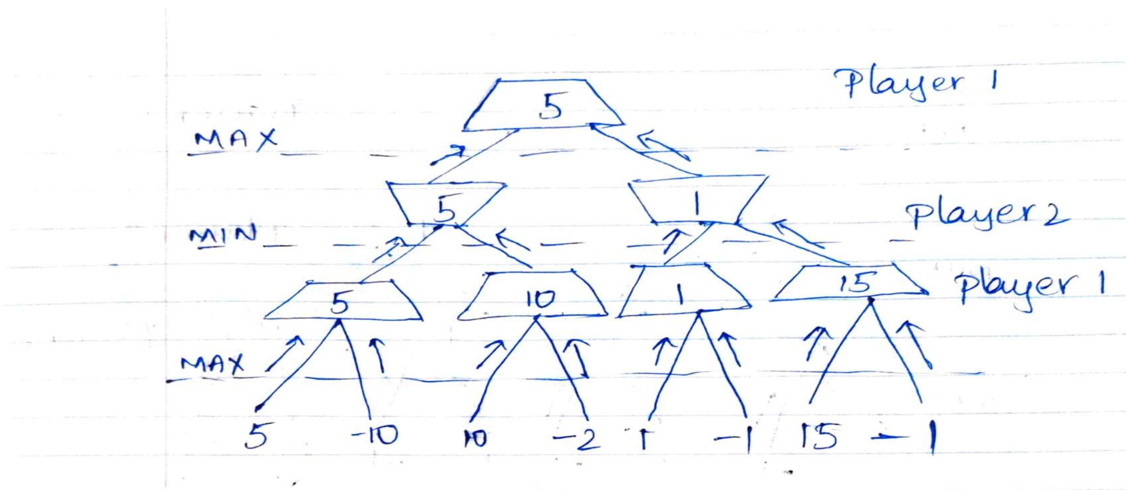
There is another path, via B and D but the above is the optimal one

