Introduction to Artificial Intelligence: Un-Informed Search

Dr. Tapas Kumar Mishra

Assistant Professor
Department of Computer Science and Engineering
SRM University
Amaravati-522502, Andhra Pradesh, India
Email: tapaskumar.m [at] srmap [dot] edu [dot] in

A* search

Idea: avoid expanding paths that are already expensive

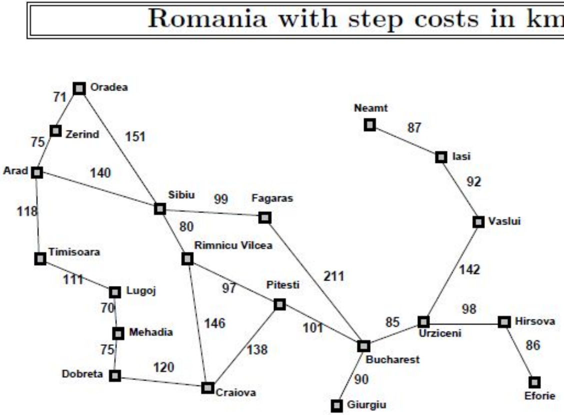
Evaluation function $f(n) = g(n) + h(n)$

$g(n)$ = cost so far to reach n
$h(n)$ = estimated cost to goal from n
$f(n)$ = estimated total cost of path through n to goal

A* search uses an admissible heuristic
i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n.
(Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G.)

E.g., $h_{\mathrm{SLD}}(n)$ never overestimates the actual road distance

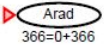
Theorem: A* search is optimal



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

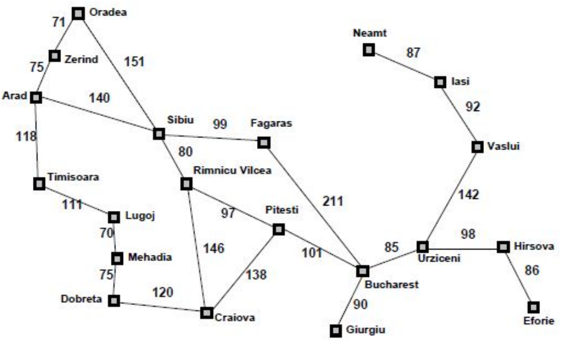
Informed Search - A∗ Search

A* search example

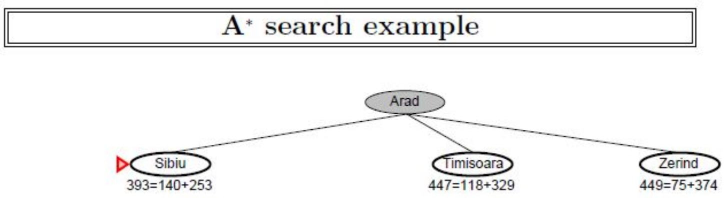


Informed Search - Best First Search

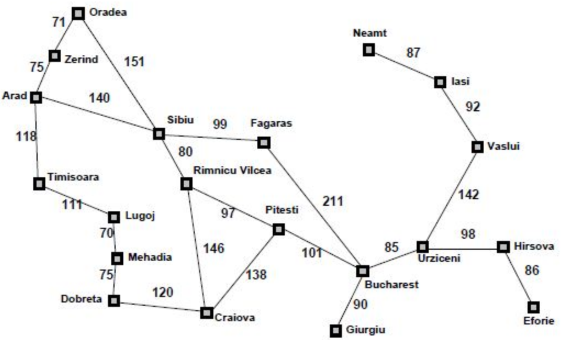
Romania with step costs in km



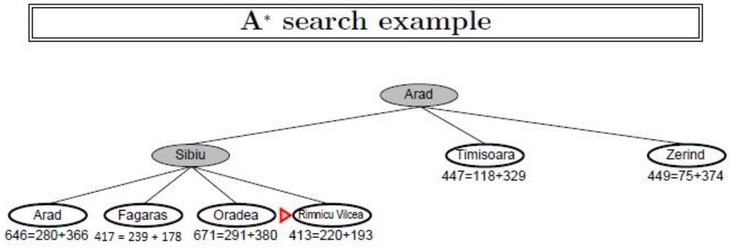
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374


