Lecture 7 (UCS and Informed Search)

1 Uniform Cost Search (UCS)

```
function UCS(problem):
    node = a node with STATE=problem.INITIAL-STATE, PATH-COST=0
    frontier = pq ordered by PATH-COST, node as only element
    explored = empty set
    while True:
        if EMPTY(frontier) return failure
        node = POP(frontier)
        if problem.GOAL-TEST(node.STATE) return SOLUTION(node)
        explored.ADD(node)
        for action in problem.ACTIONS(node.STATE):
            child = CHILD(problem, node, action)
            if child.STATE not in explored or frontier:
                frontier.INSERT(child)
            else if child.STATE in frontier and PATH-COST larger:
                replace frontier node with child
```

1. Time complexity: $O(b^{C/\varepsilon})$ (C is solution cost, ε is minimum cost)
2. Space complexity: $O(b^{C/\varepsilon})$
3. Complete
4. Optimal

Common Practice (Beam Search): Bound the frontier to a maximum size, reduces space but loses out on completeness and optimality

2 Reversible States

1. They lead to repeated states
2. This leads to loopy or redundant paths
3. Can be solved using *graph search*, but memory (and time) inefficient
4. Approximate reduction: prevent adding parent, prevent cycles

3 Informed Search

In UCS, search might happen in the *opposite* or unlikely direction wrt goal, which is suboptimal

3.1 Best First Search

1. Find node from frontier which has best evaluation
2. Give priority using a function $f(n)$

3.1.1 Heuristic

A function $h(n)$ which formulates an approximate *guess* on the future cost