INDRAJEET SARAVANAN
Saravanan.8@osu.edu

HOMEWORK - 1

1 CONCEPT CHECK.

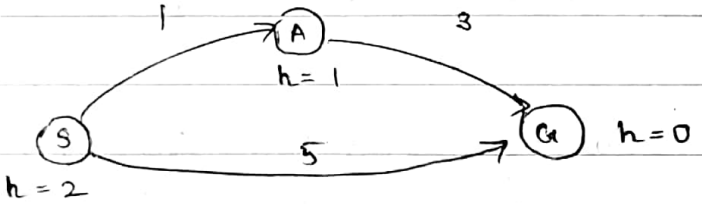
1. Data structures for collection of fringe / frontier nodes

BFS → FIFO

DFS → LIFO

A* search → Priority Queue.

2.



admissible and consistent heuristics.

$h(S) = 2 \leq$ actual cost (S to G)

$2 \leq 4$.

$h(A) = 1 \leq$ actual cost (A to G)

$1 \leq 3$.

$h(S) - h(G) <$ cost (S to G)

$2 - 0 \leq 4$

$2 \leq 4$.

$h(S) - h(A) \leq$ cost (S to A)

$2 - 1 \leq 1$

$1 \leq 1$.

$$h(A) - h(G) \leq \text{cost}(A \text{ to } G)$$

$$1 - 0 \leq 3$$

$$1 \leq 3$$

$$0 \leq h(m) \leq h^*(m)$$

$$0 \leq h(S) \leq h^*(S)$$

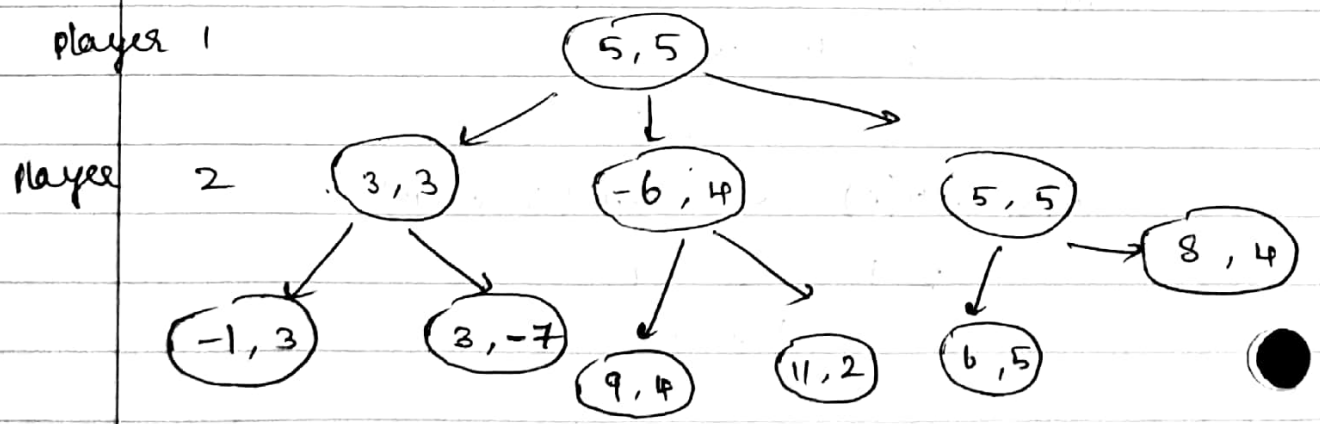
$$0 \leq 2 \leq 4$$

$$0 \leq h(A) \leq h^*(A)$$

$$0 \leq 1 \leq 3$$

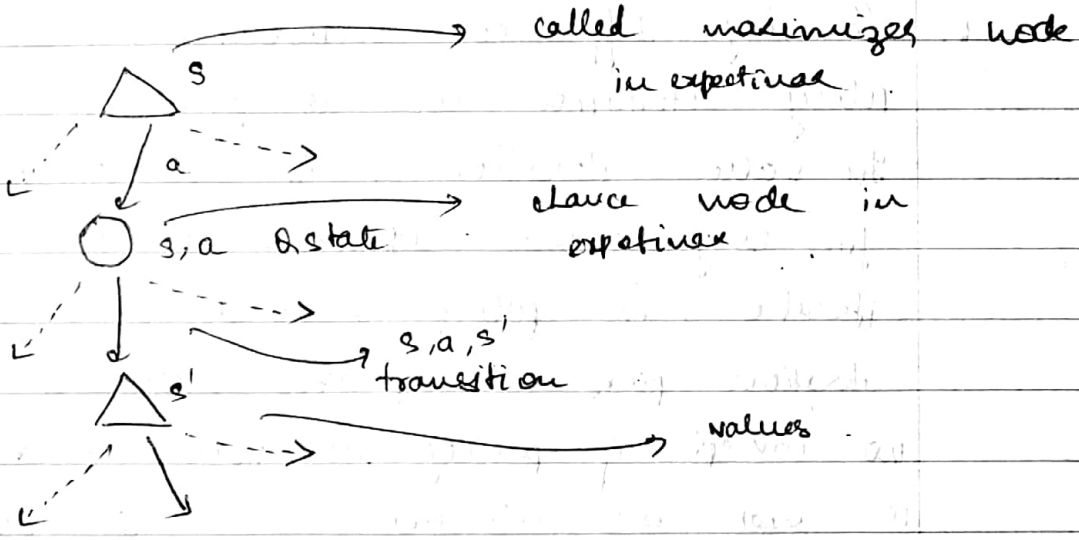
4.