A* search example
Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374

Romania with step costs in km

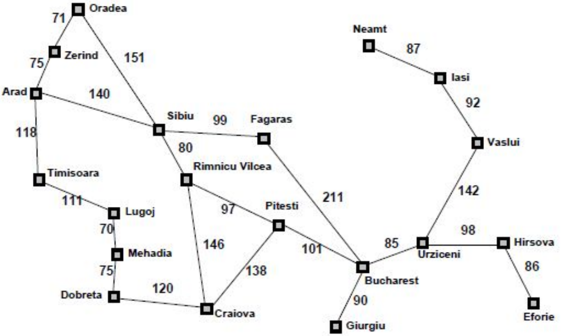


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

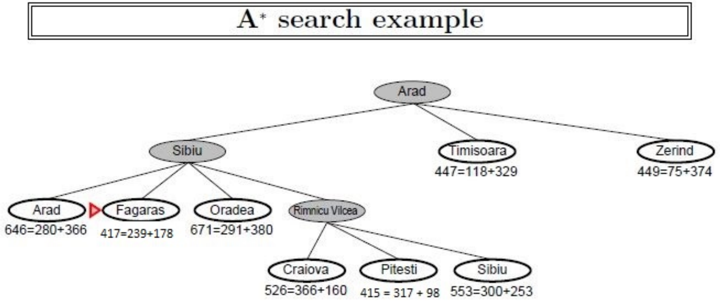


- Fagaras: $417 = 239 + 178$

Romania with step costs in km



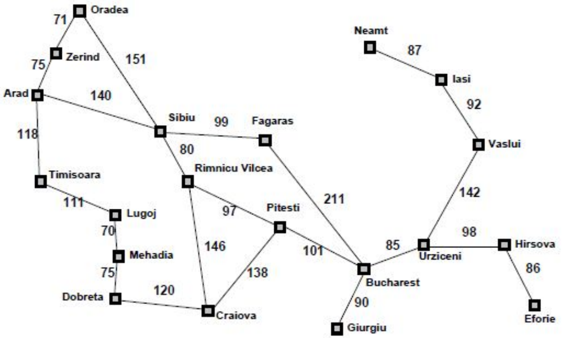
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



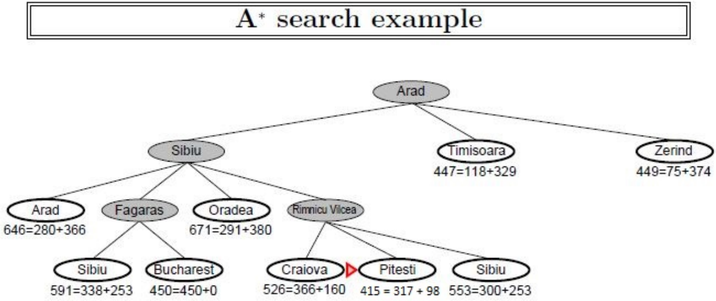
- Pitesti: $415 = 317 + 98$

Informed Search - Best First Search

Romania with step costs in km



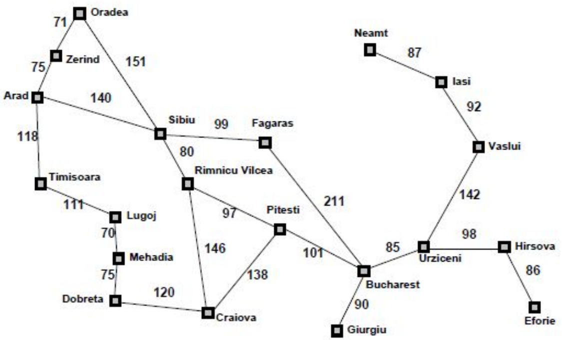
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



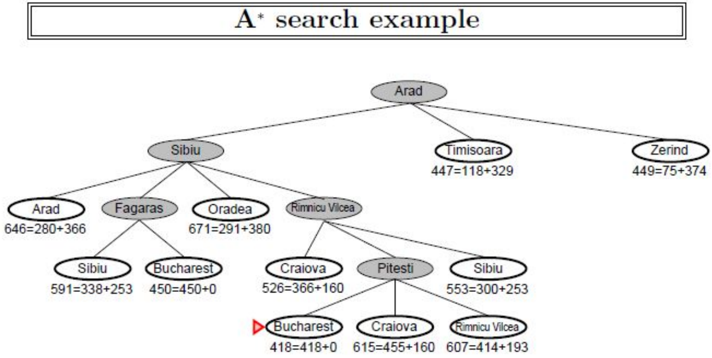
- Pitesti: $415 = 317 + 98$

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



- Pitesti: $415 = 317 + 98$

A^* Search

1. Place the starting node s on open.
2. If open is empty, stop and return failure.
3. Remove from open the node n that has the smallest value of $f^*(n)$. If the node is a goal node, return success and stop. Otherwise,
4. Expand n, generating all of its successors n' and place n on closed. For every successor n', if n' is not already on open or closed attach a back-pointer to n, compute $f^*(n')$ and place it on open.
5. Each n' that is already on open or closed should be attached to back-pointers which reflect the lowest $g^*(n')$ path. If n' was on closed and its pointer was changed, remove it and place it on open.
6. Return to step 2.

