Uninformed Search

Alice Gao

Lecture 2

Readings: RN 3.2, 3.3, 3.4.1, 3.4.3.

Outline

Learning goals

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

- Formulate a real world problem as a search problem.
- Trace the execution of and implement uninformed search algorithms (Breadth-first search, Depth-first search, Iterative-deepening search).
- Given an uninformed search algorithm, explain its space complexity, time complexity, and whether it has any guarantees on the quality of the solution found.
- Given a scenario, explain whether and why it is appropriate to use an uninformed algorithm.

Propositional Satisfiability

Propositional satisfiability: Given a formula, is there a way to assign true/false to the variables to make the formula true?

$$((((a \land b) \lor c) \land d) \lor (\neg e))$$

FCC spectrum auction: buy radio spectrums from TV broadcasters and sell them to the telecom companies

Check out this news article and this paper.

Hua Rong Dao Puzzle

Check out more initial configurations here.

8-puzzle

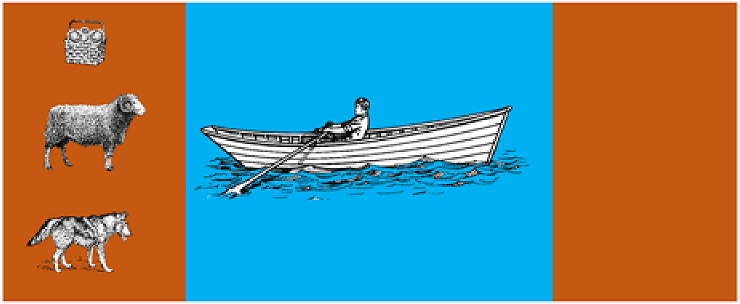
Initial State

5	3	
8	7	6
2	4	1

Goal State

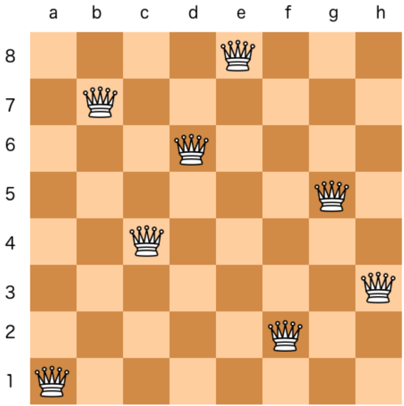
1	2	3
4	5	6
7	8	

A River Crossing Puzzle



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.



http://yue-guo.com/wp-content/uploads/2019/02/N_queen.png

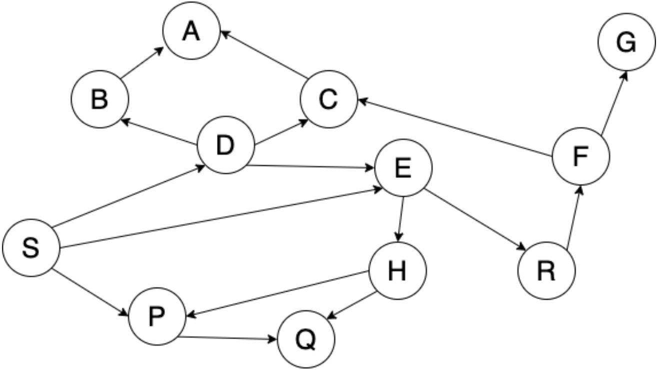
Why search?

We are facing a difficult problem.

- Given a description that helps us recognize a solution
- Not given an algorithm to solve the problem

We have to search for a solution!

Graph Searching



A Search Problem

Definition (Search Problem)

A **search problem** is defined by

- A set of **states**
- An **initial state**
- **Goal states** or **a goal test**
  - a boolean function which tells us whether a given state is a goal state
- A **successor (neighbour) function**
  - an action which takes us from one state to other states
- (Optionally) a **cost** associated with each action

A solution to this problem is a path from the start state to a goal state (optionally with the smallest total cost).

