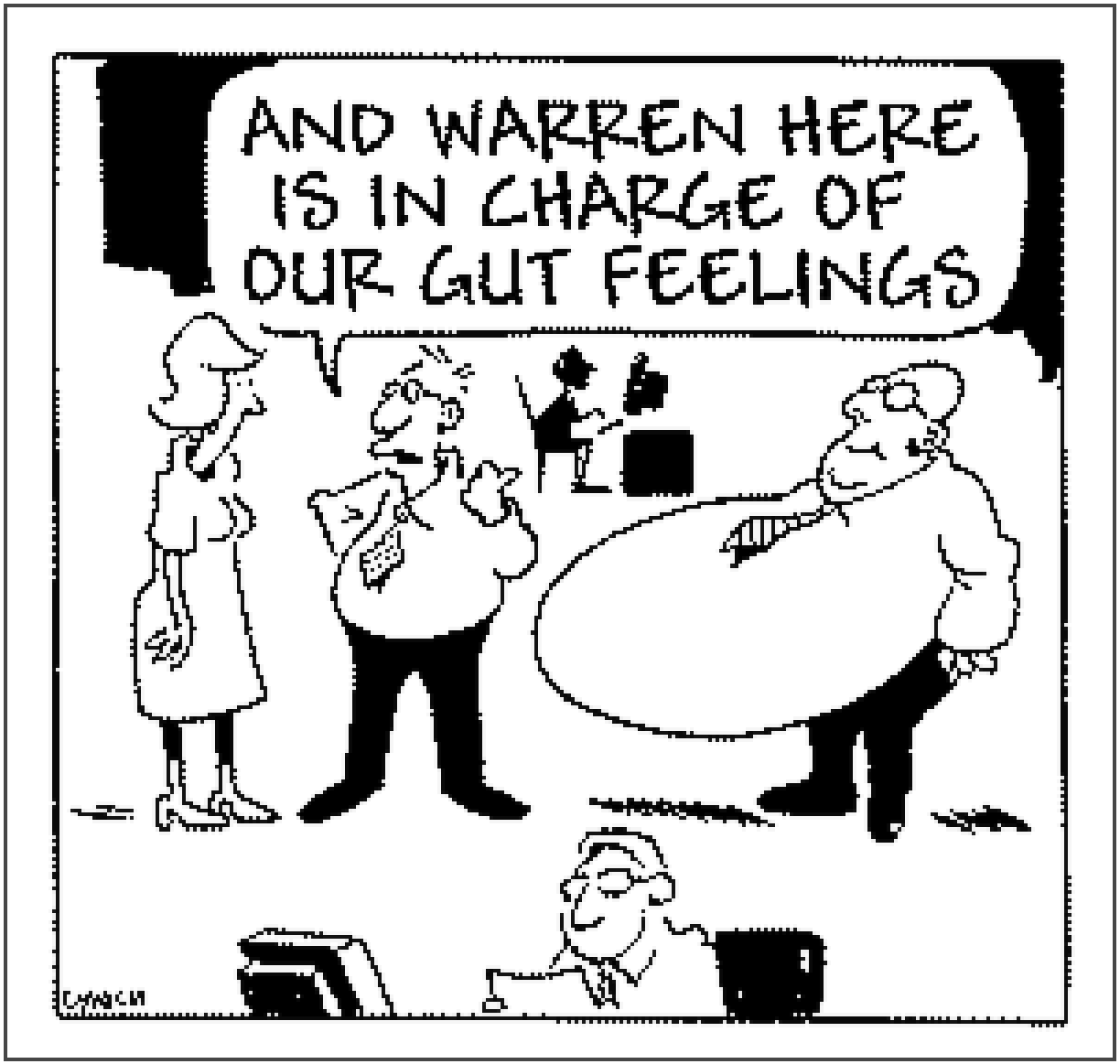
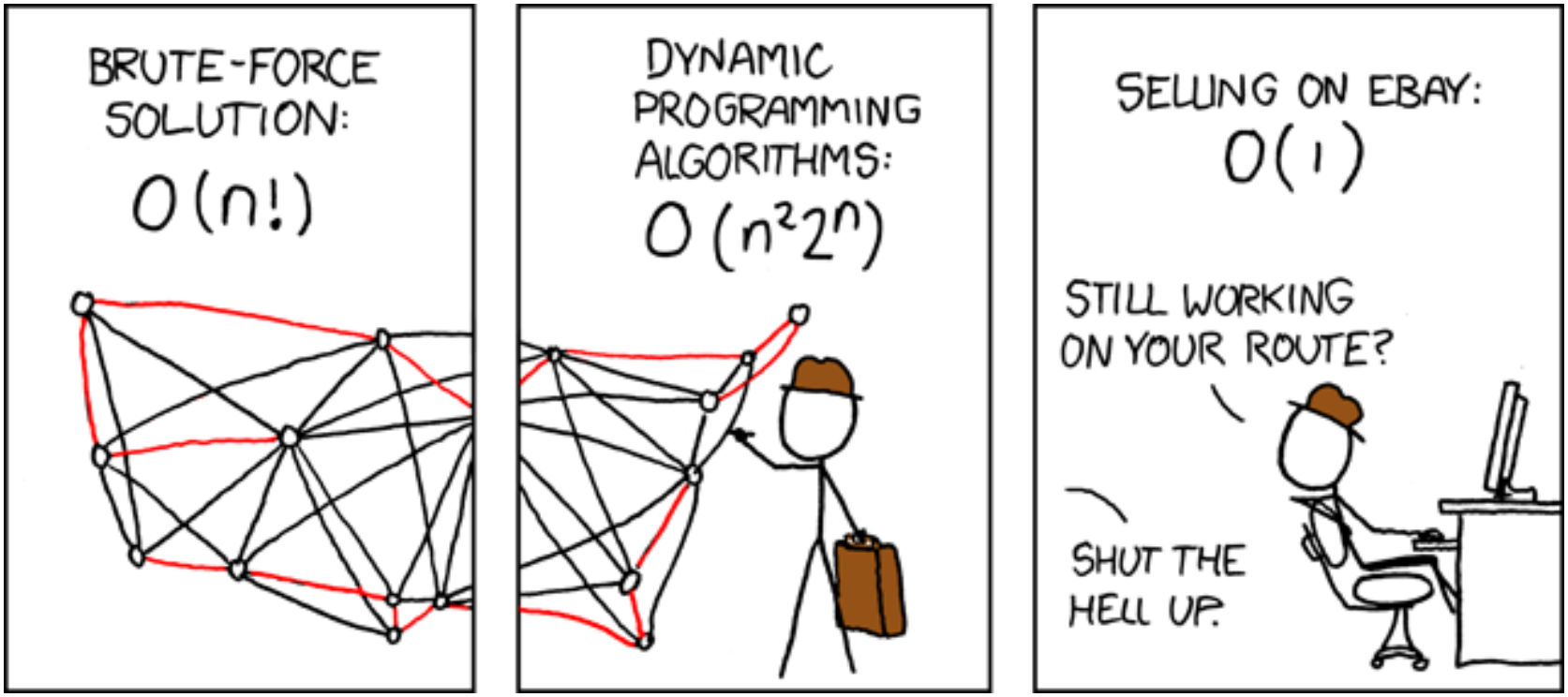
Informed search algorithms

