ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Artificial Intelligence

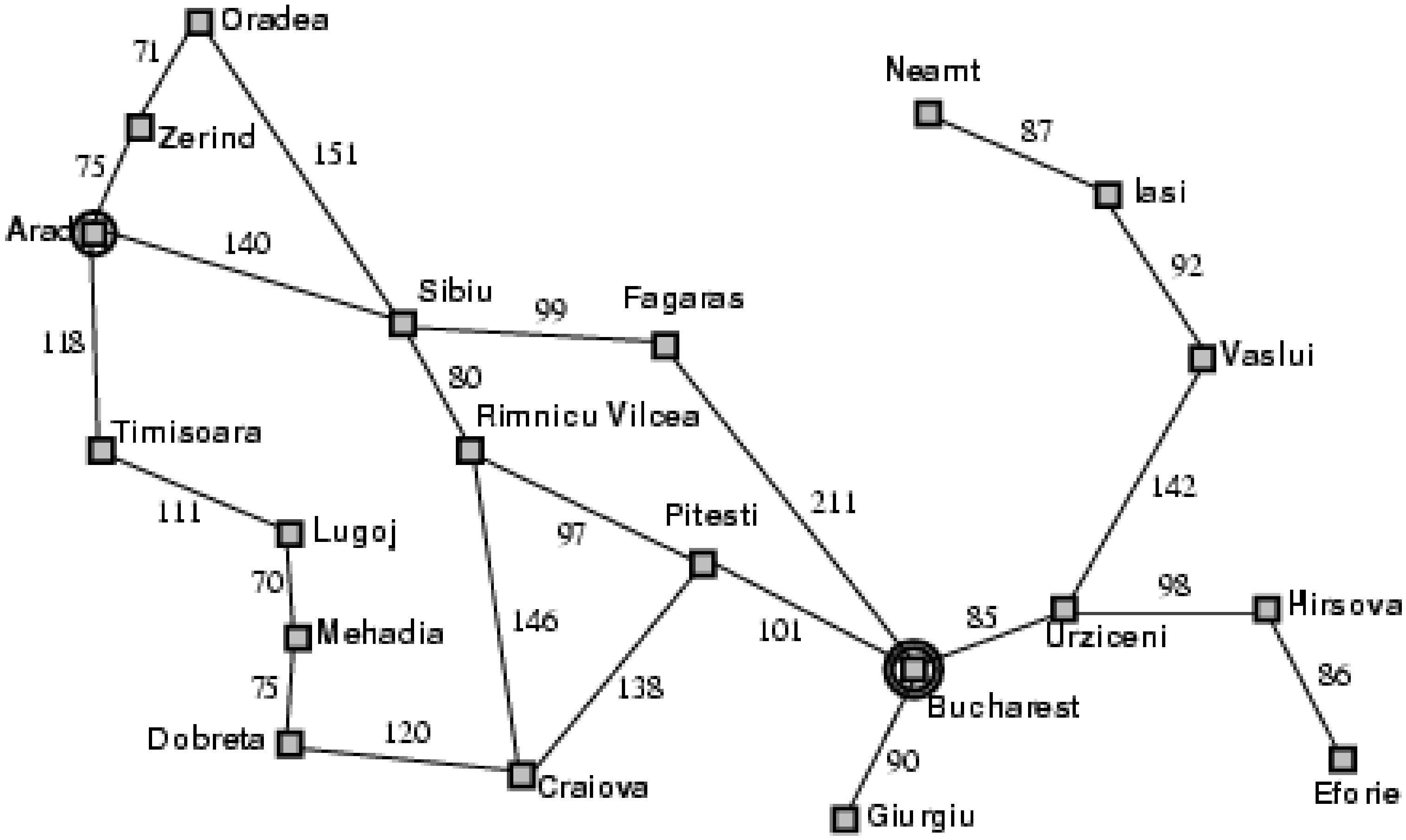
Lecture 4 - Search

School of Information and Communication Technology - HUST

Outline

- Graph search
- Best-first search
- A* search

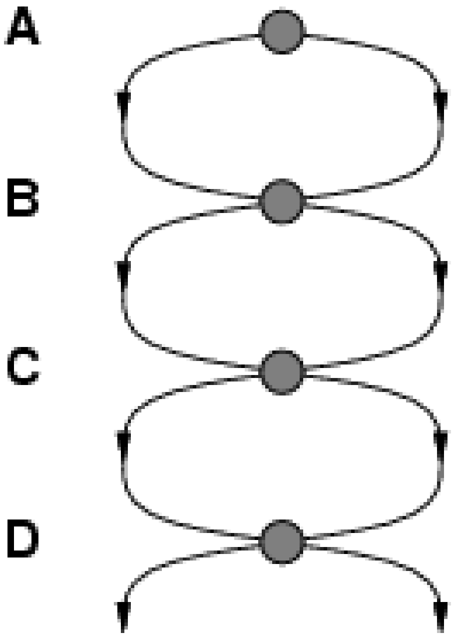
Graph search

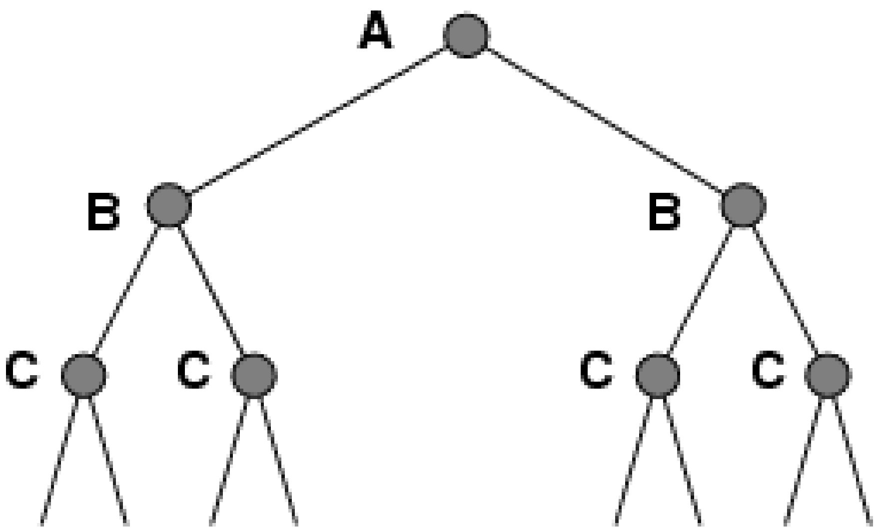


Get from Arad to Bucharest as quickly as possible

Graph search

- Failure to detect repeated states can turn a linear problem into an exponential one!