A^* Search

Admissibility Condition

Algorithm A is admissible if it is guaranteed to return an optimal solution when one exists.

Completeness Condition

Algorithm A is complete if it always terminates with a solution when one exists.

Dominance Condition

Let A_1 and A_2 be admissible algorithms with heuristics estimation functions h_1^* and h_2^*, respectively. A_1 is said to be more informed than A_2 whenever $h_1^*(n) > h_2^*(n)$ for all n. A_1 is also said to dominate A_2.

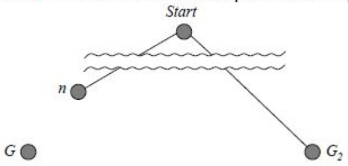
Optimality Condition

Algorithm A is optimal over a class of algorithms if A dominates all members of the class.

Informed Search - A$*$ Search

Optimality of A* (standard proof)

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.



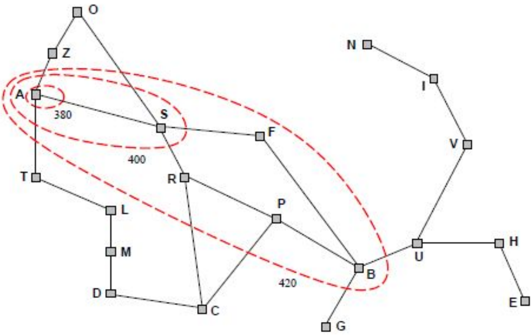
$$
\begin{aligned}
f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\
&> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\
&\geq f(n) && \text{since } h \text{ is admissible}
\end{aligned}
$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Optimality of A* (more useful)

Lemma: A* expands nodes in order of increasing f value*

Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

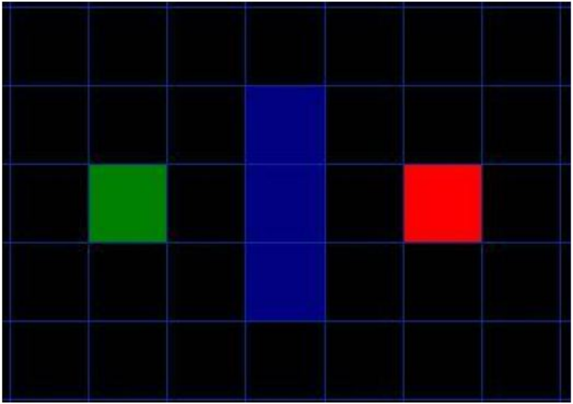
Time?? Exponential in [relative error in $h \times$ length of soln.]

Space?? Keeps all nodes in memory

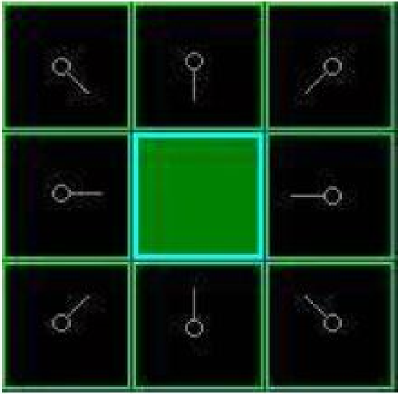
Optimal?? Yes—cannot expand f_{i+1} until f_i is finished

A* expands all nodes with $f(n) < C^*$
A* expands some nodes with $f(n) = C^*$
A* expands no nodes with $f(n) > C^*$

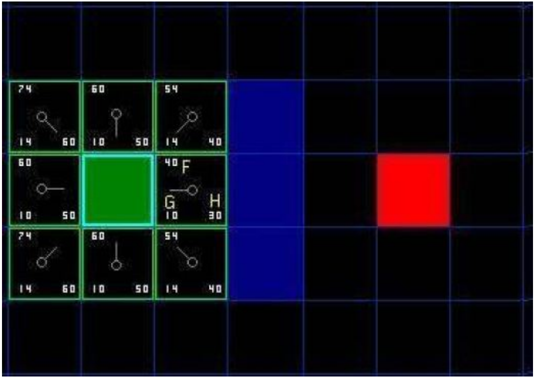
- Let's assume that we have someone who wants to get from **point A** to **point B**. Let's assume that a wall separates the two points.