Chapter 3

(Based on Slides by Stuart Russell,
Dan Klein, Richard Korf, Carl Kingsford, Subbarao Kambhampati, Eric Ringger, and UW-AI faculty)

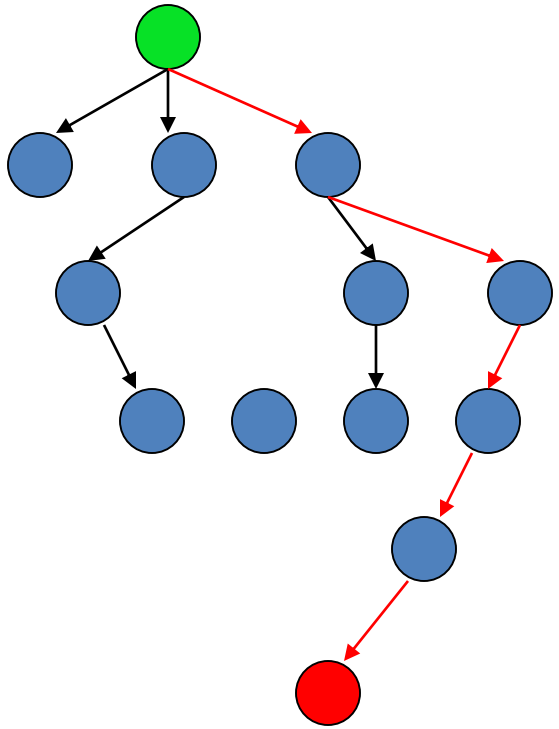
“Intuition, like the rays of the sun, acts only in an inflexibly straight line; it can guess right only on condition of never diverting its gaze; the freaks of chance disturb it.”