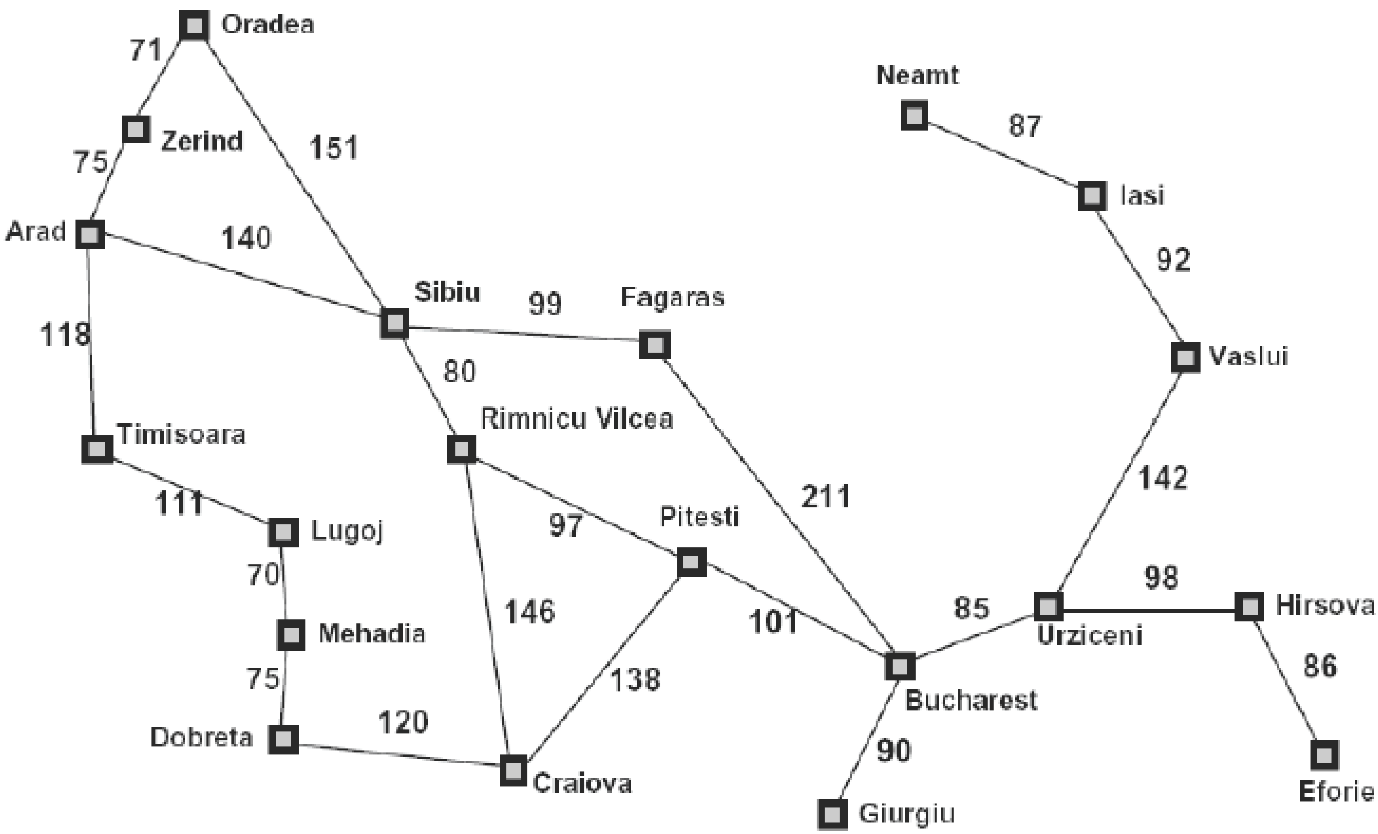
- Very simple fix: never expand a node twice

Graph search

```
function Graph-Search(problem, fringe) returns a solution, or failure
fringe ← Insert(Make-Node(Initial-State(problem)), fringe);
closed ← an empty set
while (fringe not empty)
   node ← RemoveFirst(fringe);
   if (Goal-Test(problem, State(node))) then return Solution(node);
   if (State(node) is not in closed then
      add State(node) to closed
      fringe ← InsertAll(Expand(node, problem), fringe);
   end if
end
return failure;
```

- Never expand a node twice!

Straight Line Distances



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

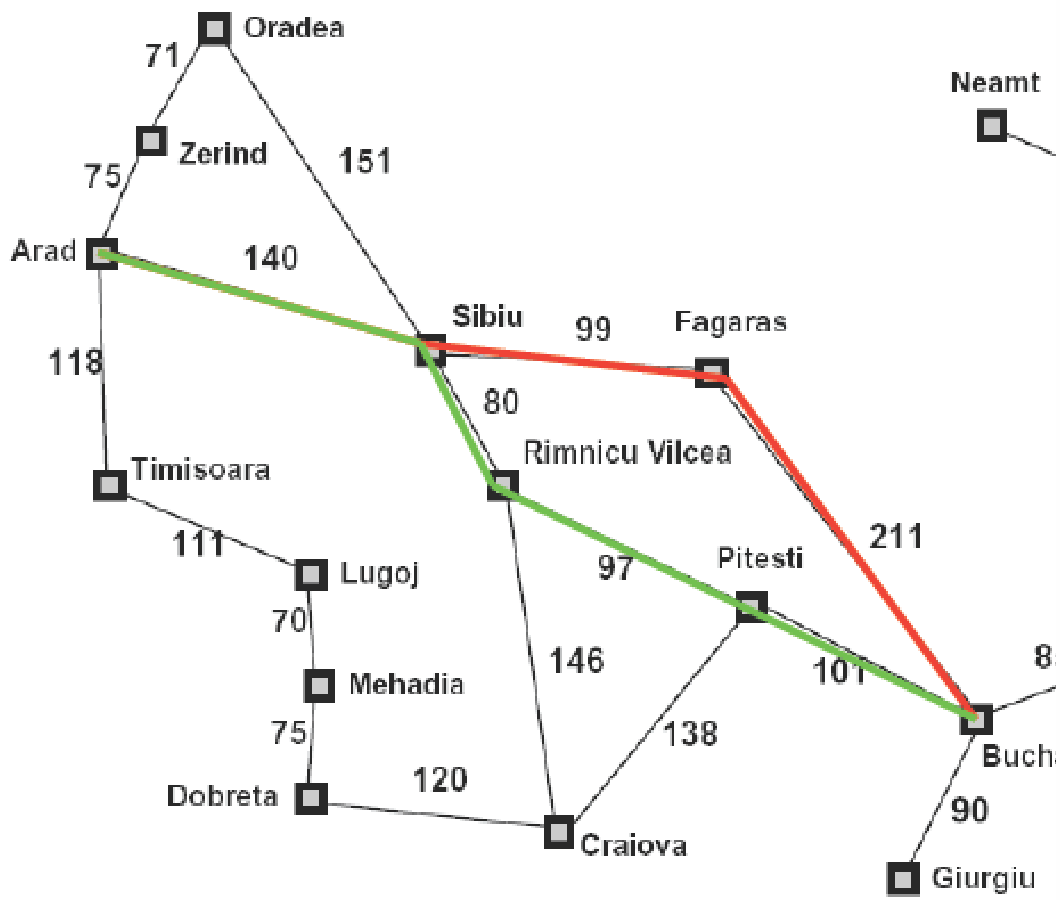
Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability"

  →Expand most desirable unexpanded node

- Order the nodes in fringe in decreasing order of desirability

- Special cases:
  - greedy best-first search
  - A^* search



Straight−line distance to Bucharest

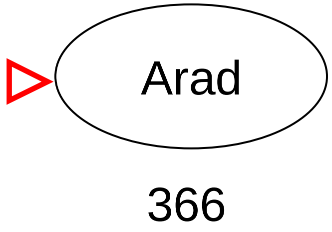
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First Search

