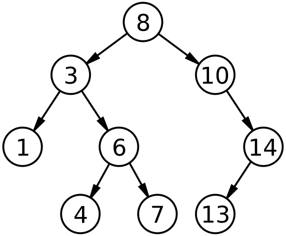
Week 20: Search II

Martha Lewis
(based on slides from Raul Santos Rodriguez)

Summary

Objective: find the minimum cost path from s_{start} to an s satisfying $Goal(s) = $TRUE.



Tree search: Backtracking, BFS, DFS, DFS-ID

Outline

Search or how to find a sequence of actions that achieves a goal when no single action will do. We will cover

- Uniform Cost Search
- Informed vs Uninformed search
- Greedy search

Breadth First vs Uniform Cost Search

When all step costs are equal, breadth-first search is optimal because it always expands the shallowest unexpanded node...

...what if the costs are not equal?

Instead of expanding the shallowest node, uniform-cost search expands the node n with the lowest path cost $g(n)$.
UCS maintains a queue of nodes, ordered by lowest path cost.
State: summary of past actions sufficient to choose future actions optimally

path cost

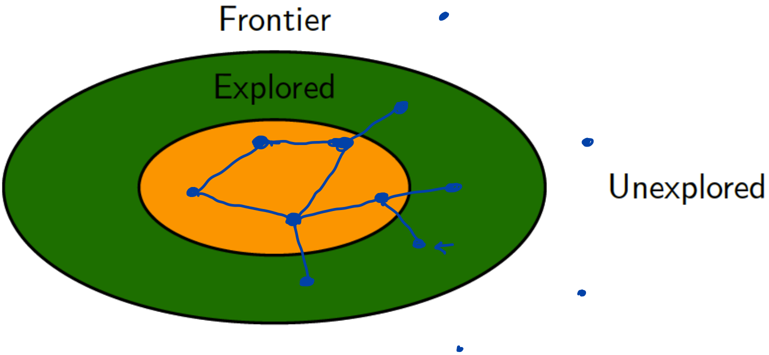
Breadth First vs Uniform Cost Search

Two main differences:

1. The goal test is applied to a node when it is selected for expansion rather than when it is first generated: the first goal node that is generated may be on a suboptimal path.
2. A test is added in case a better path is found to a node currently on the frontier.

Assumption: all action costs are non-negative

Algorithm: intuition



- **Explored**: states we've found the optimal path to
- **Frontier**: states we've seen, still figuring out how to get there cheaply
- **Unexplored**: states we haven't seen

[Liang 2014]

UCS: travel from city 1 to city n

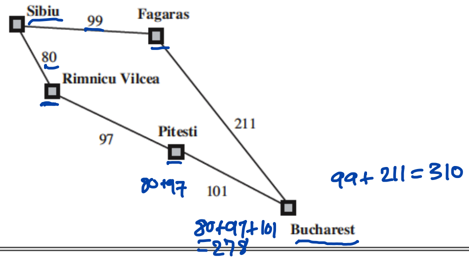
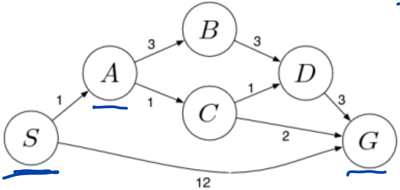


Figure 3.15 Part of the Romania state space, selected to illustrate uniform-cost search.

[Russell and Norvig]

UCS: example



Initialise S:0

1 SA:1 , SG:12

2 SAC:2, SAB:4 , SG:12

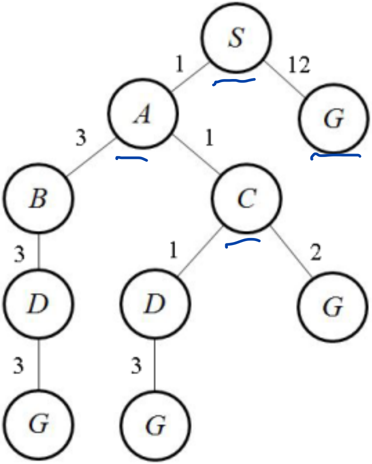
3 SACD:3, SAB:4, SACG:4, SG:12

4 SAB:4, SACG:4, SACD:6, SG:12

5 SACG:4, SACD:6, SABD:7, SG:12

6 Return path SACG

UCS: example



UCS algorithm

Algorithm

Add s_{start} to *frontier* (priority queue)

