CS 5100 01/27

HW1 extended till Wednesday 01/29

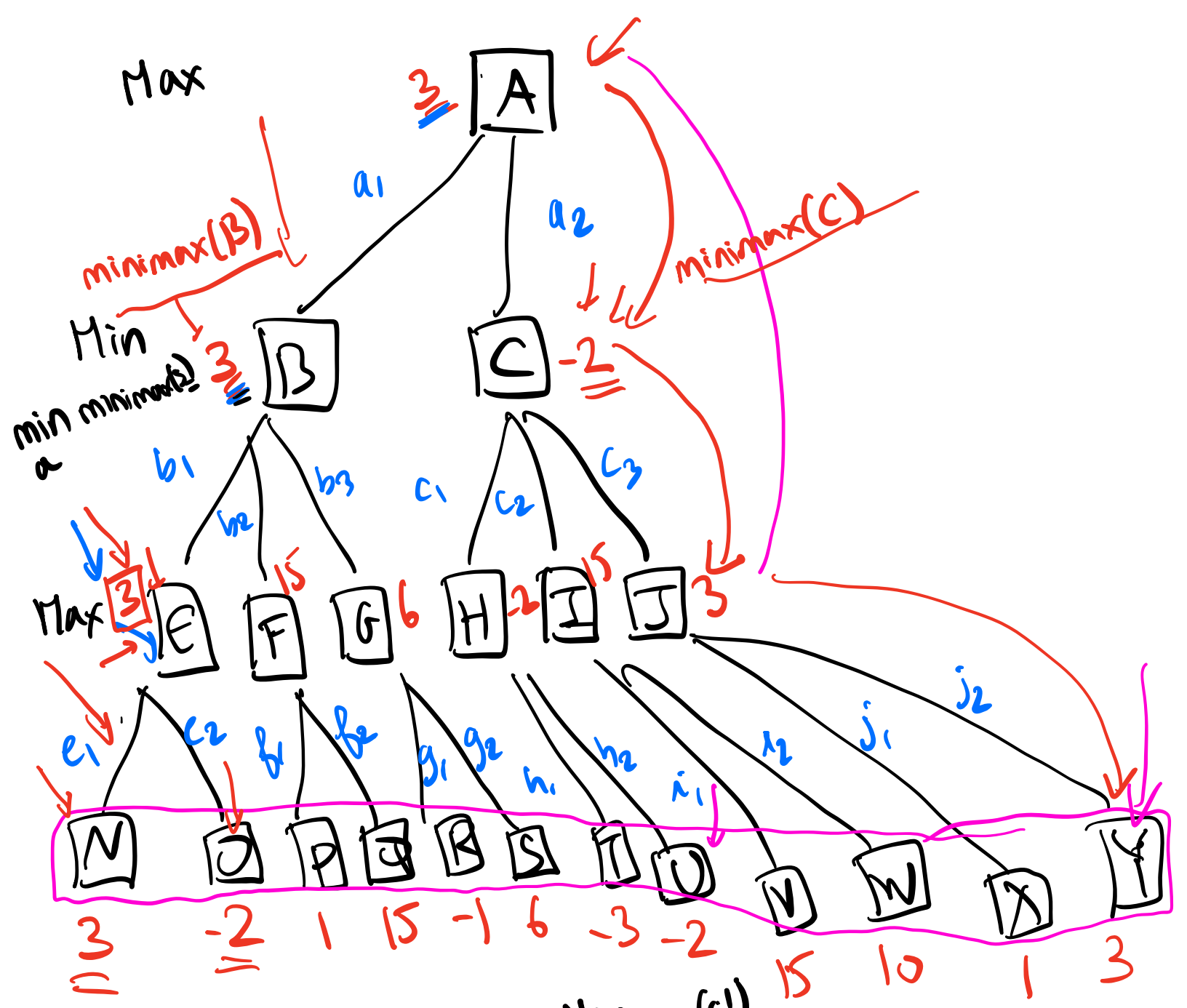
Review

Adversarial Games

* Start state
* Actions
* Terminal state (End/Goal)
* Result(s,a) (Transition model)
* Player(s)
* Utility function

Player 1 (Max) seek to maximize the utility

Player 2 (Min) " " minimize the utility function



Minimax(s) = Minimax(s')

$$\begin{cases} \min_{a:\,actions(s),\ s':\,succ(s,a)} & \text{if player is min} \\ \max_{a:\,actions(s),\ s':\,succ(s,a)} & \text{if player is max} \\ Utility(s) & \text{if } s \text{ is a terminal state} \end{cases}$$

value in red for any state in the tree above

max
a ∈ actions(s)
s': succ(s, a)

minimax(s')

Utility(s) is implemented.

Assume Result(s, a) is implemented and it returns the successor of s when applying action a

Code this algorithm

```
def action_chosen(state):
    '''
    Return the optimal action to take
    '''
```

```
def max_value(state):
    '''
    Return the max value we could get at state
    '''
```

```
def min_value(state):
    '''
    Return the min value we could get at state
    '''
```

$$3 = \max(3, -2)$$

$$= \max(\min(3,15,6), \min(-2,15,3))$$

$$= \max(\min(\max(3,-2), \max(15,1), \max(-1,6)), \min(\max(-3,-2), \max(15,10), \max(1,3)))$$

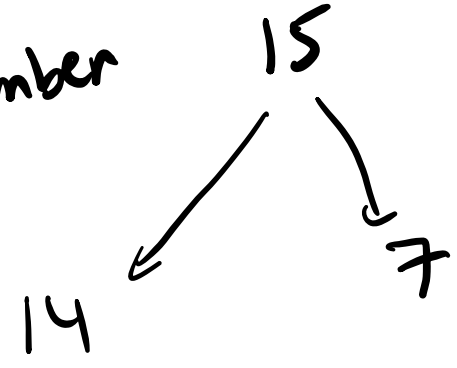
```
def max_value(state):
    """
    Return max value of a state
    """
    if terminal_state(state):
        return Utility(state)

    else:
        values_s' = []
        for a in Actions(state):
            s' = Result(state, a)
            values_s'.append(Min_value(s'))

        return max(values_s')

def Min_value(state):
    """
    Return min value of a state
    """
    if terminal_state(state):
        return Utility(state)

    else:
        values_s' = []
        for a in actions(state):
            s' = Result(state, a)
            values_s'.append(Max_value(s'))
```

```
        return min(values_s')

def action_chosen(state):
    '''
    return the optimal action at the
      given state
    '''
    # find action that maximizes the minimax value
    values_s' = []
    actions = []
    for a in actions(state):
        s' = Result(state, a)
        values_s'.append(Min-value(s'))
        actions.append(a)
    max_value = max(values_s')

    index_max = values_s'.index(max_value)

    return actions[index_max]
```

check if terminal state (if it is no action to take)

Game

starting number 15



15, 14, 7, 3, 2, 1, 0