Search
Problem Formulation and Uninformed Search

Alice Gao

Lecture 3

Readings: R & N 3.2, 3.3, 3.4.1, 3.4.3.

Based on work by K. Leyton-Brown, K. Larson, and P. van Beek

Outline

Learning goals

By the end of the lecture, you should be able to

- Formulate a real world problem as a search problem.
- Given a search problem, draw a portion of the search graph.
- Trace the execution of and implement uninformed search algorithms including Breadth-First Search and Depth-First Search.
- Given a scenario, explain why it is or it is not appropriate to use an uninformed algorithm.

Example: Sliding puzzles

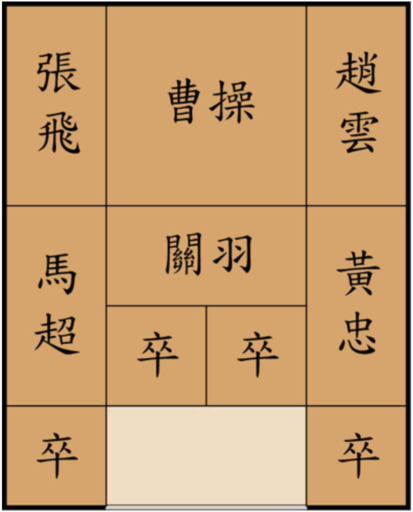
Initial State

5	3	
8	7	6
2	4	1

Goal State

1	2	3
4	5	6
7	8	

Example: Hua Rong Pass Puzzle



Example: Rubik's cube

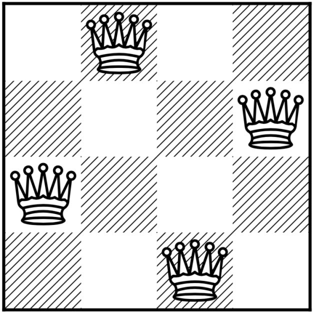
Example: River Crossing Puzzle

A parent and two children are trying to cross a river using a boat.

- The capacity of the boat is 100kg.
- The parent weighs 100kg.
- Each child weighs 50kg.

How can they get across the river?

Example: *N*-Queens Problem



The *n*-queens problem: Place *n* queens on an $n \times n$ board so that no pair of queens attacks each other.

Example: Propositional Satisfiability

Given a formula in propositional logic, determine if there is a way to assign truth values to the Boolean variables to make the formula true.

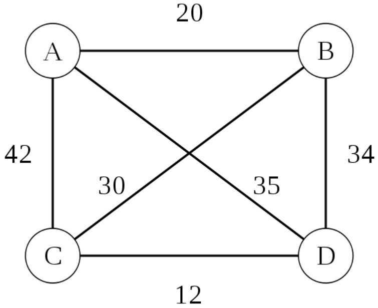
$$((((a \wedge b) \vee c) \wedge d) \vee (\neg e))$$

Applications:

- FCC spectrum auction
- Circuit design
- Planning in AI

Example: Traveling Salesperson Problem

What is the shortest path that starts at city A, visits each city only once, and returns to A?



Applications of TSP: https://bit.ly/2i9JdIV

Why search?

We would like to find a solution when we are

- Not given an algorithm to solve a problem
- Given a specification of what a solution looks like
- (Given costs associated with certain actions)

Idea: search for a solution (with the minimum cost)

A Search Problem

Definition (Search Problem)

A search problem is defined by

- A set of states
- A start state
- A goal state or goal test
  - a boolean function which tells us whether a given state is a goal state
- A successor function
  - a mapping/action which takes us from one state to other states
- A cost associated with each action

Example: 8-Puzzle

Initial State

5	3	
8	7	6
2	4	1

Goal State

1	2	3
4	5	6
7	8	

Formulating 8-Puzzle as a Search Problem

CQ: The successor function

CQ: Which of the following is a successor of $530,876,241$?