AND WARREN HERE IS IN CHARGE OF OUR GUT FEELINGS


BRUTE-FORCE SOLUTION:
O(n!)
DYNAMIC PROGRAMMING ALGORITHMS:
O(n²2ⁿ)
SELLING ON EBAY:
O(1)
STILL WORKING ON YOUR ROUTE?
SHUT THE HELL UP.

Informed (Heuristic) Search

Idea: be **smart** about what paths to try.



Blind Search vs. Informed Search

- What's the difference?

- How do we formally specify this?

  A node is selected for expansion based on an evaluation function that estimates cost to goal.

General Tree Search Paradigm

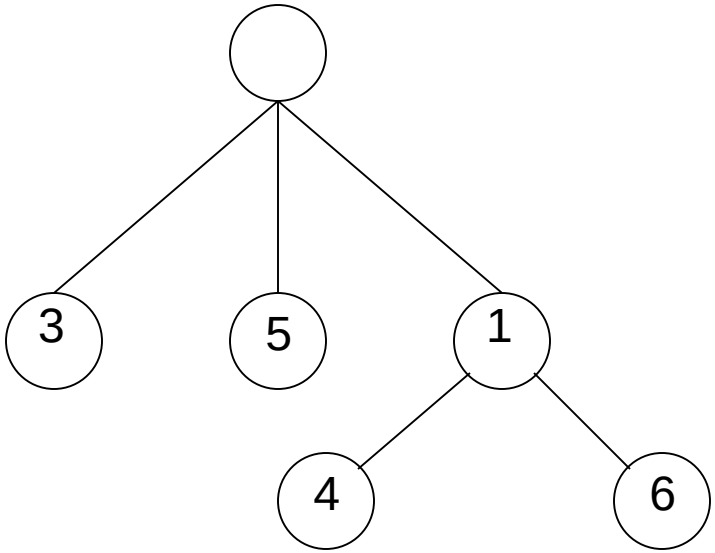
```
function tree-search(root-node)
   fringe ← successors(root-node)
   while ( notempty(fringe) )
         {node ← remove-first(fringe)
             state ← state(node)
             if goal-test(state) return solution(node)
             fringe ← insert-all(successors(node),fringe) }
   return failure
end tree-search
```

General Graph Search Paradigm

```
function tree-search(root-node)
   fringe ← successors(root-node)
   explored ← empty
   while ( notempty(fringe) )
         {node ← remove-first(fringe)
             state ← state(node)
             if goal-test(state) return solution(node)
             explored ← insert(node,explored)
             fringe ← insert-all(successors(node),fringe, if node not in explored)
         }
   return failure
end tree-search
```

Best-First Search

- Use an evaluation function f(n) for node n.
- Always choose the node from fringe that has the lowest f value.



