Definitions of AI and Uninformed Search

Alice Gao

Lecture 2

Readings: RN 1.1, 3.2, 3.3, 3.4.1, 3.4.3.

Outline

Learning goals

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

- Describe the four definitions of AI. Explain why we will pursue one over the other three.
- 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, and Lowest-cost-first 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.

Outline

What is Artificial Intelligence?

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

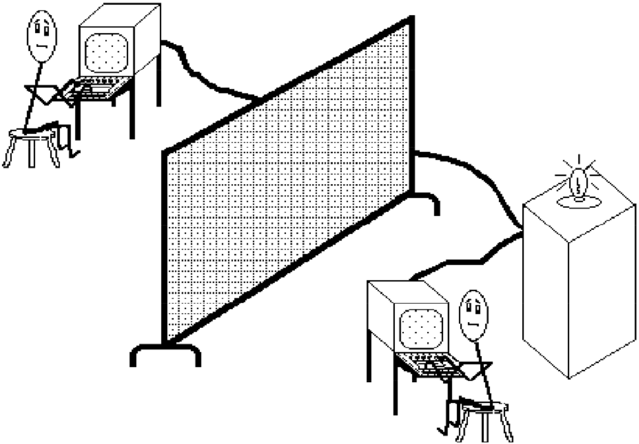
Thinking Humanly

The Cognitive Modeling Approach

- Why humans?
- How do humans think?
  - Introspection
  - Psychological experiments
  - Brain imaging (MRI)
- Cognitive science

Acting Humanly

The Turing Test Approach



The Turing Test

- An operational definition
- Is the Turing Test useful?

Rationality

- Rationality: an **abstract "ideal" of intelligence**, rather than "whatever humans do"
- A system is rational if it does the "right thing," given what it knows.

Thinking Rationally

The Laws of Thought Approach

- Greek philosopher Aristotle defined syllogisms.
- The logicist tradition
- Two obstacles for using this approach in practice

Acting Rationally

The Rational Agent Approach:

- Agent means todo.
- A rational agent acts to achieve the best (expected) outcome.
- What behaviour is rational?

CQ: Which definition of intelligence would you adopt?

CQ: If you were an Artificial Intelligence researcher, which of the following definitions of intelligence would you adopt?

(A) Systems that think like humans
(B) Systems that act like humans
(C) Systems that think rationally
(D) Systems that act rationally

Caring about Behaviour rather than Thoughts

Why do we care about behaviour instead of thought processes and reasoning?

Measure Success against Rationality rather than Humans

Why do we measure success against rationality instead of against humans?

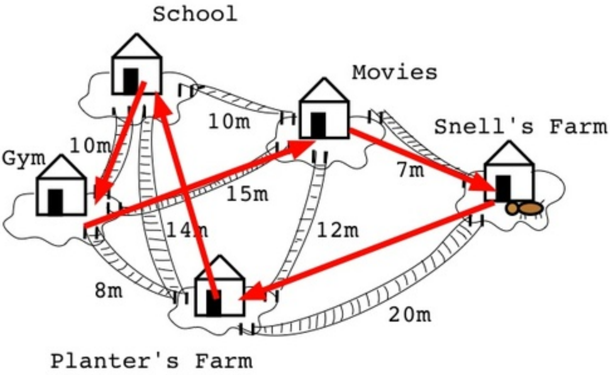
What is Artificial Intelligence?

Cognitive Modeling Systems that think like humans	**Laws of Thought** Systems that think rationally
Turing Test Systems that act like humans	**Rational Agent** Systems that act rationally

Example: Transportation and Logistics

Applications of TSP: School bus routes, Service calls, Parcel pickups, DNA sequencing. More at https://bit.ly/2i9JdIV

Traveling Salesperson Problem



Example: Propositional Satisfiability

Given a propositional formula, is there a way to assign truth values to the variables to make the formula true?

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

One application: FCC spectrum auction

Check out papers by Neil Newman and Kevin Leyton-Brown.
For example, check this one out.

