Tutorial - Week 4

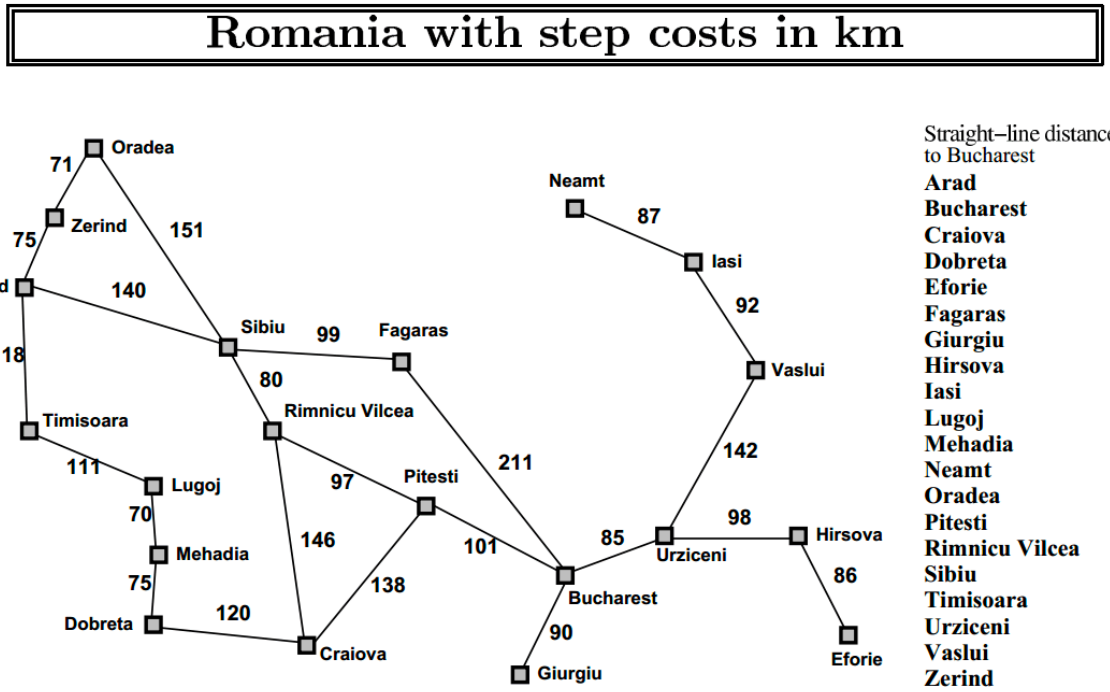
Informed Search

Activity 1

For the route from Arad to Bucharest, what order are nodes in the state space expanded for each of the following algorithms when searching for the shortest path between Arad and Bucharest? Where there is a choice of nodes, take the first one by alphabetical ordering. Make sure you understand the key properties of the different algorithms, as listed below.

(v) Greedy:
Arad (366), Sibiu (253), Fagaras (178), Bucharest (0)

(vi) A*:
Arad (366), Sibiu (393), Rimnicu Vilcea (413), Pitesti (415), Fagaras (417), Bucharest (418) (unexpanded states on the frontier are Timisoara (447), Zerind (449), Craiova (526), Oradea (671)) – for example, f(Rimnicu Vilcea) = g(Rimnicu Vilcea) + h(Rimnicu Vilcea) = 140 + 80 + 193 = 413 (remember to use the total path cost from Arad to Rimnicu Vilcea here)



Activity 2

a) Breadth First Search is a special case of Uniform Cost Search.
Uniform Cost Search reduces to Breadth First Search when all edges have the same cost.

b) Breadth First Search, Depth First Search and Uniform Cost Search are special cases of best-first search.
Best-first search reduces to Breadth-First Search when f(n)=number of edges from start node to n, to UCS when f(n)=g(n); it can be reduced to DFS by, for example, setting f(n)=-(number of nodes from start state to n) (thus forcing deep nodes on the current branch to be searched before shallow nodes on other branches). Another way to produce DFS is to set f(n)=-g(n) (but this might produce a particularly bad choice of nodes within the DFS framework - for example, try tracing the order in which nodes are expanded when traveling from Urziceni to Craiova).

c) Uniform Cost Search is a special case of A*Search.
*A*Search reduces to UCS when the heuristic function is zero everywhere, i.e. h(n)=0 for all n; this heuristic is clearly admissible since it always (grossly!) underestimates the distance remaining to reach the goal.*

Activity 3

The heuristic path algorithm is a best-first search in which the objective function is:

$$f(n) = (2-w)g(n) + wh(n)$$

What kind of search does this perform when w=0? when w=1? when w=2?

*This algorithm reduces to Uniform Cost Search when w=0, to A*Search when w=1 and to Greedy Search when w=2.*

For what values of w is this algorithm complete? For what values of w is it optimal, assuming h() is admissible?

*It is guaranteed to be optimal when $0 \leq w \leq 1$, because it is equivalent to A*Search using the heuristic $h'(n) = [w/(2-w)]h(n) \leq h(n)$*
When $w > 1$ it is not guaranteed to be optimal (however, it might work very well in practice, for some problems).

Activity 4

Suppose a manufacturing robot needs to schedule a set of activities: casting, drilling, bolting, to take place in an afternoon (from 1pm to 5pm). Each activity takes an hour. The process requires drilling before bolting. Casting and drilling cannot happen at the same time, but bolting must happen exactly two hours after casting ends.

Find a legal assignment using the basic Backtracking search algorithm if there is one.

We model this problem as a CSP.
- *Variables: c, d , and b for the starting time of casting, drilling, and bolting respectively*
- *Domain: {1, . . . , 4}.*
- *Constraints: $d < b$, $c \neq d$, $b = c + 3$*

Backtracking:
[(b=1), (b=2), (b=3), (b=4)]
[(b=2), (b=3), (b=4)] (b=1, c=1), (b=1, c=2), (b=1, c=3), (b=1, c=4) - violate b=c+3
[(b=3), (b=4)] (b=2, c=1), (b=2, c=2), (b=2, c=3), (b=2, c=4) - violate b=c+3
[(b=4)] (b=3, c=1), (b=3, c=2), (b=3, c=3), (b=3, c=4) - violate b=c+3
[(b=4, c=1)] (b=4, c=2), (b=4, c=3), (b=4, c=4) - violate b=c+3
[(b=4, c=1, d=2), (b=4, c=1, d=3)] (b=4, c=1, d=1) - violates c ≠d, (b=4, c=1, d=4) - violates d<b

There are two valid assignments.