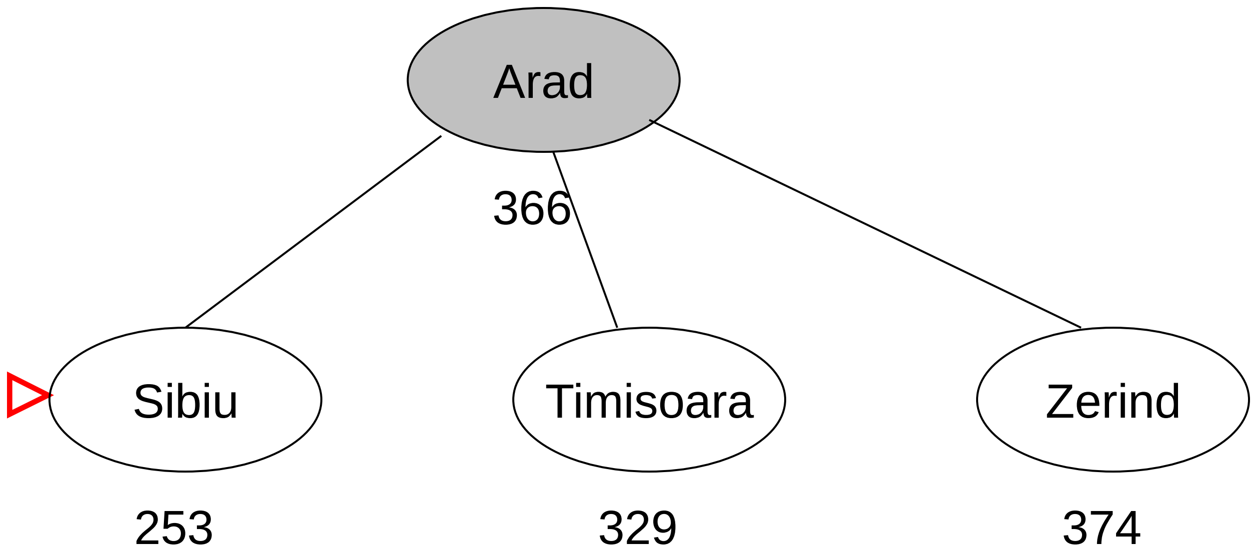
- Evaluation function $f(n) = h(n)$ (heuristic)

