# Lecture 15 (More Adversarial Searches)

## 1 Ordering - Iterative Deepening

1. We can increase the depth where we use the evaluation function at the "depth"
2. On increasing depth, use the previous evaluation to order the nodes which were evaluated earlier

## 2 Expectimax Search

Chance comes into picture when:

1. Action of adversary isn't known
2. Natural possibility of chance (rolling of dice etc)

### 2.1 Algorithm

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)

def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

Pruning cannot be performed

### 2.2 Depth Limited Expectimax

Using heuristic to evaluate the value at depth limit

# 3 General Minimax

1. Useful when there are multiple ($> 2$) players
2. Utility is represented as a tuple
3. Each player maximises its own component

# 4 Maximum Expected Utility

1. Choosing Actions that maximise expected utility
2. Agent can be in multiple states with certain probability
3. $U = \sum_i p_i U(S_i)$