Uninformed Search

CS 4804 Fall 2020
Virginia Tech
Today’s Topics

• Search Problems
• Uninformed Search Methods
  – Depth-First Search
  – Breadth-First Search
  – Uniform-Cost Search
• Homework logistics
Search Problem
Suggested day plan

San Diego Zoo
4.7 ★★★★★ (37,510)
2 hours
Zoo features giraffes, koalas, apes & more plus a safari park for tours & a conservation society.

10 minutes by foot

Museum of Photographic Arts
4.6 ★★★★★ (405)
1 hour
More than 7,000 photos from 650 artists are showcased in a replica of a grand, ornate 1915 building.

Less than 5 minutes by foot

San Diego Model Railroad Museum
4.6 ★★★★★ (298)
1 hour
Kid-friendly museum featuring many working scale models of California rail lines & landscapes.

Less than 5 minutes by foot

Botanical Building
4.7 ★★★★★ (3,000)
15 minutes
Dome-shaped building from the 1915 Panama-California Exposition, & a garden with more than 2,000 plants.

Less than 5 minutes by foot

The San Diego Museum of Art
4.6 ★★★★★ (1,269)
1.6 hours
Venerable art museum with acclaimed works, plus a courtyard bar/bistro with sculpture garden views.

5 minutes by foot

House of Pacific Relations - Hall of Nations
4.6 ★★★★★ (87)
15 minutes
Problem-solving agent

- Agents that plan ahead
- Use atomic representation
  - States of the world are considered as whole
  - No internal structure visible to the problem-solving algorithms
- Uninformed algorithms
  - Agent is unable to estimate how far it is from the goal
Search Problems and solutions

- A state space
- Initial state (Start state)
- Goal state(s)
- Actions(s)
- Transition model: RESULT(s, a)
- Action cost function: Action-Cost(s, a, s’)

A solution is a sequence of actions (path) which transforms the initial state to a goal state.

An optimal solution has the lowest path cost among all solutions.
Figure 3.1 A simplified road map of part of Romania, with road distances in miles.
Example: 8-puzzle

Start State

Goal State
State Space Graph

- A state space graph is a mathematical representation of a search problem
  - Nodes are world configurations (abstracted)
  - Arcs represent successors (action results)
  - The goal test is a set of goal nodes (maybe only one)
- In a search graph, each state occurs only once.
Search Tree

- Each node is corresponding to a state in the state space
- The start state is the root node
- Each edge is corresponding to an action
- Children nodes correspond to successors
- Each node encodes an entire path and corresponds to plans to achieve that state
Search Algorithms

- Input: Search problem
- Output: solution or an indication of failure
- Superimpose a search tree over the state space graph
Best-first Search

How do we decide which node from the frontier to expand next?

Priority queue

f(n): evaluation function

Select node with minimum f(n) value

A node that represents a path to a goal

function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
node ← NODE(STATE=problem.INITIAL)
frontier ← a priority queue ordered by f, with node as an element
reached ← a lookup table, with one entry with key problem.INITIAL and value node
while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
        s ← child.STATE
        if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
            reached[s] ← child
            add child to frontier
        return failure

function EXPAND(problem, node) yields nodes
s ← node.STATE
for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
Algorithm’s Performance

- Completeness: Guaranteed to find a solution if one exists or correctly report failure
- Cost optimality: Find the lowest path cost solution
- Time complexity: Number of operations to find the solution
- Space complexity: Amount of memory needed to find the solution
Uninformed Search Strategies

No clue about how close a state is to the goal(s)

- Breadth-First Search
- Depth-First Search
- Uniform-Cost Search
Breadth-First Search

- All actions have the same cost
- FIFO queue
- Completeness: If a solution exists, Yes
- Cost optimality: only if costs are all the same
- Time complexity: $O(b^d)$
- Space complexity: $O(b^d)$
Depth-First Search
Depth-First Search

- Finds the “leftmost” solution in the search tree
- LIFO queue
- Completeness: No
- Cost optimality: Doesn’t care about costs, No
- Time complexity: $O(b^m)$
- Space complexity: $O(bm)$
Depth-limited Search

- DFS with a depth limit $L$
- Depth limit can be chosen based on knowledge of the problem
- Completeness: No
- Cost optimality: No
- Time complexity: $O(b^L)$
- Space complexity: $O(bL)$
Iterative Deepening Search

• Depth-limited search with depth $L=1$, then $L=2$, …
• Until goal state found
• Completeness: Yes
• Cost optimality: Only if costs are all the same
• Time complexity: $O(b^d)$
• Space complexity: $O(bd)$
Bidirectional Search

- Search forward from the initial state and search backwards from the goal state(s)
- Run bidirectional best-first search
- Completeness: Yes
- Cost optimality: Yes
- Time complexity: $O(b^{d/2})$
- Space complexity: $O(b^{d/2})$
Uniform-cost Search

- Dijkstra’s algorithm
- Actions have different costs
- Expand nodes in order of cost from the initial state
- Completeness: Yes
- Cost optimality: Yes
- Time complexity: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
- Space complexity: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
- $\varepsilon$ a lower bound on the cost of each action, with $\varepsilon > 0$
Uniform-Cost Search (UCS)

- S->R: 80
- S->R->P: 177
- S->F: 99
- S->F->B: 310
- S->R->P->B: 278
## Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Uniform-Cost</th>
<th>Depth-First</th>
<th>Depth-Limited</th>
<th>Iterative Deepening</th>
<th>Bidirectional (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>Yes(^1)</td>
<td>Yes(^1,2)</td>
<td>No</td>
<td>No</td>
<td>Yes(^1)</td>
<td>Yes(^1,4)</td>
</tr>
<tr>
<td>Optimal cost?</td>
<td>Yes(^3)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes(^3)</td>
<td>Yes(^3,4)</td>
</tr>
<tr>
<td>Time</td>
<td>(O(b^d))</td>
<td>(O(b^{1+\lfloor C^*/\epsilon \rfloor}))</td>
<td>(O(b^m))</td>
<td>(O(b^\ell))</td>
<td>(O(b^d))</td>
<td>(O(b^{d/2}))</td>
</tr>
<tr>
<td>Space</td>
<td>(O(b^d))</td>
<td>(O(b^{1+\lfloor C^*/\epsilon \rfloor}))</td>
<td>(O(bm))</td>
<td>(O(b\ell))</td>
<td>(O(bd))</td>
<td>(O(b^{d/2}))</td>
</tr>
</tbody>
</table>
Recap: UCS
Recap: DFS / BFS / UCS
Project 0 - Programming

- To become familiar with Python
- To become familiar with Unix basic
- To become familiar with autograder
- To become familiar with a version control repository for your program. GitHub or Bitbucket or etc.
- (Optional) to become familiar with Docker or AWS
Homework 0 - Written

• Also in Canvas
• Check your knowledge (No grade)
Python Tutorial

• https://github.com/CS4804/tutorial
Reading and Next Class

- Uninformed Search, AIMA 3.1-3.4
- Next: Informed Search, AIMA 3.5-3.6