

CS410: Artificial Intelligence 2021 Fall  
Homework 1: Search Algorithms  
Due date: 23:59:59 (GMT +08:00), October 10 2021

1. Consider the 8-queens problem. Your goal is to place 8 queens in a chess-board so that no two queens are in the same row, column, or diagonal. Recall that to formulate it as a search problem, we need to specify several components.

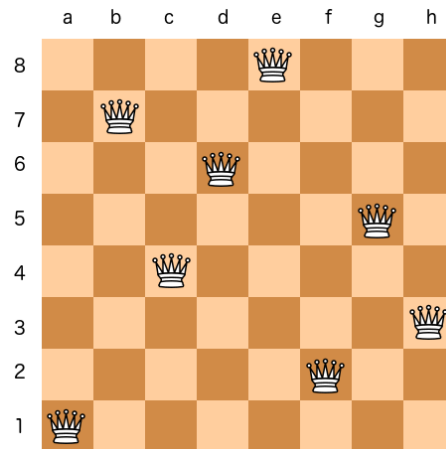


Figure 1: A feasible solution for the 8-queens problem.

- (a) Suppose we formulate the state space as the set of all arrangements of  $0, 1, 2, \dots, 8$  queens on the board. What would each component of the search problem be? How many states are there?

**Solution:**

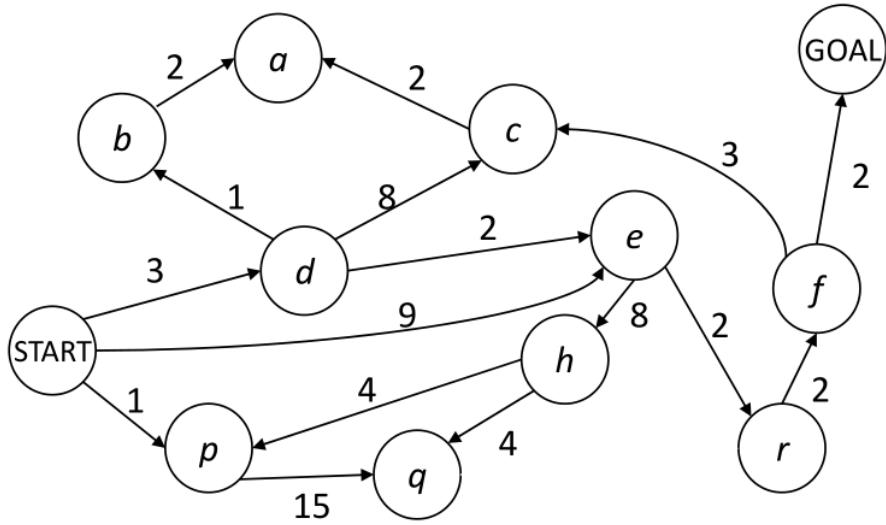
- State space: the set of all arrangements of  $0, 1, 2, \dots, 8$  queens on the board.
- Successor function: each of the successors is obtained by adding one queen in an empty square.
- Cost function: irrelevant.
- Start state: 0 queens on the board.
- Goal test: 8 queens are on the board, with no queens attacking each other.

- # states:  $\sum_{i=0}^8 \frac{64!}{(64-i)!}$ .
- (b) Give an alternative formulation which has a much smaller state space. List all the components of the search problem, and specify the number of states.

**Solution:** here is one possible formulation.

- State space: the set of all arrangements of 0, 1, 2, ..., 8 queens in the leftmost columns with no two queens attacking each other.
- Successor function: each of the successors is obtained by adding one queen in any square that is not attacked by any queen already in the board, in the leftmost empty column.
- Cost function: irrelevant.
- Start state: 0 queens on the board.
- Goal test: 8 queens are on the board.
- # states: 2057 (which can be counted by programming).

2. **Uniform cost graph search.** Consider the below state space graph. Perform UCS yourself and provide answers to the questions below regarding the nodes expanded during the search as well as the final path found by the algorithm. Remember that the search procedure should begin at node “START”, and the goal state is node “GOAL”. To break ties when adding nodes of equal cost to the fringe, follow the alphabetical order. Recall that UCS keeps track of the lowest cost,  $c(v)$ , to get from the start node to the node  $v$ .



- What is the order of nodes expanded?  
**Solution:**  $\text{START} \rightarrow p \rightarrow d \rightarrow b \rightarrow e \rightarrow a \rightarrow r \rightarrow f \rightarrow c \rightarrow \text{GOAL}$ .
- How many nodes are expanded?  
**Solution:** 10.
- What is the final path returned?  
**Solution:**  $\text{START} \rightarrow d \rightarrow e \rightarrow r \rightarrow f \rightarrow \text{GOAL}$ .
- What is the length of path?  
**Solution:** 5.
- What is the cost of path?  
**Solution:** 11.

3. **The graph-coloring problem.** Consider a graph with  $n$  nodes that has no multiple edges and loops. We want to color the nodes in this graph. We could only use 2 colors and we need to ensure that there is no edge linking any two nodes of the same color.
- (a) Please judge whether the following graphs could be colored as mentioned above. Write “Y” if you think one graph could be colored as mentioned above and write “N” otherwise.