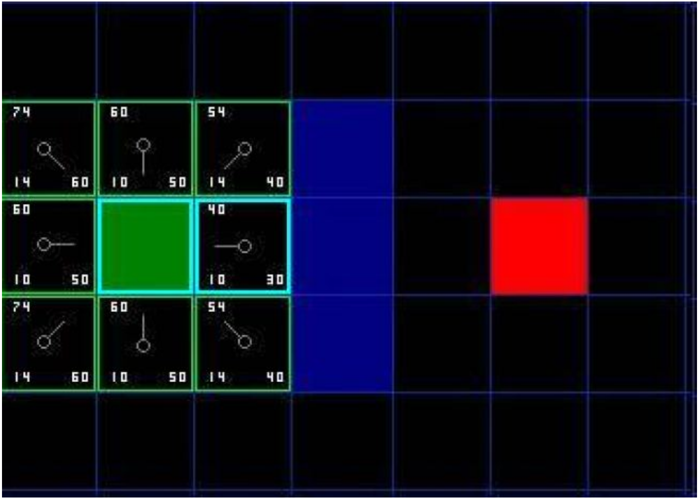
- Status: Walkable or Unwalkable
- Look at all the reachable or walkable squares adjacent to the starting point, ignoring squares with walls, water, or other illegal terrain.



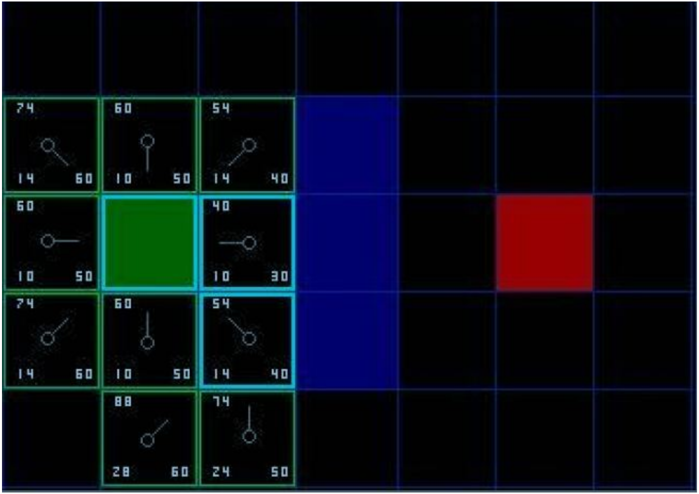
- We will assign a cost of **10** to each **horizontal** or **vertical** square moved and a cost of **14** for a **diagonal** move
- The method we use here is called the **Manhattan method**, where you calculate the total number of squares moved horizontally and vertically to reach the target square from the current square, **ignoring diagonal movement** and **ignoring any obstacles** that may be in the way.



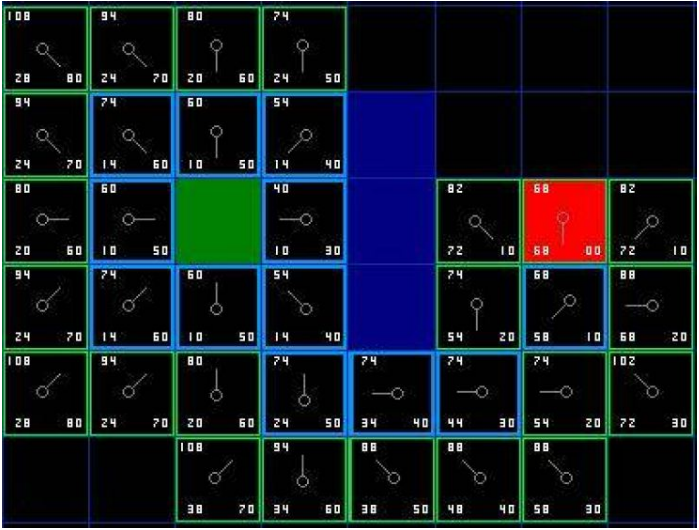
Informed Search - A∗ Search



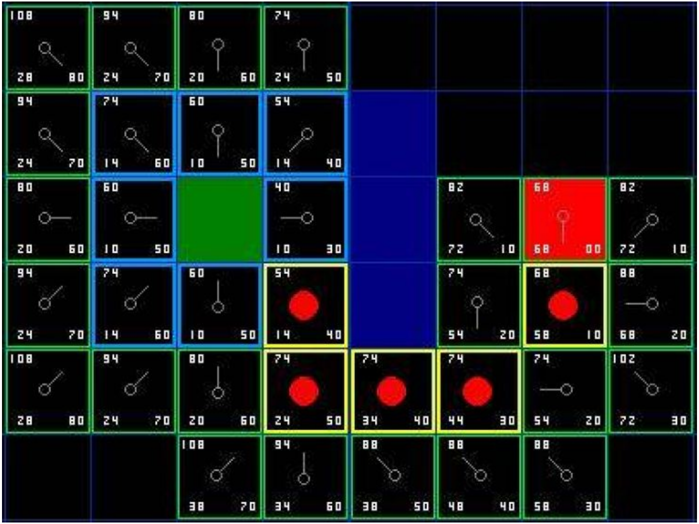
Informed Search - A∗ Search



Informed Search - A∗ Search



Informed Search - A∗ Search



Thank You!