Repeat until *frontier* is empty:

  Remove s with smallest priority p from frontier

  If Goal(s): **return** solution

  Add s to *explored*

  For each action $a \in Actions(s)$:

    Get successor $s' \leftarrow Succ(s, a)$
    If s' already in *explored* : **continue**
    Else update *frontier* with s' and priority $p + Cost(s, a)$

- UCS is optimal
- UCS does not care about the number of steps a path has, but only about their total cost: infinite loops!
- Completeness is guaranteed if $c_{ij} \geq \epsilon \; \forall x, y$ for some small constant $\epsilon > 0$
- When all step costs are the same, UCS is similar to BFS, except that BFS stops as soon as it generates a goal, whereas UCS examines all the nodes at the goal's depth to see if one has a lower cost.

Summary

- Tree search: memory efficient, suitable for huge state spaces (to the extent anything works)
- State: summary of past actions sufficient to choose future actions optimally
- Graph search: uniform cost search constructs optimal paths

Question

Suppose we want to travel from city 1 to city n (going only forward) and back to city 1 (only going backward). It costs $c_{ij} \geq 0$ to go from i to j. Which of the following algorithms will find the minimum cost path (select all that apply)?

Question

- Depth-first search
- Breadth-first search
- Uniform cost search

Uniformed vs Informed

Uninformed search strategies

- Uninformed search strategies use no information about the likely *direction* of the goal node(s)
- Uninformed search methods: Breadth-first, depth-first, depth-limited, uniform-cost, depth-first iterative deepening,...

Informed search strategies

- Also known as *heuristic search*, informed search strategies use information about the domain to (try to) head in the direction of the goal node(s)
- Informed search methods: Hill climbing, best-first, greedy search, beam search, A, A*,...

Goal make UCS faster

Problem UCS orders states by cost from s_{start} to s

Idea take into account cost from s to s_{goal}

Best-first node n is selected for expansion based on an evaluation function $f(n)$

Heuristics

Definition

A heuristic $h(n)$ is the estimated cost of the cheapest path from the state at node n to a goal state.

- Unlike $g(n)$, it depends only on the state at that node
- Encode additional knowledge of the problem
- Arbitrary and nonnegative
- Constraint: if n is a goal node, then $h(n) = 0$

Greedy best-first search

A best-first search that uses h to select the next node to expand: greedy search.

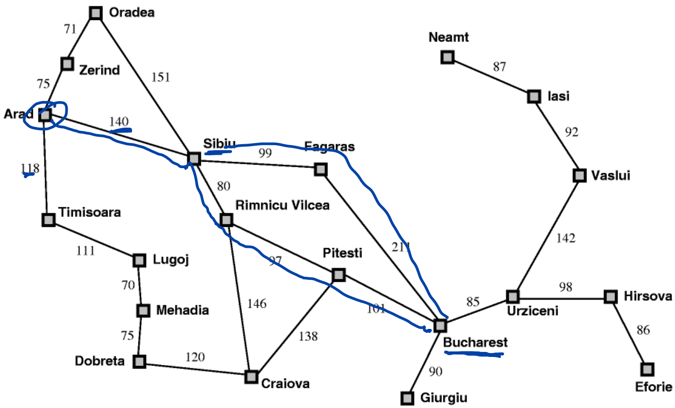
Greedy search

$$\underline{f(n)} = \underline{h(n)}$$

Is greedy search

- complete? NO (be careful with loops)
- optimal? NO

Greedy best-first search: example



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 best-first search: example