Example: 8-puzzle

Initial State

5	3	
8	7	6
2	4	1

Goal State

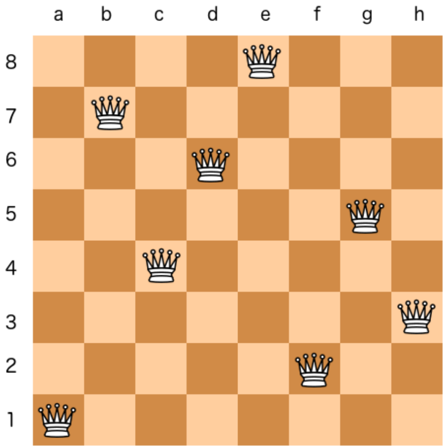
1	2	3
4	5	6
7	8	

Example: Hua Rong Pass Puzzle

Example: River Crossing Puzzle

https://www.lounge.se/MissionariesCannibalsProblem

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.

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

Why search?

We are facing a difficult problem.

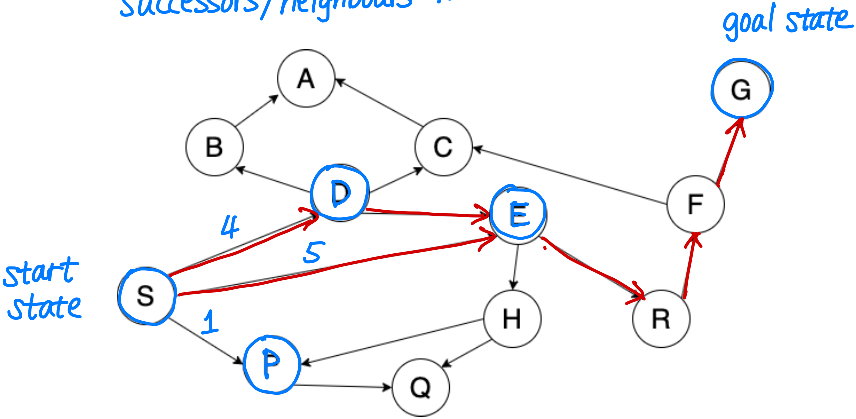
- Given a description that helps us recognize a solution
- Not given an algorithm to solve the problem

We have to search for a solution!

Graph Searching

states ← nodes

successors/neighbours function ← directed arcs



Graph Search Formally

Definition (Search Problem)

A **search problem** is defined by

- A set of **states**
- A **start 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

530, 876, 241

Goal State

1	2	3
4	5	6
7	8	

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.

$i, j \in \{0, 1, 2\}$ $\quad x_{ij} \in \{0, \ldots, 8\}$

$x_{ij} = 0$ if and only if the square is empty.

initial state: 530, 876, 241.

goal state: 123, 456, 789.

successor function: to generate a successor, move the empty square left, right, up, or down, whenever possible.

cost function: every 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) $538, 076, 241$

530
876
241

503
876
241

536
870
241

Why does the formulation matter?

- The state definition can affect the amount of information needed to encode each state, and how difficult it is to implement the successor function.

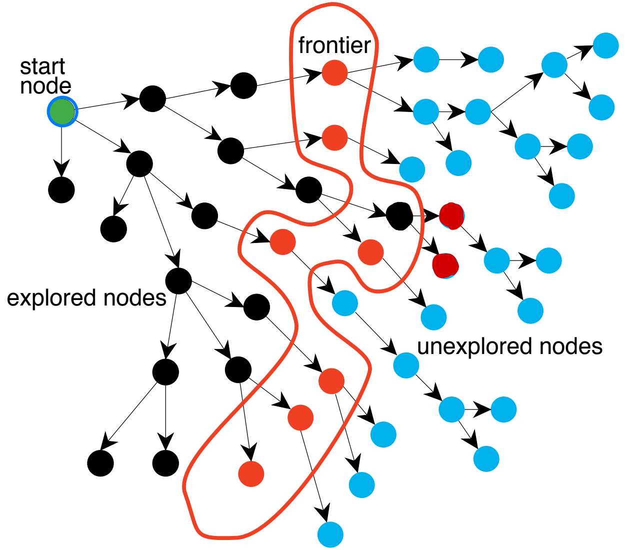
  A state is defined by 8 coordinates.