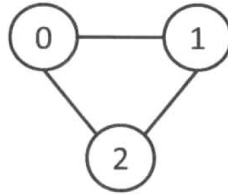


Figure 2: Graph 1.

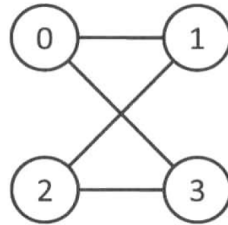


Figure 3: Graph 2.

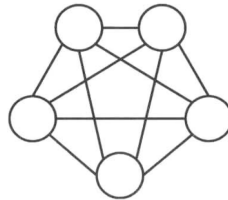


Figure 4: Graph 3.

**Solution:** Graph 1 and Graph 3 could not be colored as required but Graph 2 could be colored as required.

- (b) Please briefly describe your findings about the properties of the graphs that could be colored as mentioned above.

**Solution:** The graph which could be colored as required needs to be a bipartite graph. Other reasonable answers are acceptable (e.g., every circle in the graph should have an even length).

- (c) We further assume the graph is connected. Let 1 and -1 be the two color values denoting the two colors respectively. Please fill in the following pseudocode for judging whether a graph could be colored as stated in the problem. (Algorithm 2 returns “True” if a graph could be colored as stated in the problem and returns “False” otherwise)

---

**Algorithm 1** DFS

---

```

1: Input: node index  $v$ , color value  $c$ , color list  $color$ 
2:  $color[v] \leftarrow c$ 
3: for  $neighbor$  in  $get\_neighbor\_node(v)$  do
4:   if  $color[neighbor] == \text{-- } c \text{ --}$  then
5:     return False
6:   end if
7:   if  $color[neighbor] == 0$  and not  $\text{--DFS}(neighbor, -c, color)\text{--}$  then
8:     return False
9:   end if
10: end for
11: return True

```

---



---

**Algorithm 2** Color the Connected Graph

---

```

1: Initialize color list  $color \leftarrow (0)_{i=1}^N$ 
2: if  $\text{DFS}(1, 1, color)$  then
3:   return  $\text{-- True --}$ 
4: end if
5: return  $\text{-- False --}$ 

```

---

- (d) Please modify Algorithm2 to make Algorithm2 adapt to the graph which might be not connected. (Algorithm 3 returns “True” if a graph could be colored as stated in the problem and returns “False” otherwise)

---

**Algorithm 3** Color the Graph (might be not connected)

---

```

1: Initialize color list  $color \leftarrow (0)_{i=1}^N$ 
2: for  $node$  in  $get\_all\_node()$  do
3:   if  $color[node] == 0$  then
4:     if not  $\text{DFS}(node, 1, color)$  then
5:       return  $\text{-- False --}$ 
6:     end if
7:   end if
8: end for
9: return  $\text{-- True --}$ 

```

---

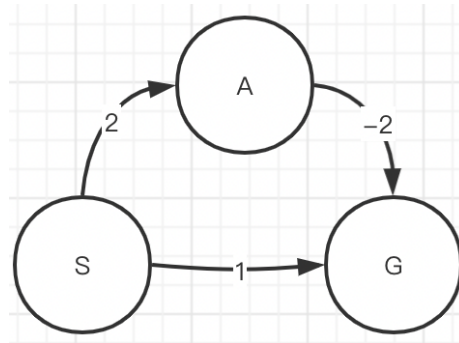


Figure 5: A graph with negative cost makes UCS fail to work.

#### 4. Explore the uniform-cost search in more depth.

- (a) What are the differences between the uniform-cost search and Dijkstra's algorithm? (You could click [here](#) for the reference of Dijkstra's algorithm.)

**Solution:**

- i. Dijkstra's algorithm finds the shortest path from the start state to every other states. While the uniform-cost search (UCS) searches for shortest paths in terms of cost from the start state to the goal state.
- ii. UCS has less memory consumption since the priority queue used in UCS is filled gradually while Dijkstra's algorithm adds all the states to the priority queue in the beginning with an infinite cost.

(Other reasonable answers are acceptable.)

- (b) If there are some arcs with negative costs in the graph, does the uniform-cost search still work? Give a concrete example to support your argument.

**Solution:** No. Consider the graph with negative cost in Figure 5. UCS will find the path  $S \rightarrow G$  with cost 1 but the path with minimum cost is  $S \rightarrow A \rightarrow G$  whose cost is 0.

- (c) Introduce one algorithm which you know and is able to find the least-cost path in the graph containing the arcs with negative costs.

**Solution:** Floyd-Warshall algorithm is one of the algorithms which could find the least-cost path in the graph containing the arcs with negative costs. The Floyd-Warshall algorithm compares all possible paths through the graph between each pair of vertices. It is able to do this with  $\Theta(|V|^3)$  comparisons in a graph and it does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal. This algorithm is based on the thought of dynamic programming. Other reasonable answers are acceptable.