**(ii) A* Search**

SDACG

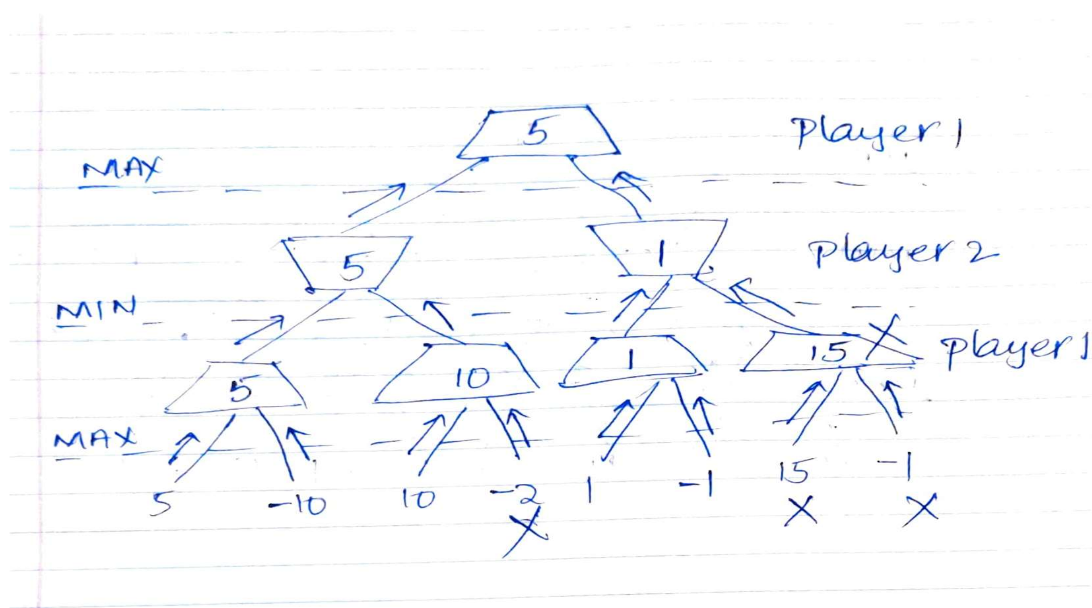
**14)**

**(a)**



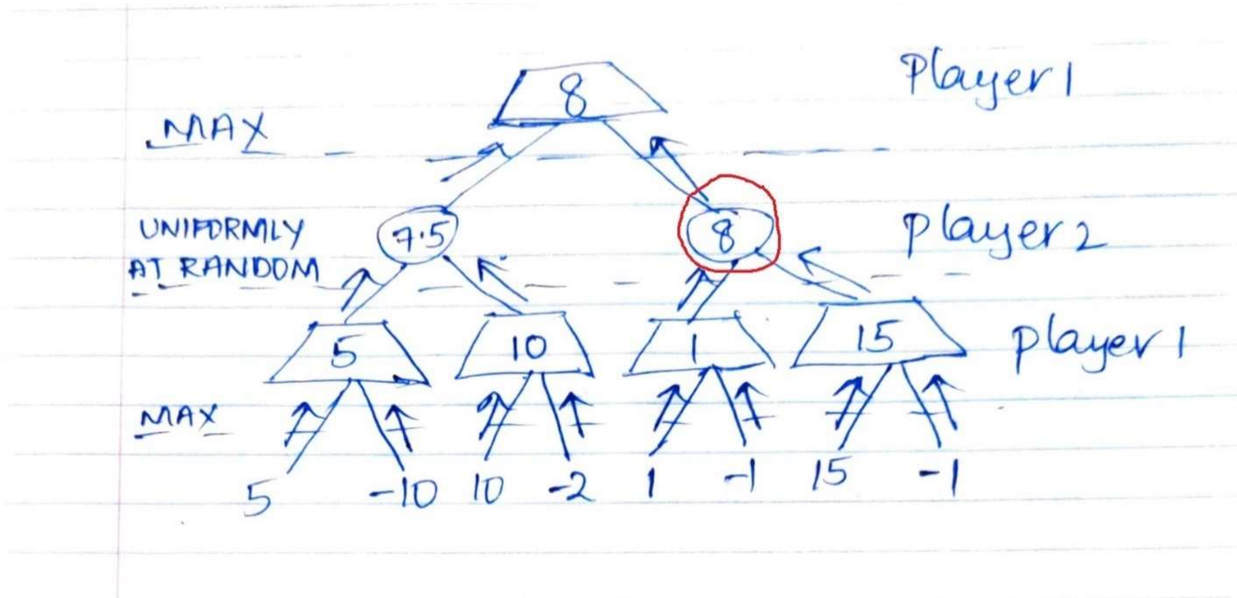
5
MAX
Player 1
5
1
MIN
Player 2
5
10
1
15
player 1
MAX
5
-10
10
-2
1
-1
15
-1


**(b)**



5
MAX
Player 1
5
1
MIN
Player 2
5
10
1
15
player 1
MAX
5
-10
10
-2
1
-1
15
-1

**(c)**




**Extra Credit**

The farmer can complete the challenge in exactly 7 steps.

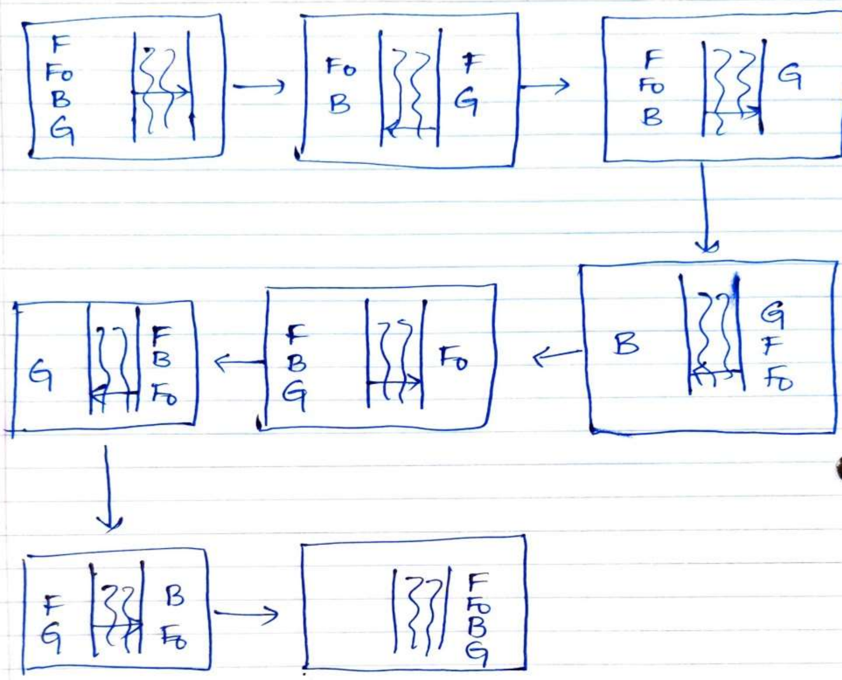
**The solution:**

- Farmer takes goose and crosses the river
- Farmer returns to the other bank
- Farmer takes the fox and crosses the river
- Farmer returns with the goose to the other bank
- Farmer takes beans and crosses the river
- Farmer returns to the other bank
- Farmer takes goose and crosses the river

Now the farmer has safely crossed the river keeping all his purchases intact

Let F denote Farmer, Fo denote Fox, B denote Beans, and G denote Goose.

The below is state transition diagram of the above mentioned solution


F
Fo
B
G
Fo
B
F
G
F
Fo
B
G
B
G
F
Fo
F
B
G
Fo
G
F
B
Fo
F
G
B
Fo
F
Fo
B
G