Best First Search (Heuristic Search)

- a greedy search
- We evaluate a heuristic function.

Movement should be such that we move in a forward direction.

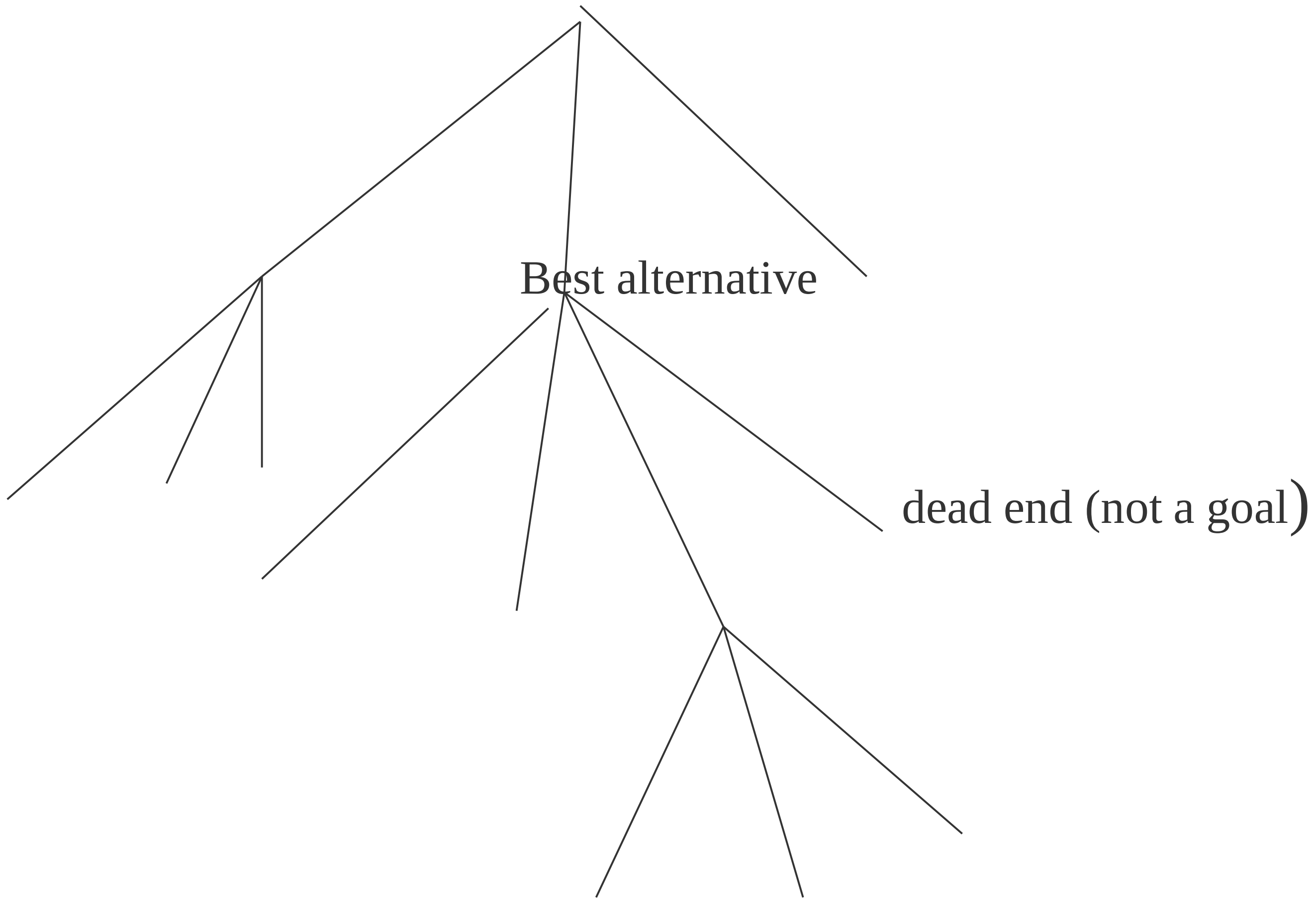
(1) Begin with a set of rules which are consistent and try to arrive at a solution.

(2) User did not explicitly specify the sequence of use of rules. Sequence of rules is decided by the KB (i.e. by the system).

In some way, the solution space should be explored.

AI systems attempt to get solution best in some sense (acceptable solution), they do not attempt to get optimality. Solutions obtained by AI systems may not be optimal.

The best does not mean optimal. **Best is in heuristic sense**.



This avoids searching completely. This is a Greedy algorithm.

The word **heuristic** comes from **Greek word heuriskein** meaning **to discover**.

Heuristic function is a function that gives a judgement from given stage to goal stage, e.g., nearest neighbor.

