Introduction to Artificial Intelligence: Uninformed Search

Dr. Tapas Kumar Mishra

Assistant Professor
Department of Computer Science and Engineering
SRM University
Amaravati-522 240, Andhra Pradesh, India
Email: tapaskumar.m [at] srmap [dot] edu [dot] in

Uninformed/Blind Search - Depth-Limited Search

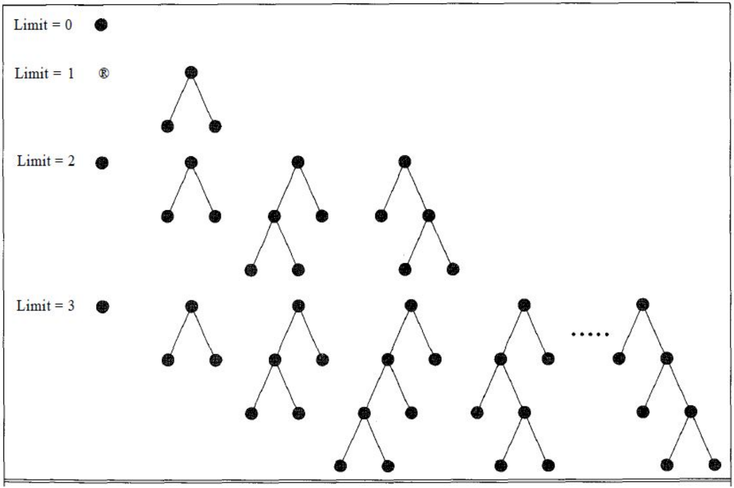
- Depth-limited search **avoids** the **pitfalls** of **depth-first search** by imposing a **cutoff** on the **maximum depth** of a path.
- **Example:** On the map of Romania, there are **20 cities**, so we know that if there is a **solution**, then it must be of **length 19** at the **longest**.
- If you are in **city A** and have **travelled** a path of **less than 19 steps**, then generate a new state in **city B** with a path length that is **one greater**.
- We are guaranteed to find the solution if it exists.

Properties

1. Complete: Yes, if $l > d$
   - If we choose a depth limit that is too small, then depth-limited search is not even complete.
2. Optimal: No
3. Time: $O(b^l)$ // l is the depth limit.
4. Space: $O(bl)$

- The hard part about **depth-limited search** is picking a **good limit**.
- We would discover that **any city** can be reached from **any other city** in at most **9 steps** instead of **19**.
- Diameter of the state space = 9
- **Iterative deepening search** is a strategy that sidesteps the issue of choosing the **best depth limit** by trying all possible **depth limits**: first **depth 0**, then **depth 1**, then **depth 2**, and so on.
- The order of **expansion of states** is similar to **breadth-first**, except that some states are **expanded multiple times**.
- **Iterative deepening search** may seem **wasteful**, because so many states are expanded multiple times.

Uninformed/Blind Search - Iterative Deepening Search



- The number of expansions in a **depth-limited search** to **depth** d with **branching factor** b is

$$1 + b + b^2 + \cdots + b^{d-2} + b^{d-1} + b^d$$

- $b = 10$ and $d = 5$

$$1 + 10 + 10^2 + 10^3 + 10^4 + 10^5 = 111111$$

- The total number of expansions in an **iterative deepening search** is

$$(d+1)1 + (d)b + (d-1)b^2 + \cdots + 3b^{d-2} + 2b^{d-1} + 1b^d$$

- $b = 10$ and $d = 5$

$$(5+1)1 + (5)10 + (5-1)10^2 + (5-2)10^3 + (5-3)10^4 + (5-4)10^5$$

$$= 6 + 50 + 400 + 3000 + 20000 + 100000 = 123456 \text{ (11\% more nodes)}$$

Uninformed/Blind Search - Iterative Deepening Search

Properties

1. Complete: Yes
2. Optimal: Yes
3. Time: $O(b^d)$
4. Space: $O(bd)$

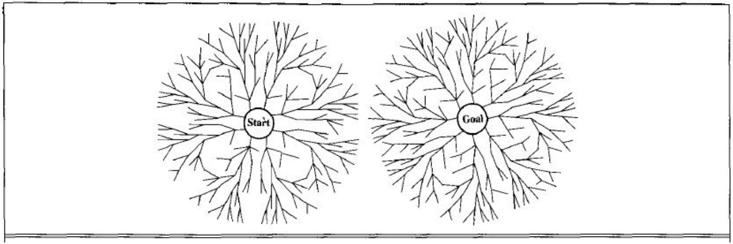
- Iterative deepening is the preferred search method when there is a large search space and the depth of the solution is not known.