  (x_i, y_i) are the coordinates for the tile i where $i \in \{1, \ldots, 8\}$.

- The successor function can affect the size and structure of the search graph.

  Missionaries and cannibals problem.

Generating the Search Tree



Searching for a Solution

- Construct the search tree as we explore paths incrementally from the start node.
- Maintain a frontier of paths from the start node.
- Frontier contains all the leaf nodes available for expansion.
- Expanding a node: removes it from the frontier, generating all of its neighbours and adding the neighbours to the frontier.

Graph Search Algorithm

Algorithm 1 A Generic Search Algorithm

procedure SEARCH(Graph, Start node s, Goal test $goal(n)$)

    frontier := $\{\langle s \rangle\}$;   determines the search strategy.

    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

Uninformed Search Algorithms

- Depth-first search
- Breadth-first search
- Iterative-deepening search
- Lowest-cost-first search

Learning Goals

Definitions of Artificial Intelligence

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms
- Depth-first search
- Breadth-first search
- Iterative-deepening search
- Lowest-cost-first search

Revisiting the Learning Goals

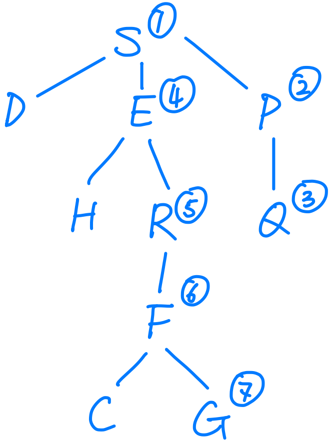
Depth-First Search

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

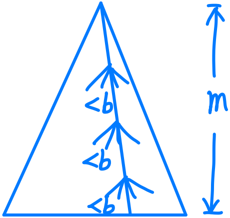
Trace DFS on the search graph on slide 25

frontier: ~~S~~ D ~~E~~ ~~P~~ ~~Q~~ H ~~R~~ ~~F~~ C ~~G~~

Add nodes to the frontier in alphabetical order.



Properties of DFS



- Space complexity? (size of frontier)

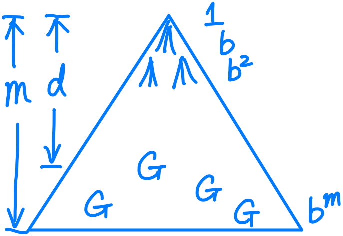
  $O(bm)$ linear in m.

- Time complexity? worst case : visit all the states.

  $O(b^m)$ exponential in m.

- Guaranteed to find a solution if one exists? (completeness)

  No. If the search tree has an infinite path, DFS will not terminate.



Useful quantities:

b: branching factor; **m**: maximum depth;
d: depth of the shallowest goal node

Search w/ Cycle Pruning

Algorithm 2 Search w/ Cycle Pruning

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**
            if $n \notin \langle n_0, \ldots, n_k \rangle$ **then**
                add $\langle n_0, \ldots, n_k, n \rangle$ to frontier;
    return no solution

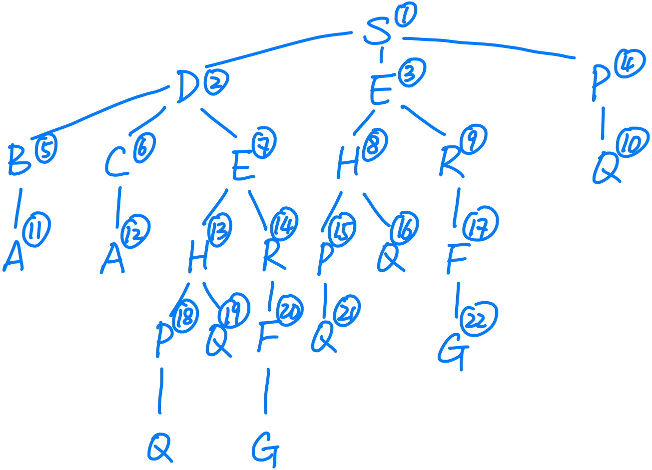
Breadth-First Search

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

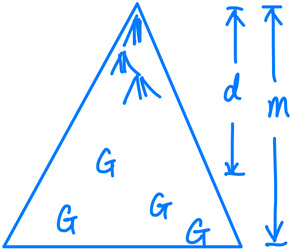
Trace BFS on the search graph on slide 25