  = 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

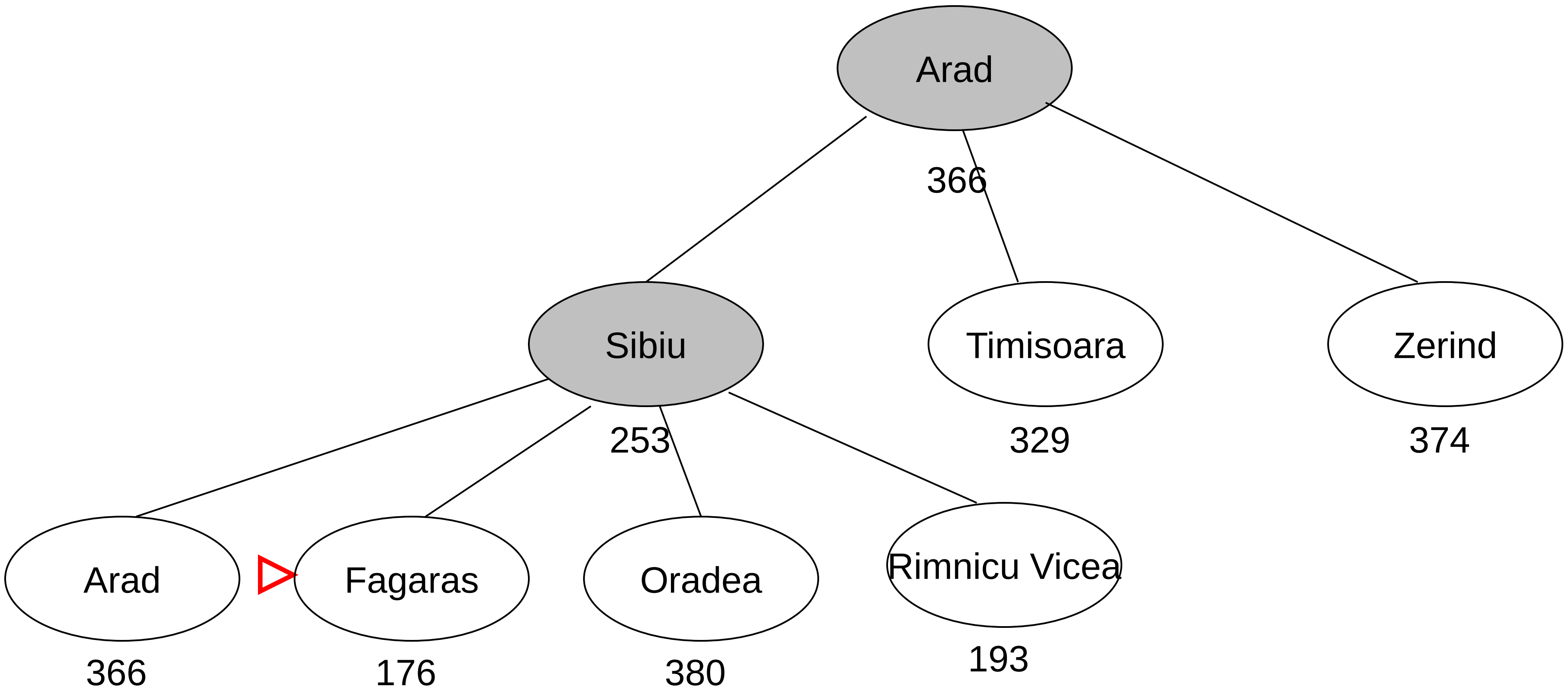
Greedy best-first search example



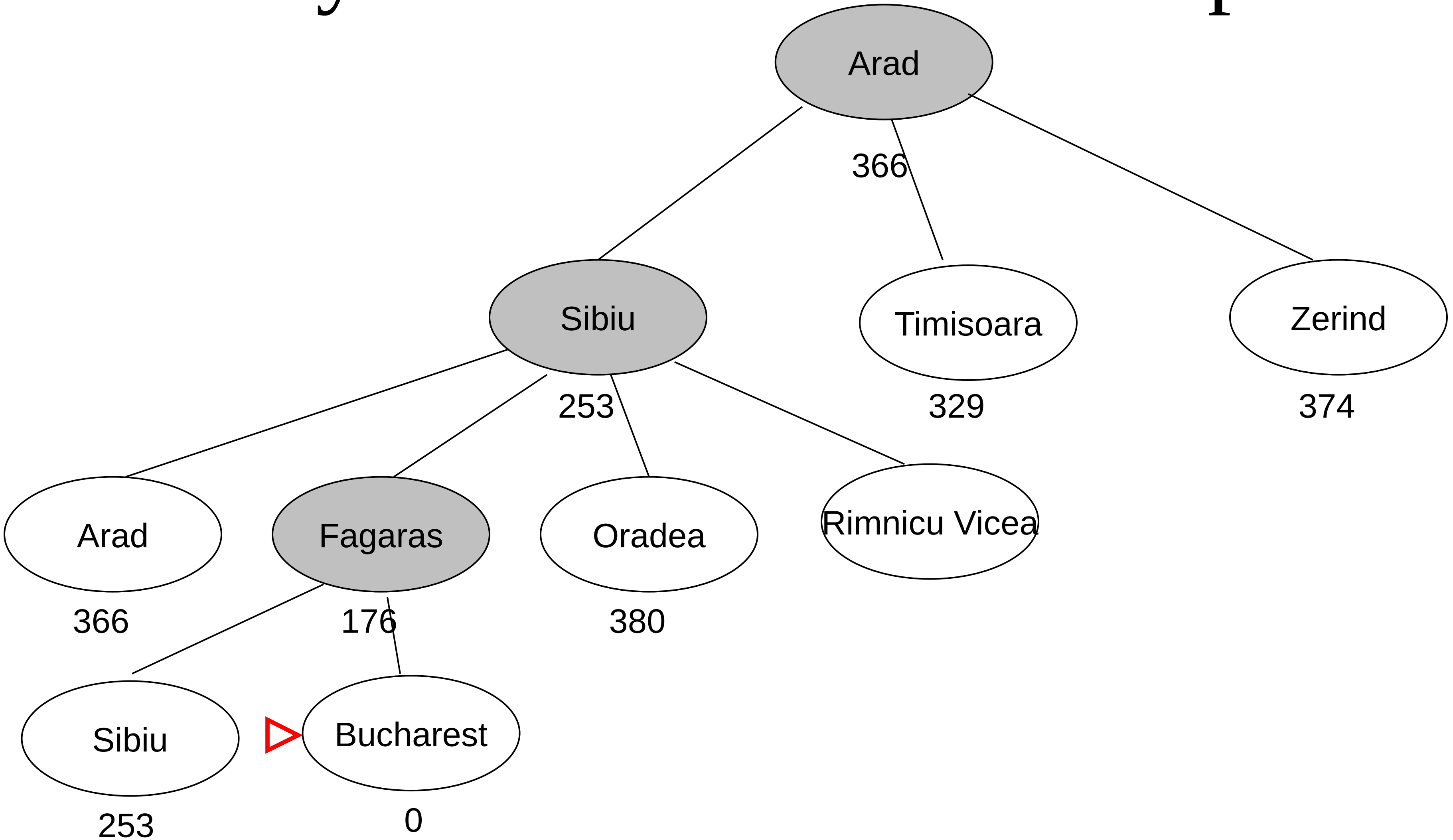
Greedy best-first search example



Greedy best-first search example

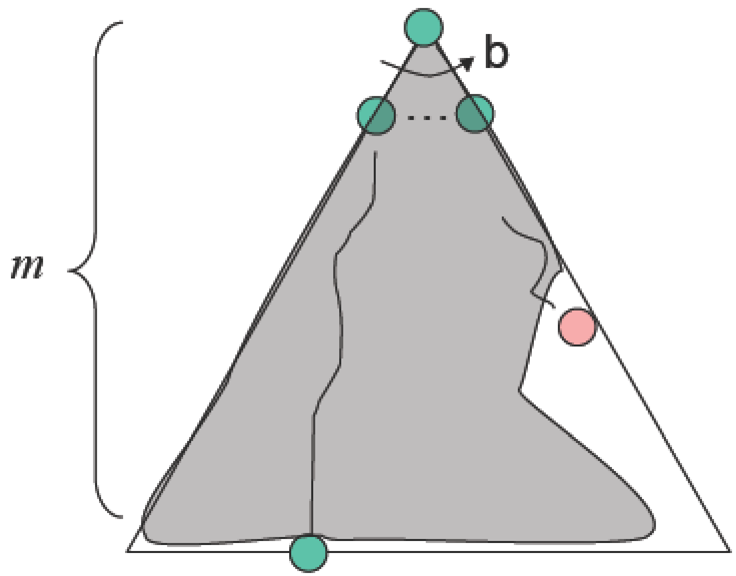


Greedy best-first search example



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



- What do we need to do to make it complete?

⇒ A* search

- Can we make it optimal? → No

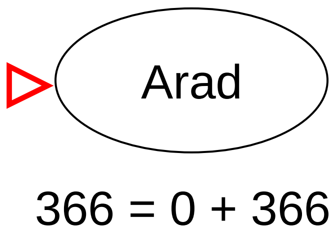
A* search

- Idea: Expand unexpanded node with lowest evaluation value

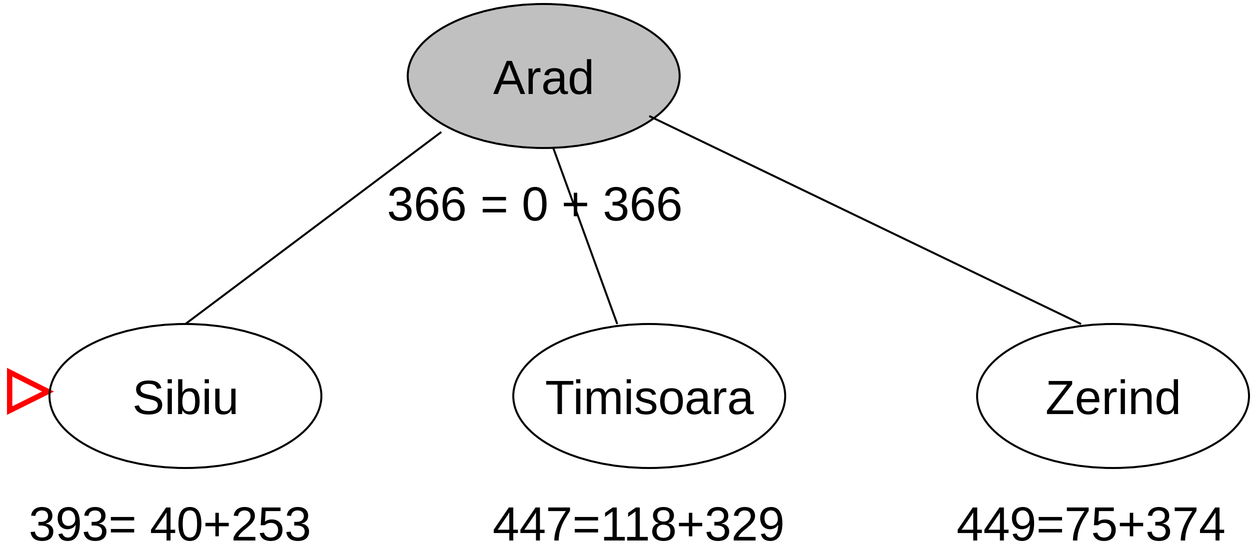
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

- Nodes are ordered according to f(n).