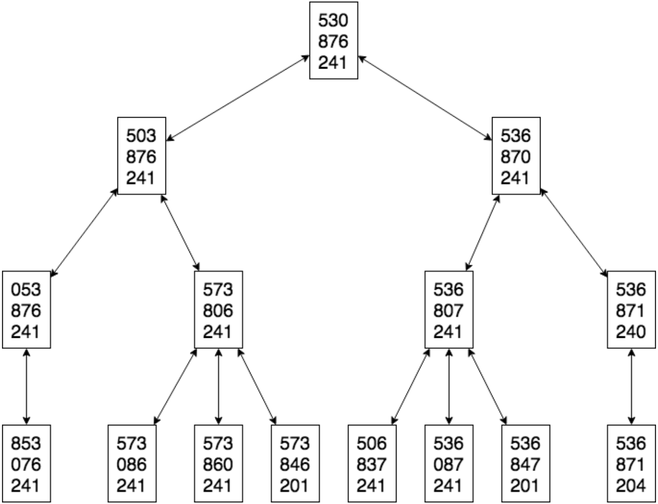
(A) $350,876,241$

(B) $536,870,241$

(C) $537,806,241$

(D) $538,076,241$

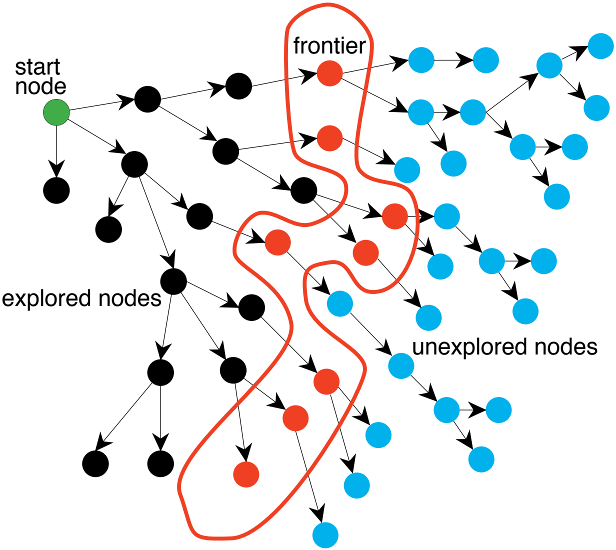
The Search Graph



Terminologies

- Search graph contains all the states and all the edges for the successor function.
- Search tree is constructed as we execute the algorithm.
- Frontier contains all the leaf nodes available for expansion.
- Expanding a node removes it from from the frontier.
- Generating a node adds the node to the frontier.

The Search Tree



CQ: What do we have to store in memory?

CQ: Assume that we have hard-coded all the information of 8-puzzle into our program. When we execute a search algorithm on the 8-puzzle, what do we have to store in memory?

(A) The frontier only

(B) The search graph and the frontier

(C) The search tree and the frontier

(D) The search graph, the search tree, and the frontier

Graph Search Algorithm

Algorithm 1 Search

```
let the frontier to be an empty list
add initial state to the frontier
while the frontier is not empty do
    remove curr_state from the frontier
    if curr_state is a goal state then
        return curr_state
    end if
    get all the successors of curr_state
    add all the successors to the frontier
end while
return no solution
```

Uninformed Search Algorithms

The search algorithms differ by the order in which we remove nodes from the frontier.

- **Breadth-first search** treats the frontier as a queue (FIFO).
- **Depth-first search** treats the frontier as a stack (LIFO).

The Execution of BFS

The Execution of DFS

Comparing BFS and DFS

Consider the scenarios below. Which of BFS and DFS would you choose? Why?

1. Memory is limited.
2. All solutions are deep in the tree.
3. The search graph contains cycles.
4. The branching factor is large.
5. We must find the shallowest goal node.
6. Some solutions are very shallow.

CQ: BFS v.s. DFS

CQ: Suppose that we have very limited memory to solve a problem. Which of BFS and DFS would you choose?

(A) I prefer BFS over DFS.

(B) I prefer DFS over BFS.

(C) Both are good choices.

(D) Neither is a good choice.

CQ: BFS v.s. DFS

CQ: Suppose that the search graph for a problem contains cycles. Which of BFS and DFS would you choose?

(A) I prefer BFS over DFS.

(B) I prefer DFS over BFS.

(C) Both are good choices.

(D) Neither is a good choice.

Dealing with Cycles in the Search Graph

- Prune explored states by storing them in a hash table.
- How would the properties of DFS change if we prune explored states?

Dealing with Cycles in the Search Graph

Algorithm 2 Search

```
let the frontier to be an empty list
add initial state to the frontier
while the frontier is not empty do
    remove curr_state from the frontier
    if curr_state is a goal state then
        return curr_state
    end if
    get all the successors of curr_state
    add all the successors to the frontier
end while
return no solution
```

Revisiting the learning goals

By the end of the lecture, you should be able to

- Formulate a real world problem as a search problem.
- Given a search problem, draw a portion of the search graph.
- Trace the execution of and implement uninformed search algorithms including Breadth-First Search and Depth-First Search.
- Given a scenario, explain why it is or it is not appropriate to use an uninformed algorithm.