Uninformed/Blind Search - Bidirectional Search

- The idea behind bidirectional search is to simultaneously search both **forward** from the **initial** state j and **backward** from the **goal** and stop when the **two searches** meet in the **middle**.
- The solution will be found in $O(2b^{d/2}) = O(b^{d/2})$ steps because the **forward** and **backward** searches each have to go only **half way**.

Issues

1. What does it mean to search backwards from the goal?
2. When all operators are **reversible**, the predecessor and successor sets are **identical**. calculating **predecessors** can be **very difficult**.
3. What can be done if there are many possible goal states?



Properties

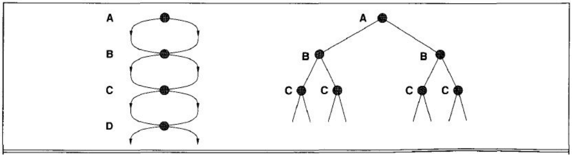
1. Complete: Yes
2. Optimal: Yes
3. Time: $O(b^{d/2})$
4. Space: $O(b^{d/2})$

Avoiding Repeated States

- For many problems, repeated states are unavoidable.
  - Missionaries and Cannibals problem
- The space contains only $m + 1$ states, where m is the maximum depth.

Three ways to deal with repeated states

- Do not return to the state you just came from.
- Do not create paths with cycles in them.
- Do not generate any state that was ever generated before.



Best-first search

Idea: use an evaluation function for each node
– estimate of "desirability"

⇒ Expand most desirable unexpanded node

Implementation:
fringe is a queue sorted in decreasing order of desirability

Special cases:
greedy search
A* search

Greedy search

Evaluation function $h(n)$ (heuristic)
$\quad = $ estimate of cost from n to the closest goal

E.g., $h_{\mathrm{SLD}}(n) = $ straight-line distance from n to Bucharest

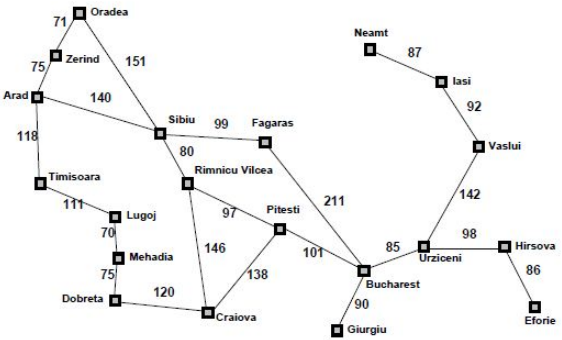
Greedy search expands the node that appears to be closest to goal

We look at two basic approaches.

- To expand the node closest to the goal
- To expand the node on the least-cost solution path

Informed Search - Best First Search

Romania with step costs in km



Straight−line distance to Bucharest

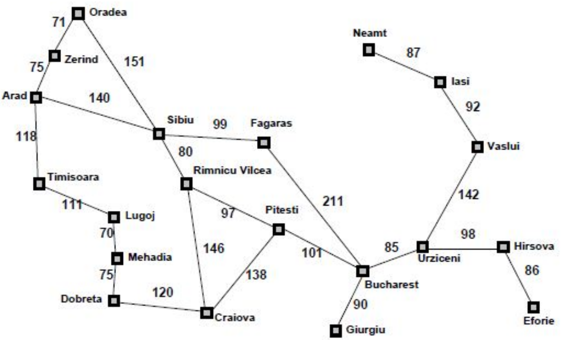
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy search example

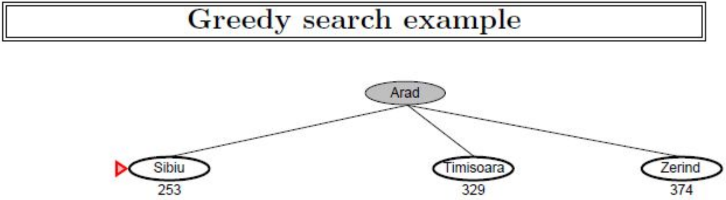
- $h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.
- $h_{SLD}(n)$ = straight-line distance between n and the goal location.