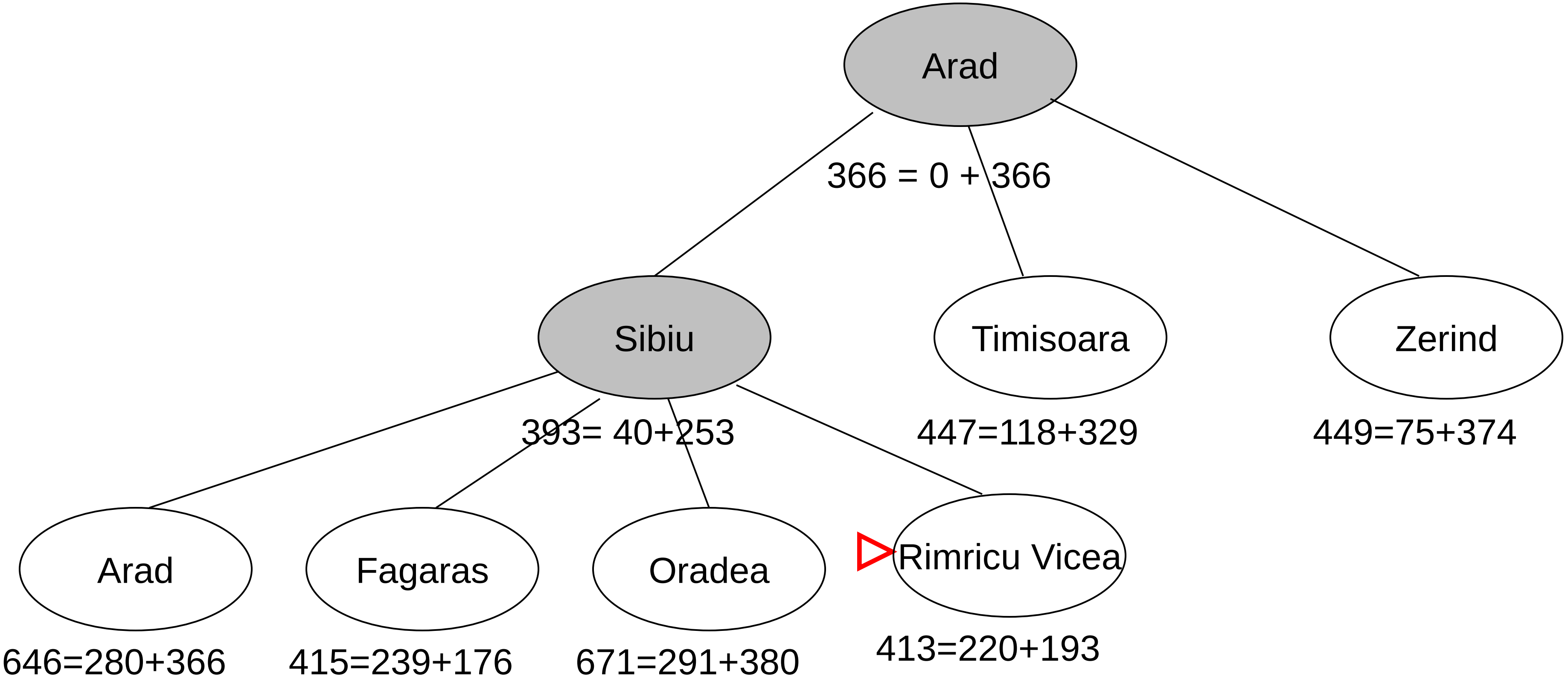
A* search example



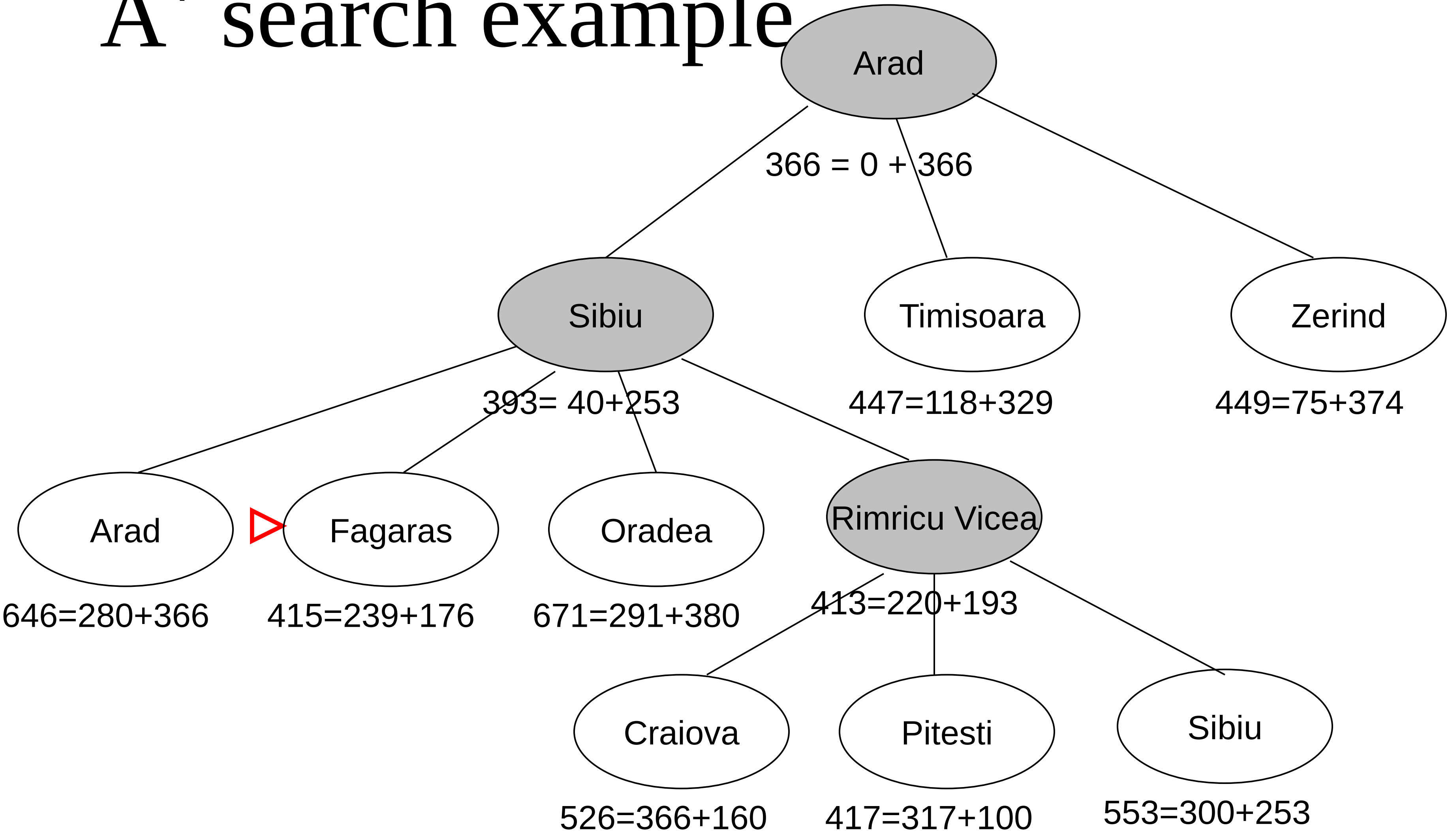
A* search example



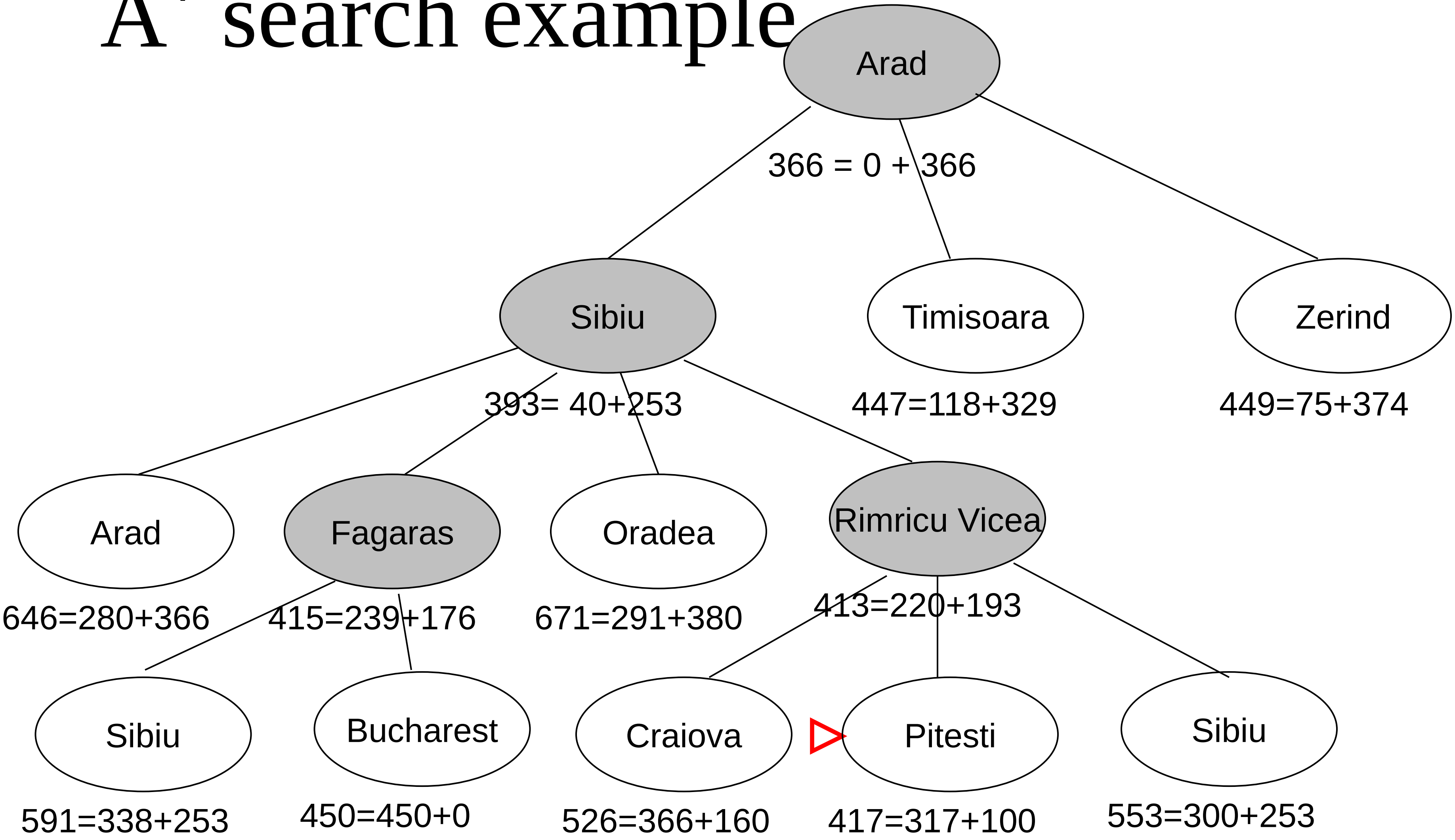
