Exercise 1

1. What is the main difference of informed and uninformed search?

- All search strategies are distinguished by the order in which nodes are expanded.
- Uninformed means that the strategies have no additional information about states beyond that provided in the problem definition.
  All they can do is generate successors and distinguish a goal state from a non-goal state.
- Strategies that know whether one non-goal state is “more promising” than another are called informed search or heuristic search strategies.

2. A basic wooden railway set contains the pieces shown in Figure 3.32. The task is to connect these pieces into a railway that has **no overlapping tracks** and **no loose ends** where a train could run off onto the floor.
   a. Suppose that the pieces fit together exactly with no slack. Give a precise formulation of the task as a search problem.
   b. Identify a suitable uninformed search algorithm for this task and explain your choice.
   c. Explain why removing any one of the "fork" pieces makes the problem unsolvable.

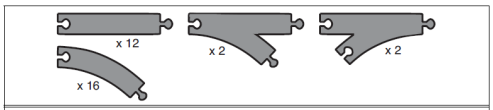


Figure 3.32 The track pieces in a wooden railway set; each is labeled with the number of copies in the set. Note that curved pieces and "fork" pieces ("switches" or "points") can be flipped over so they can curve in either direction. Each curve subtends 45 degrees.

a. Initial state: select a piece, e.g., a straight piece.
Successors: e.g., for any open peg select a remaining piece. Curved and Forks have also on orientation.
Goal test: all pieces are used, no open pegs and holes. No overlapping tracks
Cost: One per piece

b. All solutions are at the same level because all pieces must be used: 32. Therefore, DFS is the best solution

c. If a fork is removed there will be an open peg or hole.

3. Consider the problem of finding the shortest path between two points on a plane that has convex polygonal obstacles as shown in Figure 3.31. This is an idealization of the problem that a robot has to solve to navigate in a crowded environment.
   a. Suppose the state space consists of all positions (x, y) in the plane. How many states are there? How many path are there to the goal?
   b. Explain briefly, why the shortest path from one polygon vertex to any other in the scene must consist of straight-line segments joining some of the vertices of the polygons. Define a good state space now. How large is this state space?

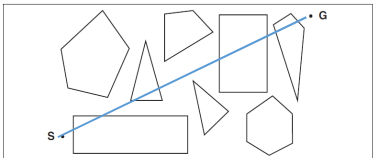


Figure 3.31 A scene with polygonal obstacles. *S* and *G* are the start and goal states.

a. When considering all possible points, then we have an infinite number of states, and path

b. The shortest path between S and G is a straight line (without obstacles).
The promising solution is a straight line connecting vertices and only minimal deviating from the straight line.
Therefore, we only consider vertices. The state space contains the 35 vertices of the figure

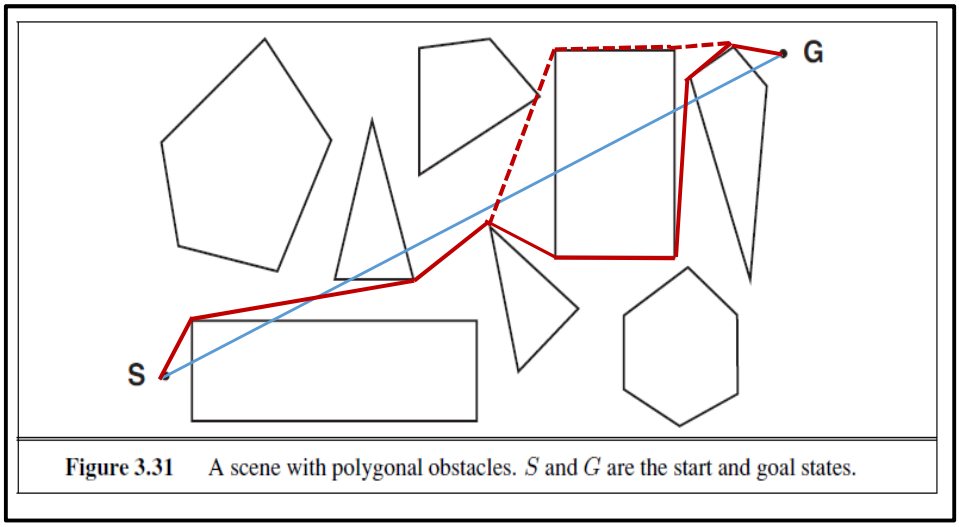


Figure 3.31 A scene with polygonal obstacles. S and G are the start and goal states.