Best-first search

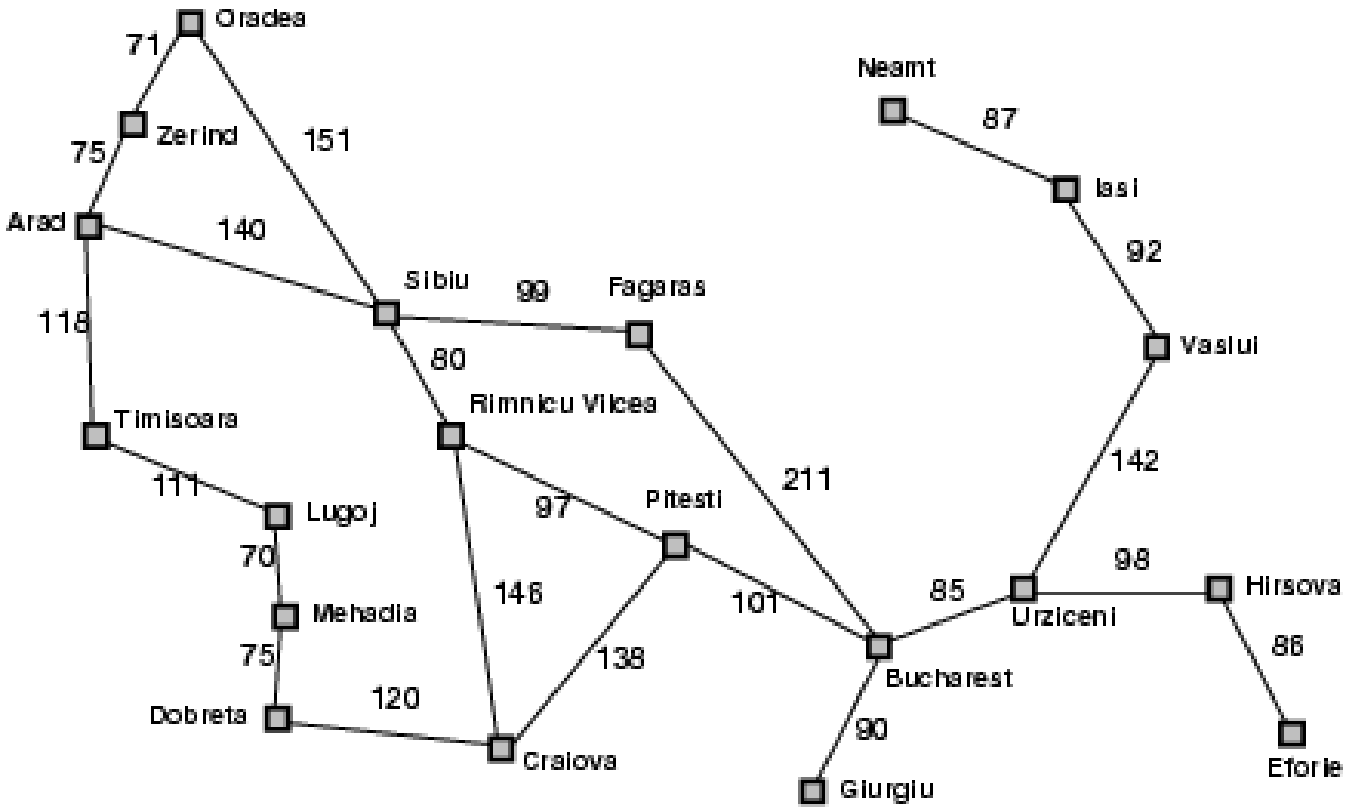
- A search strategy is defined by picking the order of node expansion
- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability“

  → Expand most desirable unexpanded node

- Implementation:
  Order the nodes in fringe in decreasing order of desirability

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

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

Old Friends

- Breadth First =
  - Best First
  - with f(n) = depth(n)

- Uniform cost search =
  - Best First
  - with f(n) = the sum of edge costs from start to n

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

Properties of greedy best-first search

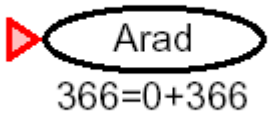
- Complete?
- No – can get stuck in loops, e.g., Iasi → Neamt → Iasi → Neamt →
- Time?
- $O(b^m)$, but a good heuristic can give dramatic improvement
- Space?
- $O(b^m)$ -- keeps all nodes in memory
- Optimal?
- No