A* search example



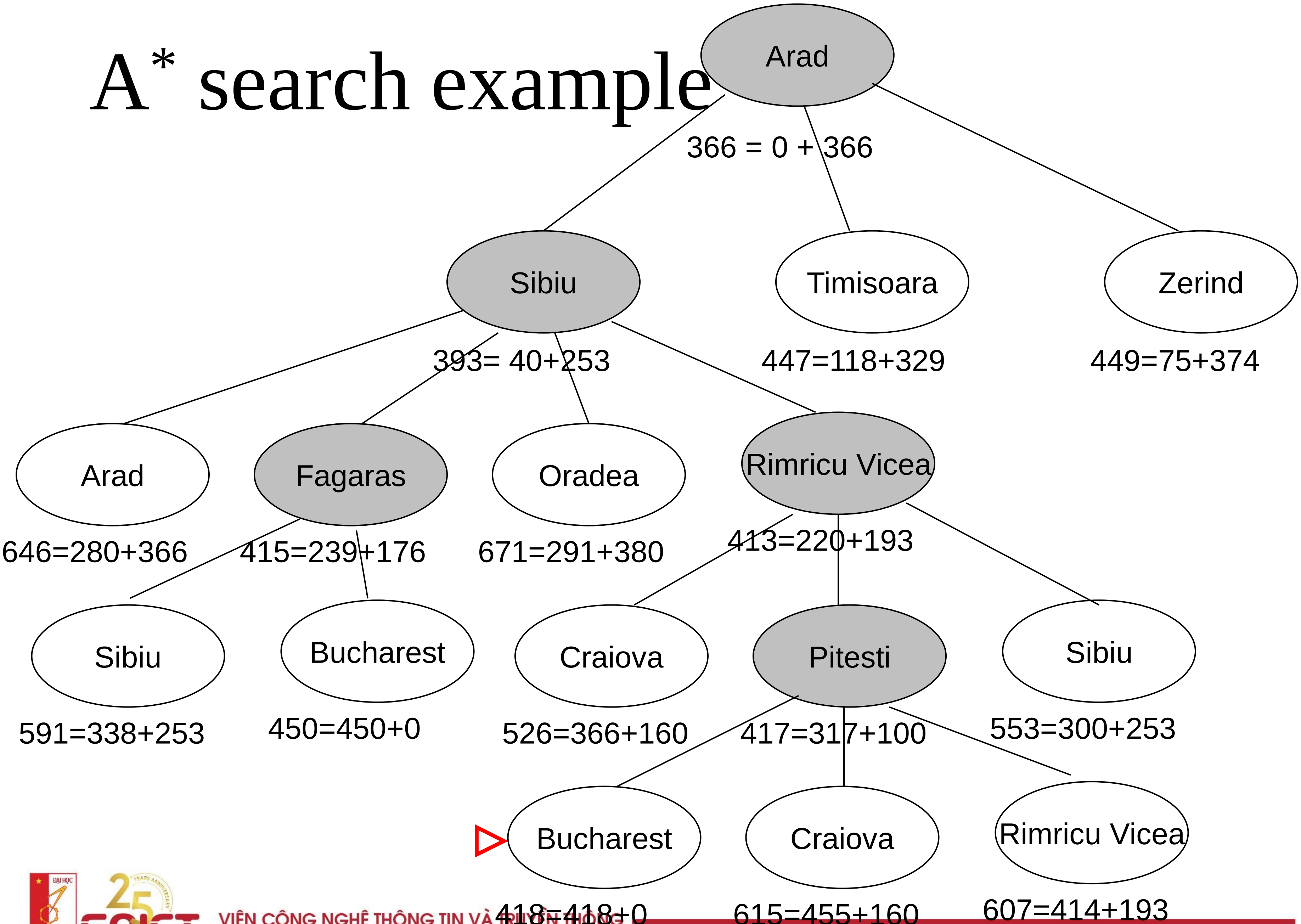
A* search example



A* search example



A* search example



Can we Prove Anything?