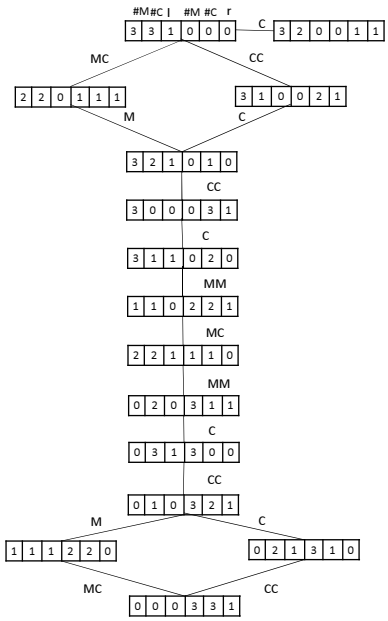
4. The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. A boat cannot drive alone. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint.
   a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution.
   b. Draw the state graph for the problem
   c. Implement and solve the problem using Bread-First-Search, Depth-Limited-Search and Iterative Deepening Search. Is it a good idea to check for repeated states?

a. One possible representation:

- **State**: A state is a six-tuple of integers listing the number of missionaries, cannibals, and boats each side.
  Start is all are on the left side.
- **Successor**: Successors of a state are all the states that move 1 or 2 people and 1 boat from one side to another.
  Not violating the restrictions: M ≥ C
- **Goal**: The goal is a state with 3 missionaries and 3 cannibals on the right side.
- **Cost**: The cost function is 1 per action.

b. See next slide

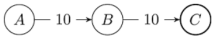
- $M \geq C$
- boat: min 1, max 2



4. The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. A boat cannot drive alone. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint.
   a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution.
   b. Draw the state graph for the problem
   c. Implement and solve the problem using Bread-First-Search, Depth-Limited-Search and Iterative Deepening Search. Is it a good idea to check for repeated states?

   c. Almost all moves are either illegal or revert to the previous state. A graph search make sense. Depth is 11.

5. What are admissible heuristics? How can we create them?
Consider the following search tree with start node A and goal node C



and the following heuristic functions h1 and h2:

$$h_1(A) = 20; \quad h_2(A) = 8$$
$$h_1(B) = 10; \quad h_2(B) = 11$$
$$h_1(C) = 0; \quad h_2(C) = 0$$

Can we compare the total cost of *real cost, h1, h2* to decide whether the heuristics are admissible?

- Admissible heuristics never overestimate the cost for reaching a goal state.
  h(n) ≤ h*(n), where h*(n) is the true cost to reach the goal state from n.
- Can be created by relaxing the rules of a problem.

- Sum real cost= 10+10=20
- Sum h1=20+10=30 > sum real cost
- Sum h2=8+11=19 < sum real cost

But: h(n) ≤ h*(n)

h1 is admissible. For each node h1 ≤ h*(n)
h2 is not admissible h2(B) =11 > h*(B)

9

6. Informally describe all functions that are required to solve the 8-Queens problem with Local-Beam-Search

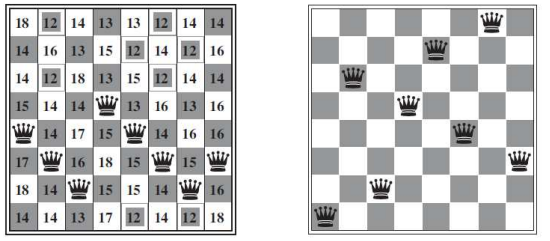


Figure 4.3 (a) An 8-queens state with heuristic cost estimate $h = 17$, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has $h = 1$ but every successor has a higher cost.

Most of the functions are related to checking conflicting queens and possible movements.
The rest is quite simple. You have to maintain a priority queue for k states

10

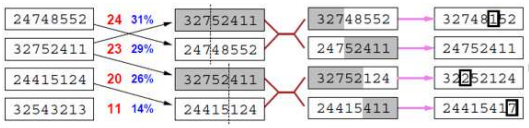
7. Explain the principals of the local search methods
   a. simulated annealing
   b. local beam search.
   c. genetic algorithm.

a. Simulated annealing uses a temperature vector that contains decreasing numbers together with the goodness of a state to allow next states that do not increase the goodness of the current state.