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

- State:
- Initial state:
- Goal states:
- Successor function:
- Cost function:

Formulating 8-Puzzle as a Search Problem

- State: $x_{00}x_{01}x_{02}, x_{10}x_{11}x_{12}, x_{20}x_{21}x_{22}$
  x_{ij} is the number in row i and column j. $i, j \in \{0, 1, 2\}$.
  $x_{ij} \in \{0, \ldots, 8\}$. $x_{ij} = 0$ denotes the empty square.

Formulating 8-Puzzle as a Search Problem

- State: $x_{00}x_{01}x_{02}, x_{10}x_{11}x_{12}, x_{20}x_{21}x_{22}$
  x_{ij} is the number in row i and column j. $i, j \in \{0, 1, 2\}$.
  $x_{ij} \in \{0, \ldots, 8\}$. $x_{ij} = 0$ denotes the empty square.

- Initial state: $530, 876, 241$.

Formulating 8-Puzzle as a Search Problem

- State: $x_{00}x_{01}x_{02}, x_{10}x_{11}x_{12}, x_{20}x_{21}x_{22}$
  x_{ij} is the number in row i and column j. $i, j \in \{0, 1, 2\}$.
  $x_{ij} \in \{0, \ldots, 8\}$. $x_{ij} = 0$ denotes the empty square.

- Initial state: $530, 876, 241$.

- Goal states: $123, 456, 780$.

Formulating 8-Puzzle as a Search Problem

- State: $x_{00}x_{01}x_{02}, x_{10}x_{11}x_{12}, x_{20}x_{21}x_{22}$
  x_{ij} is the number in row i and column j. $i, j \in \{0, 1, 2\}$.
  $x_{ij} \in \{0, \ldots, 8\}$. $x_{ij} = 0$ denotes the empty square.

- Initial state: $530, 876, 241$.

- Goal states: $123, 456, 780$.

- Successor function: Consider the empty square as a tile. State B is a successor of state A if and only if we can convert A to B by moving the empty tile up, down, left, or right by one step.

Formulating 8-Puzzle as a Search Problem

- State: $x_{00}x_{01}x_{02}, x_{10}x_{11}x_{12}, x_{20}x_{21}x_{22}$
  x_{ij} is the number in row i and column j. $i, j \in \{0, 1, 2\}$.
  $x_{ij} \in \{0, \ldots, 8\}$. $x_{ij} = 0$ denotes the empty square.
- Initial state: $530, 876, 241$.
- Goal states: $123, 456, 780$.
- Successor function: Consider the empty square as a tile. State B is a successor of state A if and only if we can convert A to B by moving the empty tile up, down, left, or right by one step.
- Cost function: Each move has a cost of 1.

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) $538, 076, 241$

Choosing among multiple formulations

Choosing among multiple formulations

- The state definition determines the nodes.
  The successor function determines the directed edges.

Choosing among multiple formulations

- The state definition determines the nodes.
  The successor function determines the directed edges.

- Ideally, we want to minimize the number of nodes and edges in the graph.

Choosing among multiple formulations

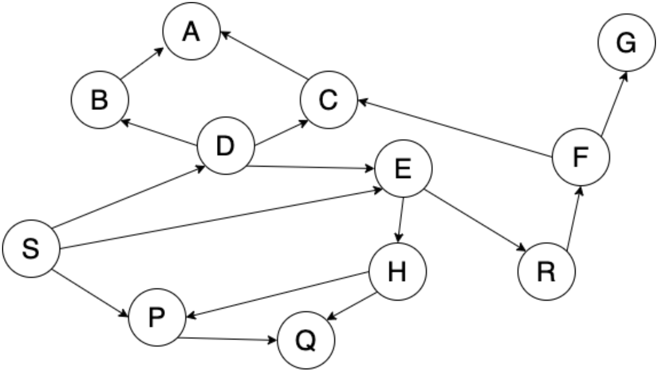
- The state definition determines the nodes.
  The successor function determines the directed edges.