- If the state space is finite and we avoid repeated states, the search is complete
- If the state space is finite and we do not avoid repeated states, the search is in general not complete
- If the state space is infinite, the search is in general not complete

Admissible heuristic

- Let $h^*(N)$ be the **true** cost of the optimal path from N to a goal node
- Heuristic $h(N)$ is admissible if:

$$0 \leq h(N) \leq h^*(N)$$

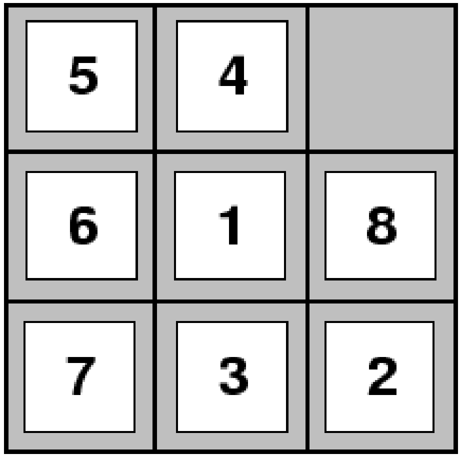
- An admissible heuristic is always optimistic

Admissible heuristics

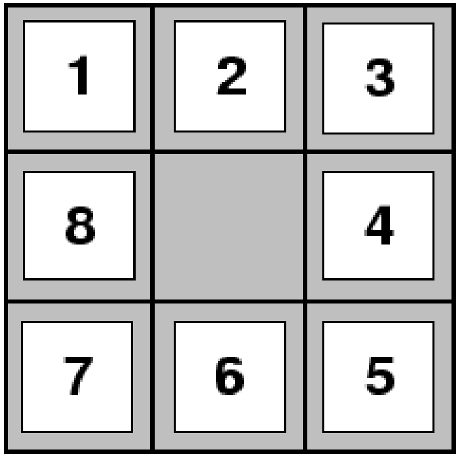
The 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



Start State



Goal State

- <u>$h_1(S)$ = ?</u> 7
- <u>$h_2(S)$ = ?</u> 2+3+3+2+4+2+0+2 = 18

Heuristic quality

- Effective branching factor b*
  - Is the branching factor that a uniform tree of depth *d* would have in order to contain *N+1* nodes.

$$N + 1 = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

  - Measure is fairly constant for sufficiently hard problems.
    - Can thus provide a good guide to the heuristic's overall usefulness.
    - A good value of b* is 1.

Heuristic quality and dominance

- 1200 random problems with solution lengths from 2 to 24.
- If $h_2(n) >= h_1(n)$ for all *n* (both admissible)
  then h_2 *dominates* h_1 and is better for search

	Search Cost			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Inventing admissible heuristics

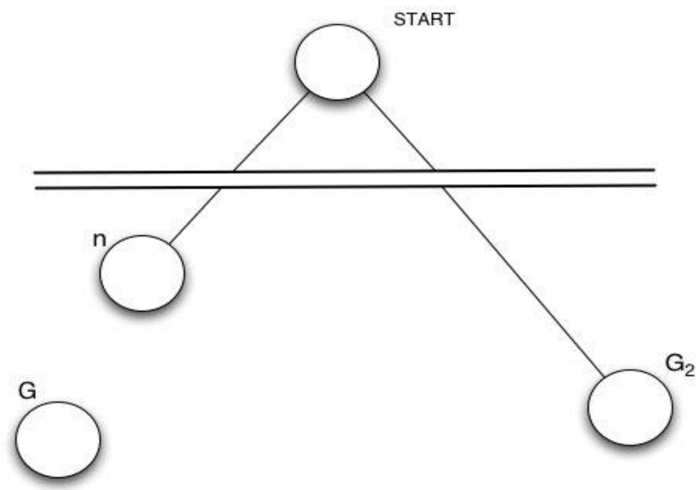
- Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem:
  - Relaxed 8-puzzle for h_1 : a tile can move anywhere

    As a result, $h_1(n)$ gives the shortest solution
  - Relaxed 8-puzzle for h_2 : a tile can move to any adjacent square.

    As a result, $h_2(n)$ gives the shortest solution.

The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem.

Optimality of A*(standard proof)



- Suppose suboptimal goal G_2 in the queue.
- Let n be an unexpanded node on a shortest to optimal goal G.

$f(G_2)$	$= g(G_2)$	since $h(G_2)=0$
	$> g(G)$	since G_2 is suboptimal
	$>= f(n)$	since h is admissible

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

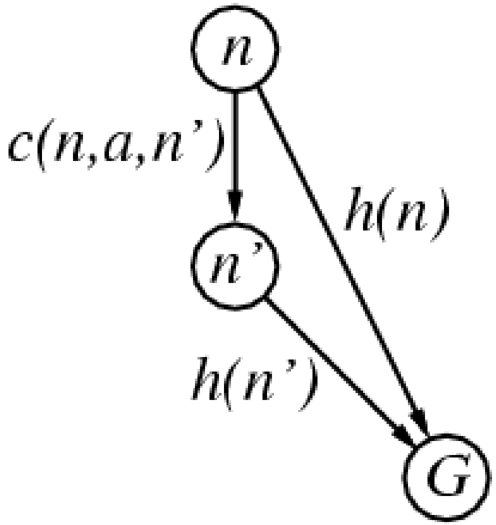
Optimality for graphs?

- Admissibility is not sufficient for graph search
  - In graph search, the optimal path to a repeated state could be discarded if it is not the first one generated
  - Can fix problem by requiring <u>consistency property</u> for h(n)

- A heuristic is consistent if for every successor *n'* of a node *n* generated by any action *a*,

$$h(n) \leq c(n,a,n') + h(n')$$

(aka "monotonic")



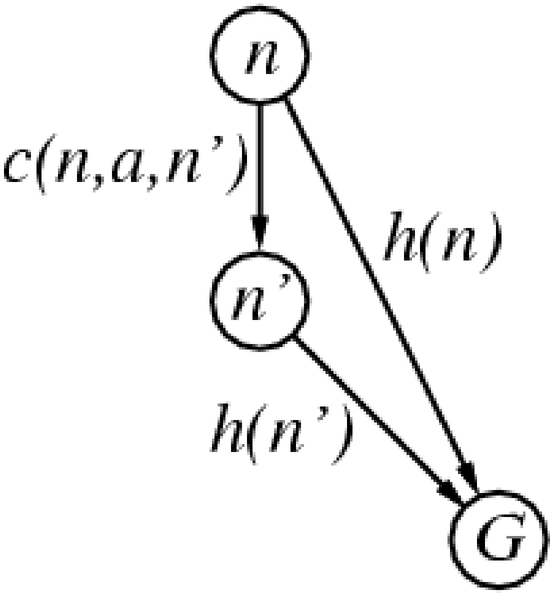
- admissible heuristics are generally consistent

A* is optimal with consistent heuristics

- If *h* is consistent, we have

$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n,a,n') + h(n') \\
&\geq g(n) + h(n) \\
&= f(n)
\end{aligned}
$$

i.e., *f(n)* is non-decreasing along any path.