$next \leftarrow$ a randomly selected successor of $current$
$\Delta E \leftarrow$ VALUE[$next$] – VALUE[$current$]
if $\Delta E > 0$ **then** $current \leftarrow next$
else $current \leftarrow next$ only with probability $e^{\Delta\, E/T}$

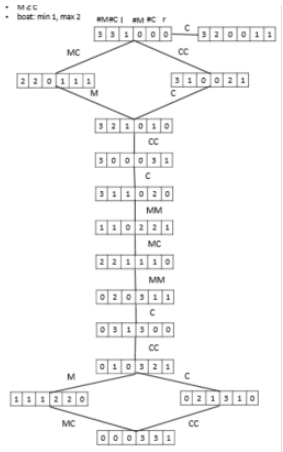
b. Local beam search puts only the k best next states in the frontier (recursively). It always expands the k most promising nodes.

c. Genetic search uses **k** states (population). It uses a *fitness function* to probabilistically generate next states. Afterwards, a *crossover function* combines two next states. In the last step it uses a *mutation function* to probabilistically change parts of the state.



11

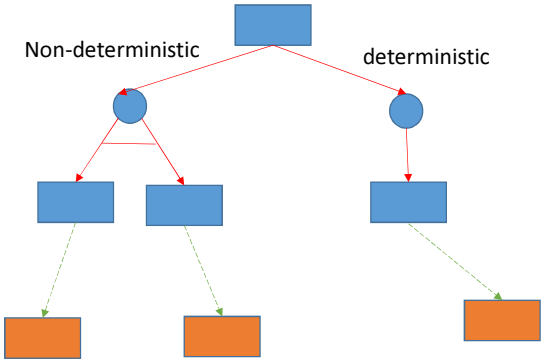
8. Does it make sense to use local beam search for the Cannibal problem?



The state space of the graph is small and there are maximal two successors.
Therefore, local beam search does not make sense

12

9. What are AND-OR graphs used for?

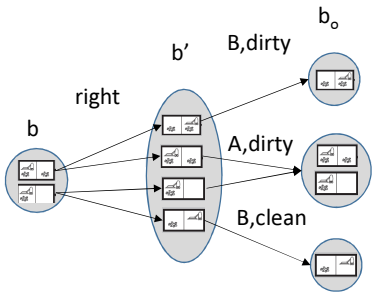


If we want to build a **contingency plan** for a problem with **uncertain actions** we introduce an AND branch representing a belief state with multiple states. For each of the states of the belief state a solution must be created.

13

10. How can partial observability and uncertain actions be handled in state search algorithms. Explain the functions that must be implemented

- A successor function that returns all possible next states for an action **a**: b'=Predict(b, a)
- An observation function that returns all possible observation **o** of b' Percepts(b')={o: o=PERCEPT(b')}



- Now we can define an Update of belief state for observation: b_o= UPDATE(b',o)= {s:o = PERCEPT(s) and s∈b'}

14

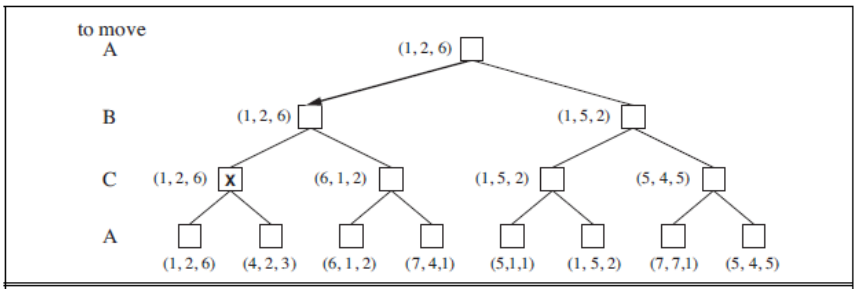
11. Give an informal description of the minimax algorithm of the lecture

    1. Generate game tree completely
    2. Determine utility of each terminal state
    3. Propagate the utility values upward in the tree by applying MIN and MAX operators on the nodes in the current level
    4. At the root node use minimax decision to select the move with the max (of the min) utility value

15

12. How can minimax be extended for multiplayer games?

Extend a leaf/node to a vector of values for each player. Each player chooses the vector with the highest value for him



16