- Ideally, we want to minimize the number of nodes and edges in the graph.

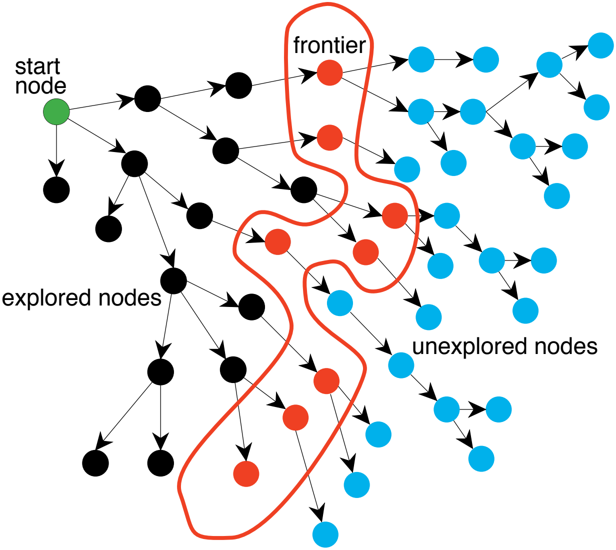
- Choosing a state definition may make it easier or harder to implement the successor function.

  An alternative state definition for the 8-puzzle:
  A state is defined by 8 coordinates.
  (x_i, y_i) is the coordinates for tile i where $1 \leq i \leq 8$.

The Search Graph



The Search Tree



Generic Search Algorithm

Algorithm 1 A Generic Search Algorithm

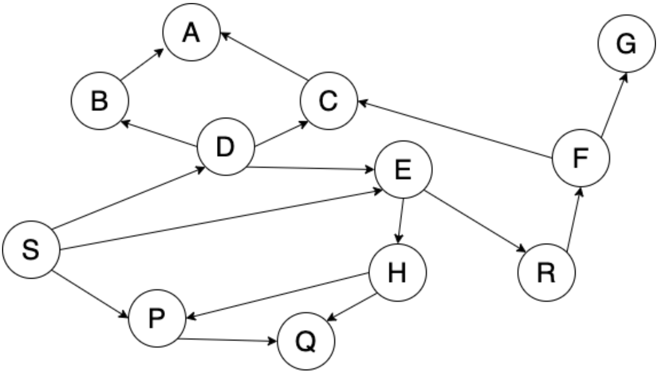
procedure SEARCH(Graph, Start node s, Goal test $goal(n)$)
    frontier := $\{\langle s \rangle\}$
    while frontier is not empty **do**
        select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from frontier
        if goal(n_k) **then**
            return $\langle n_0, \ldots, n_k \rangle$
        for every neighbour n of n_k **do**
            add $\langle n_0, \ldots, n_k, n \rangle$ to frontier
    return no solution

Depth-First Search

- Treats the frontier as a stack (LIFO).
- Expands the last/most recent node added to the frontier.

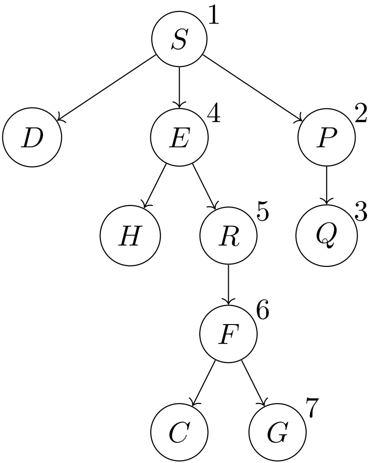
Trace DFS on a Search Graph

- Trace DFS on the search graph below.
- Add nodes to the frontier in alphabetical order.



Trace DFS on a Search Graph

The final search tree:



Properties of DFS

Properties

- Space Complexity
- Time Complexity
- Completeness
- Optimality

Useful Quantities

- b is the branching factor.
- m is the maximum depth of the search tree.
- d is the depth of the shallowest goal node.