At each step of the best first search process, we select the most promising of the nodes we have generated so far.We then expand the chosen node by using the rules to generate its successors.

If one of them is a solution, we then quit. If not, all those new nodes are added to the nodes generated so far. Again the most promising node is selected and the process continues.

Usually, what happens is that a bit of depth-first occurs as the most promising branch is explored. But eventually, if a solution is not found, that branch will start to look less promising than one of the top level branches that had been ignored. At that point, the new more promising previously ignored branch will be explored. But the old branch is not forgotten. Its last node remains in the set of generated but unexpanded nodes.The search can return to it whenever all the others get bad enough that it is again the most promising path.

Best-first search for Graphs

Each node will contain

- a description of the problem state it represents
- an indication of how promising it is
- a parent link that points back to the best node from which it came
- a list of nodes that were generated from it.

The **parent link** will make it possible to recover the path to the goal once the goal is found.

The **list of successors** will make it possible. If a better path is found to an already existing node, to propagate the improvement down to its successors.

This type of graph is called an OR graph, since each of its branches represent an alternative problem-solving path.

To implement such a graph search procedure, we need to use two list of nodes.

OPEN- list of nodes(generated and heuristic function calculated) in form of a **priority queue** with the highest priority of those nodes with the most promising value of the heuristic function.

CLOSED - nodes that have already been examined We need to keep these nodes in memory if we want to search a graph rather than a tree, since whenever a new node is generated, we need to check whether it has been generated before.

$$f' = g + h'$$

where

f' = approximation. to f (heuristic function)

g = cost of getting from the initial state to the current node

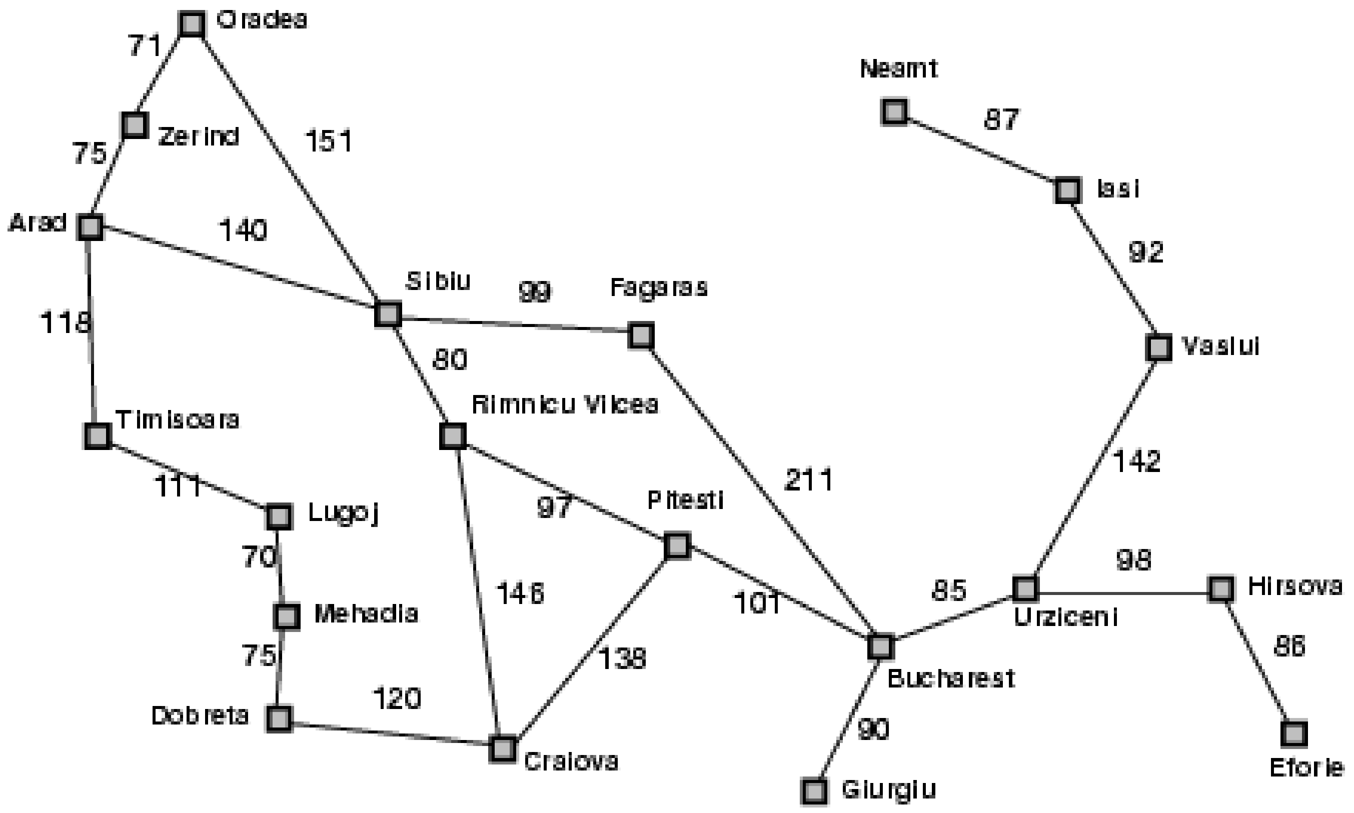
h' = estimate of the cost of getting from the current node to the goal state.