frontier : ~~S~~ ~~B~~ ~~C~~ ~~E~~ ~~H~~ ~~R~~ ~~Q~~ ~~A~~ ~~A~~ ~~H~~ ~~R~~ ~~R~~ ~~Q~~ ~~F~~ ~~R~~ ~~Q~~ ~~F~~ ~~Q~~ ~~G~~ Q G

Add nodes to the frontier in alphabetical order.



Properties of BFS



- Space complexity? (size of the frontier)

  $O(b^d)$ exponential in d.

- Time complexity?

  $O(b^m)$ exponential in m.

- Guaranteed to find a solution if one exists?

  Yes.

- Guaranteed to find the shallowest goal node?

  Yes.

Useful quantities:

b: branching factor; **m**: maximum depth;

d: depth of the shallowest goal node

Search w/ Multi-Path Pruning

Algorithm 3 Search w/ Multi-Path Pruning

procedure SEARCH(Graph, Start node s, Goal test *goal*(n))
    frontier := $\{\langle s \rangle\}$;
    explored := $\{\}$;
    while frontier is not empty **do**
        select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from frontier;
        if $n_k \notin$ **explored then**
            add n_k to explored
            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

BFS and DFS — the best of both worlds

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

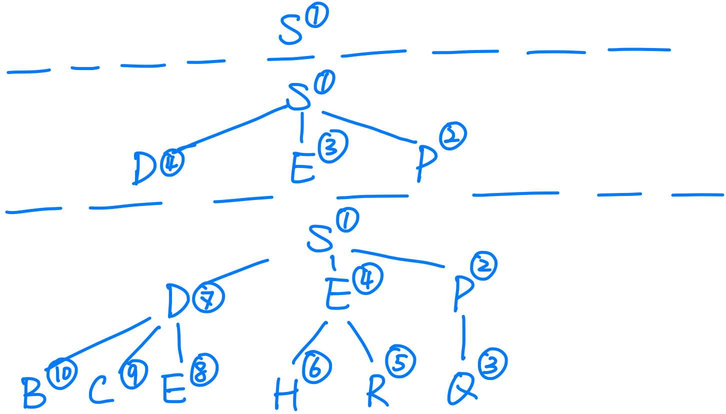
BFS	**DFS**
$O(b^d)$ exponential space	$O(bm)$ linear space
Guaranteed to find a solution if one exists	May get stuck on infinite paths

Iterative-deepening search

frontier: ~~S~~ ~~S~~ ~~D~~ ~~E~~ ~~P~~ ~~S~~ ~~D~~ ~~E~~ ~~P~~ ~~Q~~ ~~H~~ ~~R~~ ~~B~~ ~~C~~ ~~E~~

depth limit: ~~1~~ ~~2~~ 3

For every depth limit,
perform depth-first search until the depth limit is reached.



Properties of IDS

BFS: $b^d + b^{d-1} + b^{d-2} + \cdots + b + 1 = b^d\left(\frac{b}{b-1}\right) - \frac{1}{b-1}$

IDS: $b^d + 2b^{d-1} + 3b^{d-2} + \cdots + db + (d+1) \leq b^d\left(\frac{b}{b-1}\right)^2$

- Space complexity:

  $O(bd)$ linear in d (similar to DFS)

- Time complexity: similar to BFS but a bit worse

  $O\left(b^d\left(\frac{b}{b-1}\right)^2\right)$ due to repeated computation.

- Guaranteed to find a solution if one exists?

  Yes. (same as BFS)

- Guaranteed to find the shallowest goal node?

  Yes. (same as BFS)

Revisiting the learning goals

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

- Describe the four definitions of AI. Explain why we will pursue one over the other three.
- 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.