Properties of DFS - Space Complexity

Space Complexity

Properties of DFS - Space Complexity

Space Complexity

- $O(bm)$

  b is the branching factor.
  m is the max depth of the search tree.

- Linear in m

- Remembers m nodes on current path and at most b siblings for each node.

Properties of DFS - Time Complexity

Time Complexity

Properties of DFS - Time Complexity

Time Complexity

- $O(b^m)$
- Exponential in m.
- Visit the entire search tree in the worst case.

Properties of DFS - Completeness

Is DFS guaranteed to find a solution if a solution exists?

Properties of DFS - Completeness

Is DFS guaranteed to find a solution if a solution exists?

- No.
- Will get stuck in an infinite path.
- An infinite path may or may not be a cycle.

Properties of DFS - Optimality

Is DFS guaranteed to return an optimal solution if it terminates?

Properties of DFS - Optimality

Is DFS guaranteed to return an optimal solution if it terminates?

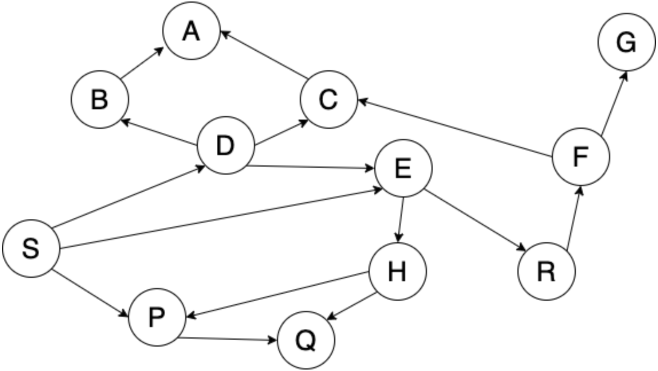
- No.
- Pays no attention to the costs and makes no guarantee on the solution's quality.

Breadth-First Search

- Treats the frontier as a queue (FIFO).
- Expands the first/oldest node added to the frontier.

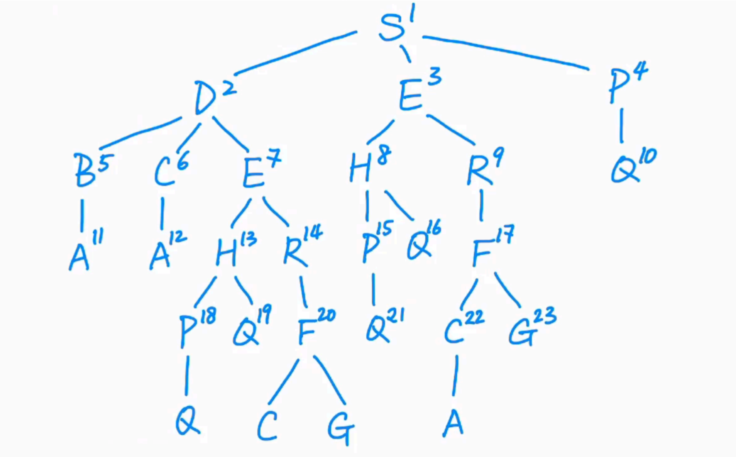
Trace BFS on a Search Graph

- Trace BFS on the search graph below.
- Add nodes to the frontier in alphabetical order.



Trace BFS on a Search Graph

- Trace BFS on the search graph below.
- Add nodes to the frontier in alphabetical order.



Properties of BFS - Space Complexity

Space Complexity

Properties of BFS - Space Complexity

Space Complexity

- $O(b^d)$

  b is the branching factor.
  d is the depth of the shallowest goal node.

- Exponential in d.

- Must visit the top d levels.
  Size of frontier is dominated by the size of level d.