Informed Search - Best First Search

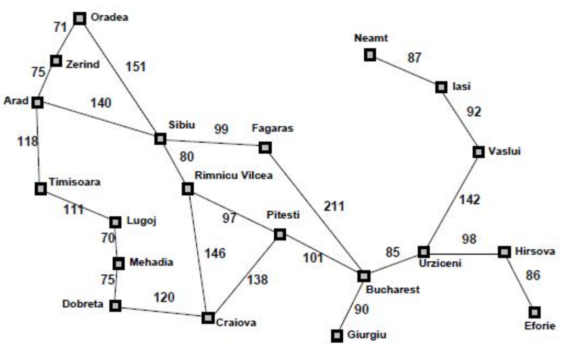
Romania with step costs in km



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374


Greedy search example
Arad
Sibiu
253
Timisoara
329
Zerind
374

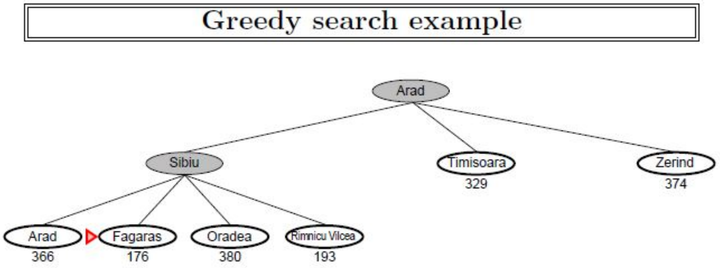
Romania with step costs in km



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

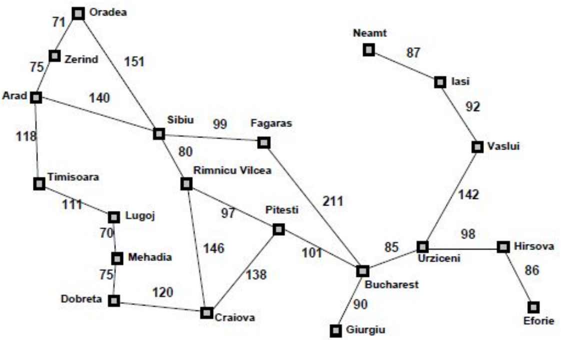
Informed Search - Best First Search



- **Fagaras:** 178

Informed Search - Best First Search

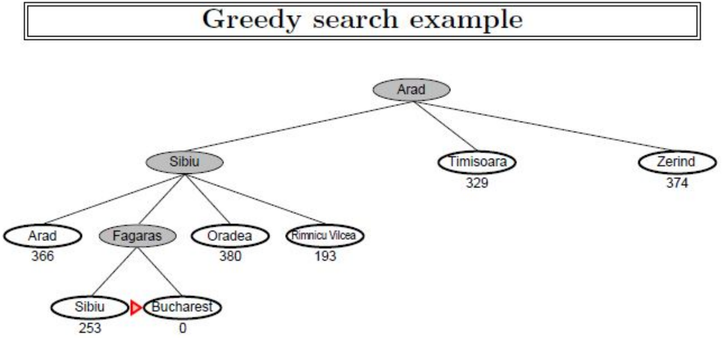
Romania with step costs in km



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Informed Search - Best First Search



- It suffers from the same defects as **depth-first search**-it is **not optimal**, and it is **incomplete** because it can start down an **infinite path** and never return to try other possibilities.
- The worst-case time complexity for greedy search is $O(b^m)$, where m is the maximum depth of the search space.
- Because greedy search retains all nodes in memory, its space complexity is the same as its time complexity.

Thank You!