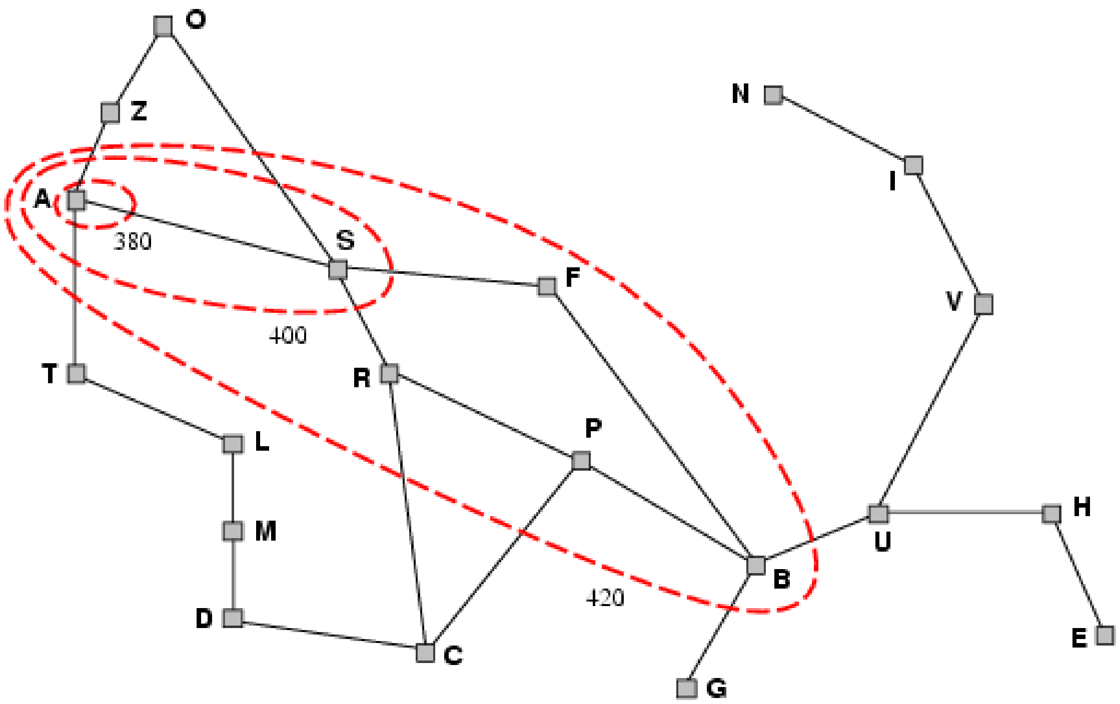


Thus, first goal-state selected for expansion must be optimal

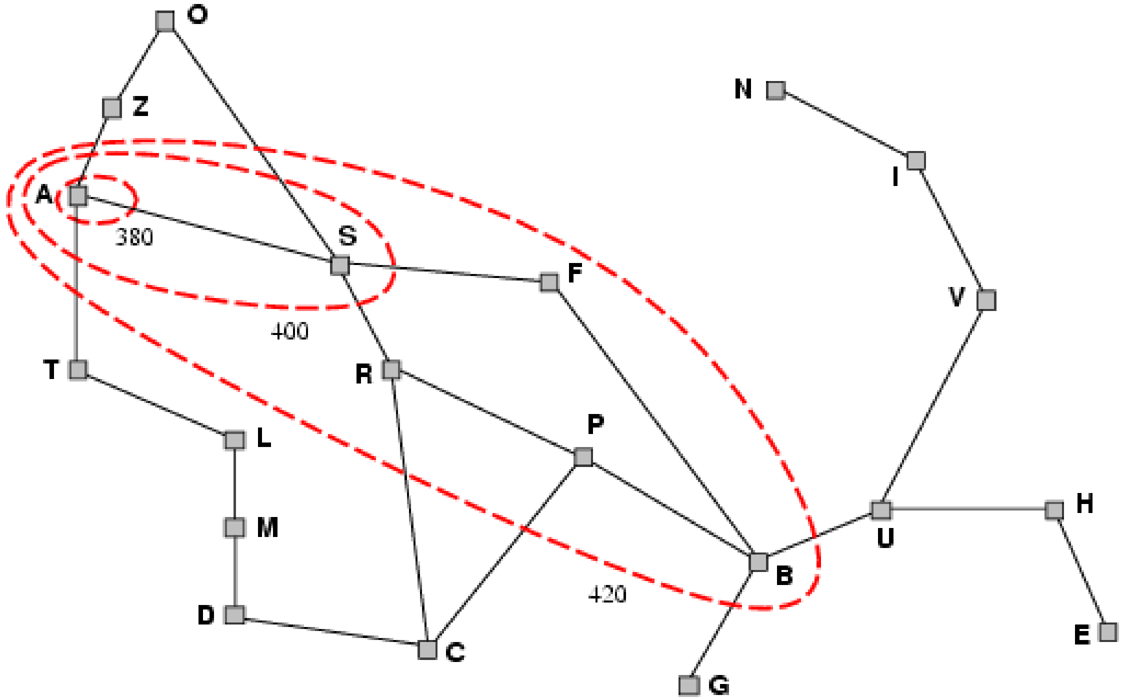
- Theorem:
  - If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal
  -

Contours of A* Search

- A* expands nodes in order of increasing *f* value
- Gradually adds "*f*-contours" of nodes
- Contour *i* has all nodes with $f=f_i$, where $f_i < f_{i+1}$



Contours of A* Search



- With uniform-cost (h(n) = 0, contours will be circular
- With good heuristics, contours will be focused around optimal path
- A* will expand all nodes with cost f(n) < C*

A* search, evaluation

- Completeness: YES
  - Since bands of increasing *f* are added
  - Unless there are infinitely many nodes with $f<f(G)$

A* search, evaluation

- Completeness: YES
- Time complexity:
  - Number of nodes expanded is still exponential in the length of the solution.

A* search, evaluation

- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity:
  - It keeps all generated nodes in memory
  - Hence space is the major problem not time

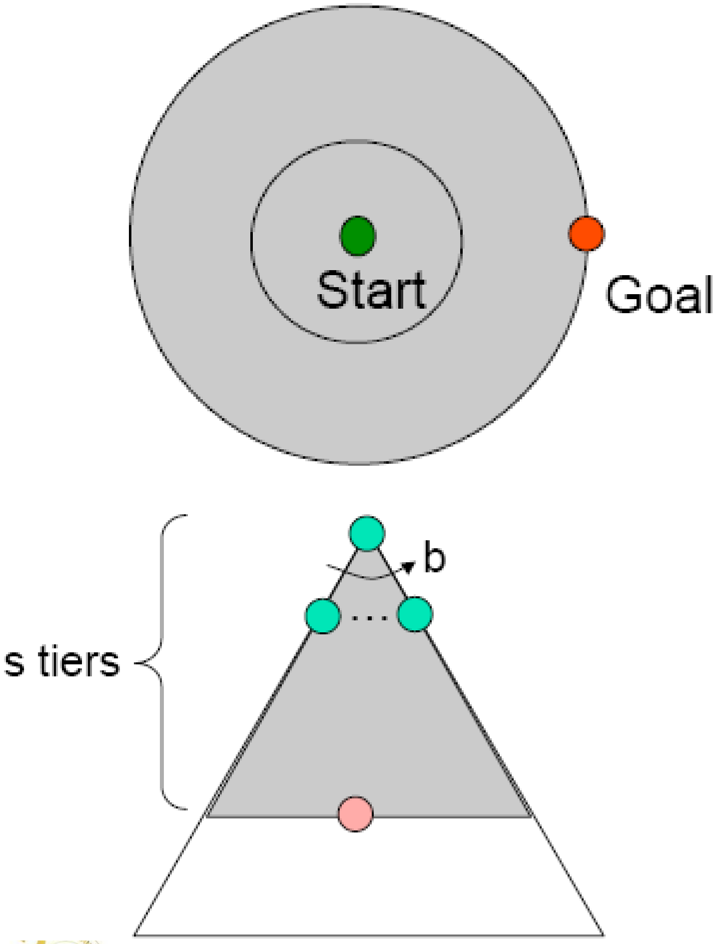
A* search, evaluation

- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity:(all nodes are stored)
- Optimality: 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 $f(n)>C^*$

Also *optimally efficient* (not including ties)

Compare Uniform Cost and A*

- Uniform-cost expanded in all directions



- A* expands mainly toward the goal, but does hedge its bets to ensure optimality