Properties of BFS - Time Complexity

Time Complexity

Properties of BFS - Time Complexity

Time Complexity

- $O(b^d)$
- Exponential in d.
- Visit the entire search tree in the worst case.

Properties of BFS - Completeness

Is BFS guaranteed to find a solution if a solution exists?

Properties of BFS - Completeness

Is BFS guaranteed to find a solution if a solution exists?

- Yes.
- Explores the tree level by level until it finds a goal.

Properties of BFS - Optimality

Is BFS guaranteed to return an optimal solution if it terminates?

Properties of BFS - Optimality

Is BFS guaranteed to return an optimal solution if it terminates?

- No.

- Guaranteed to find the shallowest goal node.

Combining The Best of BFS and DFS

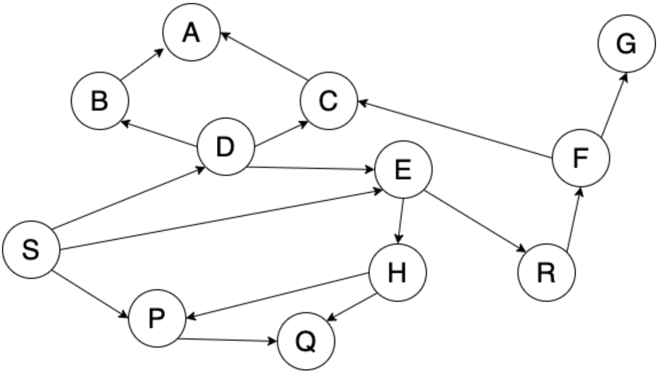
Can we create a search algorithm
that combines the best of BFS and DFS?

BFS	**DFS**
$O(b^d)$ exponential space	$O(bm)$ linear space
Guaranteed to find a solution if one exists	May get stuck on infinite paths

Iterative-Deepening Search:
For every depth limit,
perform depth-first search until the depth limit is reached.

Trace IDS on a Search Graph

- Trace IDS on the search graph below.
- Add nodes to the frontier in alphabetical order.



Trace IDS on a Search Graph

- Trace IDS on the search graph below.
- Add nodes to the frontier in alphabetical order.

Properties of IDS - Space Complexity

Space Complexity

Properties of IDS - Space Complexity

Space Complexity

- $O(bd)$

  b is the branching factor.
  d is the depth of the shallowest goal node.

- Linear in d.
  Similar to DFS.

- Executes DFS for each depth limit.
  Guaranteed to terminate at depth d.

Properties of IDS - Time Complexity

Time Complexity

Properties of IDS - Time Complexity

Time Complexity

- $O(b^d)$
- Exponential in d.
  Same as BFS.
- Visits all the nodes on the top d levels in the worst case.

Properties of IDS - Completeness

Is IDS guaranteed to find a solution if a solution exists?

Properties of IDS - Completeness

Is IDS guaranteed to find a solution if a solution exists?

- Yes.
  Same as BFS.

- Explores the tree level by level until it finds a goal.

Properties of IDS - Optimality

Is IDS guaranteed to return an optimal solution if it terminates?

Properties of IDS - Optimality

Is IDS guaranteed to return an optimal solution if it terminates?

- No.

- Guaranteed to find the shallowest goal node.
  Same as BFS.

A Summary of IDS Properties

- Space Complexity:
  $O(bd)$, linear in d. Similar to DFS.
- Time Complexity:
  $O(b^d)$, exponential in d. Same as BFS.
- Completeness:
  Yes. Same as BFS.
- Optimality:
  No, but guaranteed to find the shallowest goal node.
  Same as BFS.

Revisiting the learning goals

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

- Formulate a real world problem as a search problem.
- Trace the execution of and implement uninformed search algorithms (Breadth-first search, Depth-first search, Iterative-deepening search).
- Given an uninformed search algorithm, explain its space complexity, time complexity, and whether it has any guarantees on the quality of the solution found.
- Given a scenario, explain whether and why it is appropriate to use an uninformed algorithm.