Algorithm : Best-First Search

```
1.    Start with OPEN containing the initial state.
2.    Until a goal is found or there are no nodes left in OPEN do
      (a)    Pick the best node in OPEN.
      (b)    Generate its successors.
      (c)    For each successor do
             (i)     if it has not been generated before, evaluate it,
                     add it to OPEN and record its parents.
             (ii)    if it has been generated before, change the
                      parent if this new path is better than the
                      previous one.
```

In that case, update the cost of getting to this node and to any successors that this node may already have.

Romania with step costs in km



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

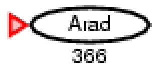
Evaluation function $f(n) = h(n)$ (heuristic)

= estimate of cost from n to *goal*

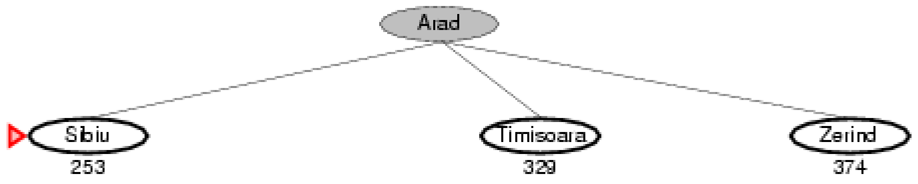
e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest

Greedy best-first search expands the node that appears to be closest to goal

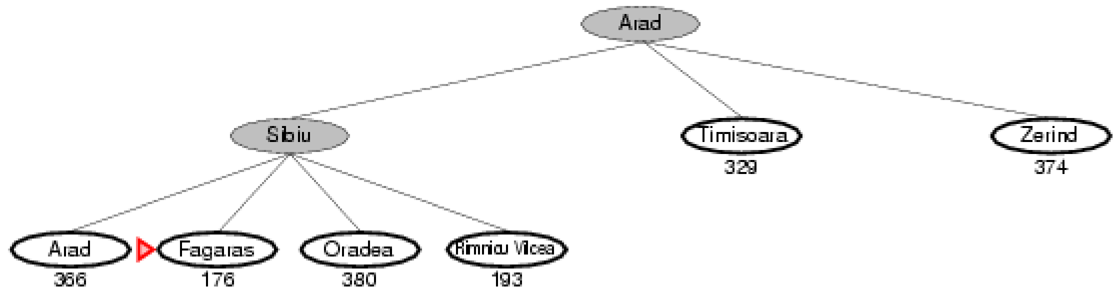
Greedy best-first search example



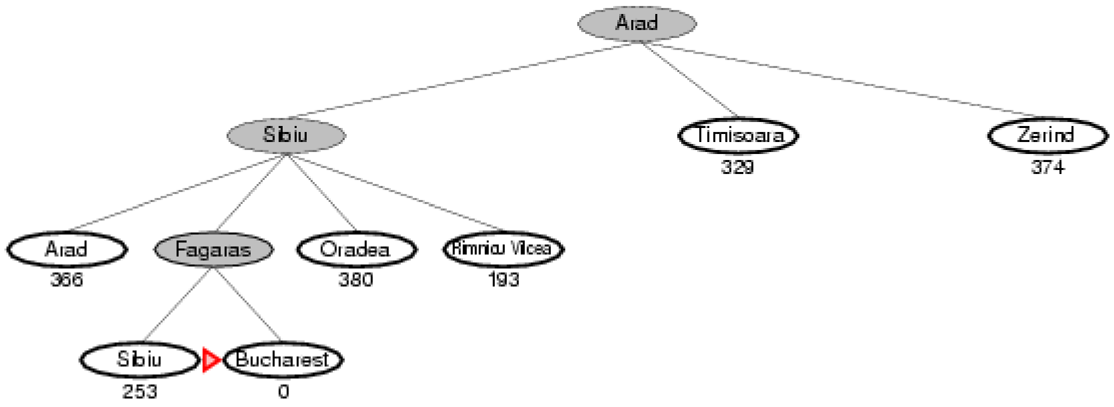
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

Complete? No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt →

Time? $O(b^m)$, but a good heuristic can give dramatic improvement

Space? $O(b^m)$ -- keeps all nodes in memory

Optimal? No