3. In a non-zero game setting, we can't make use of alpha-beta pruning. This is because the players trying to maximize the values are independent. Which implies there is no more the case where one player tries to let down other player.



In the above tree, there is no minimizer node and only 2 player who play to maximize their values are present.

4. The relationship between expectimax algorithm and Q-states working in MDPs are as follows :



A maximizer node is the state(s) in MDP. The when we take an action (a) we land in Q state. We call this as a dance node in expetimax. A random transition will happen based on probability values and the resulting state s'.

5. The assumption that our decision processes are Markov based can be attributed to the inherent goal of MDP which is to formulate a policy that specifies an action to be taken with respect to current state. The Markov

property is necessary since the policy aims to increase the cumulative rewards as a function of current state. Hence, only the current state plays a role in deciding the outcome which is directly what we expect of a policy.

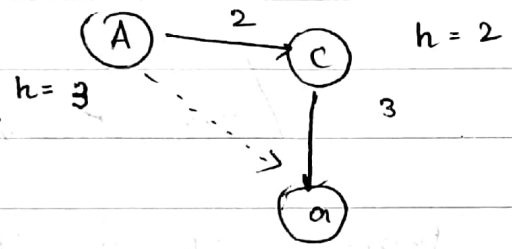
1. The key differences between value and policy iteration are as follows:
In value iteration, the values are updated for every round whereas in the policy iteration, the policy is fixed and multiple iterations pass in updating values. There is no concept of policy in value iteration, only the max over all values is the action to be taken. In policy iteration, all actions are taken as per the policy.

3 FUN WITH PROOFS

1. ADMISSIBILITY : $h(A) \leq$ actual cost from A to G
CONSISTENCY : $h(A) - h(C) \leq$ cost (A to C).

∴ All consistent heuristics will be admissible

This can be showed as follows :-



consistency is

$h(A) - h(c) \leq \text{cost}(A \text{ to } c)$

$\text{cost}(A \text{ to } c) = \text{cost}(A \text{ to } G) - \text{cost}(c \text{ to } G)$

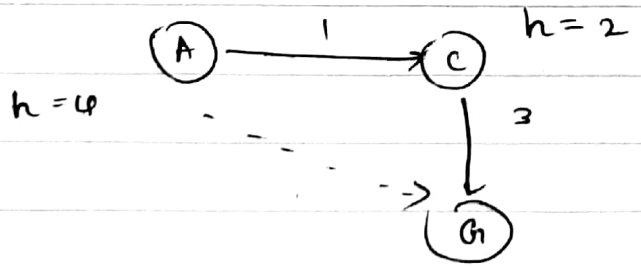
$\text{cost}(A \text{ to } c) = \underbrace{h(A) - h(G)}$ if they are same.

$h(A) - h(c) \leq \text{cost}(A \text{ to } G) - \text{cost}(c \text{ to } G)$

$h(A) \leq \text{cost}(A \text{ to } G)$

$h(c) \leq \text{cost}(c \text{ to } G)$

Automatically they turn out to be admissible however, not all admissible heuristic are consistent.

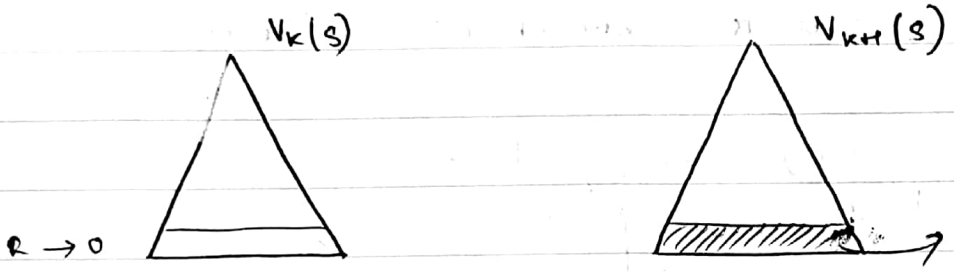


$h(A) - h(c) \leq \text{cost}(A \text{ to } c)$

$4 - 2 \leq 1$

$2 \not\leq 1$ Hence, proved.

2.



The only difference between V_K and V_{K+1} is the last layer which is

$$\left[V_K \times R'_{MAX} \quad (or) \quad V_K \times R_{MIN} \right]$$

$$V_{K+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_K(s') \right]$$

γ^K because for last step the discount factor is raised accordingly

When the discount is less than 1, the resulting values in the last layer will settle since as K approaches infinity the value will approach zero or constant.