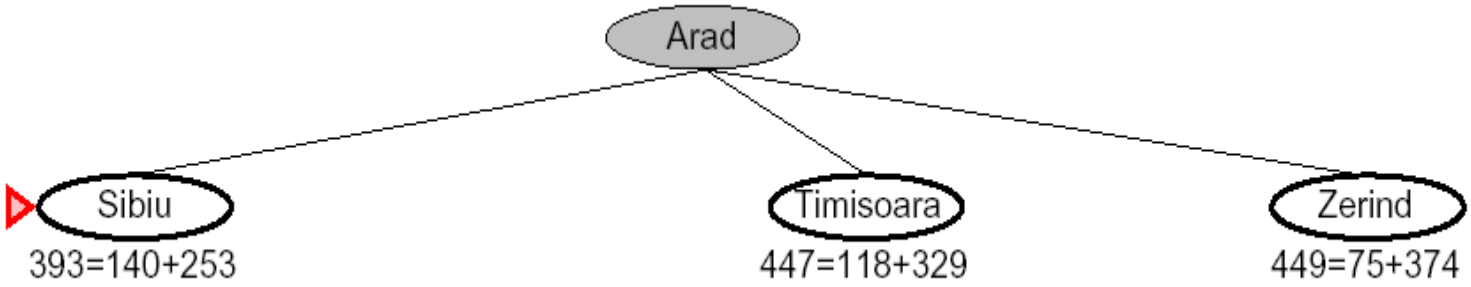
A* search

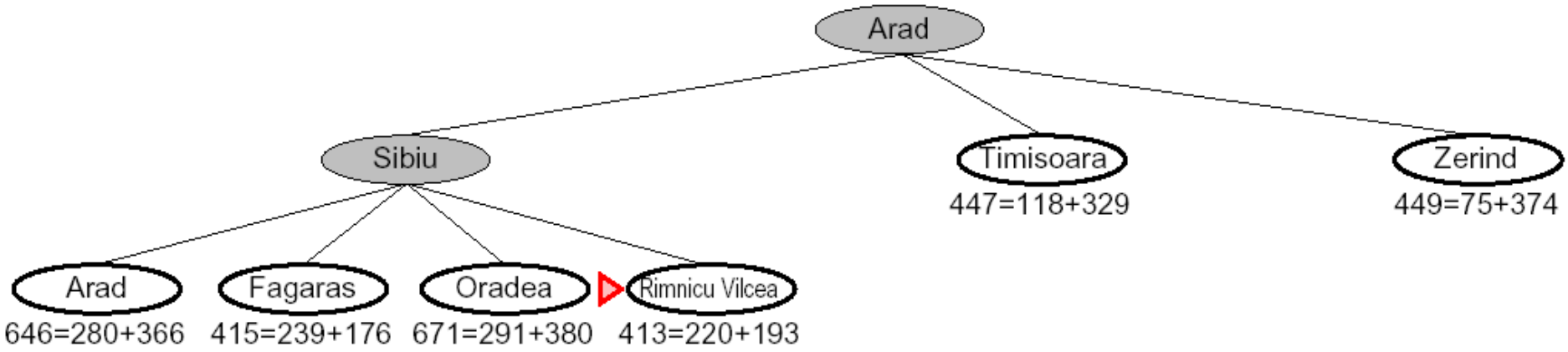
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$

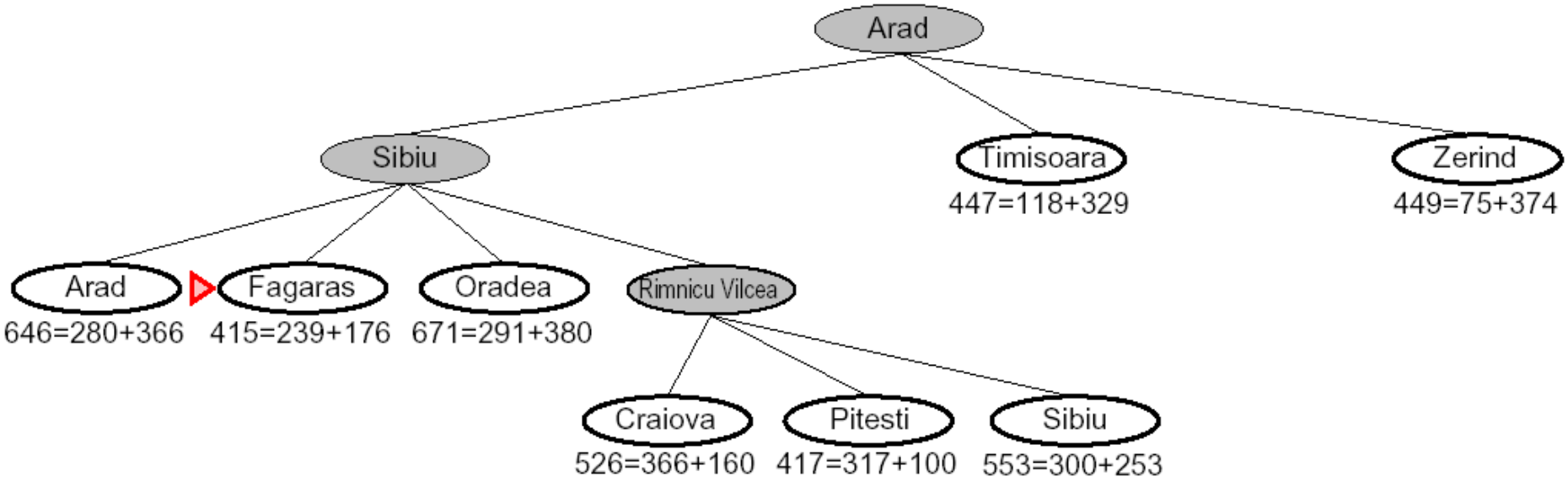
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

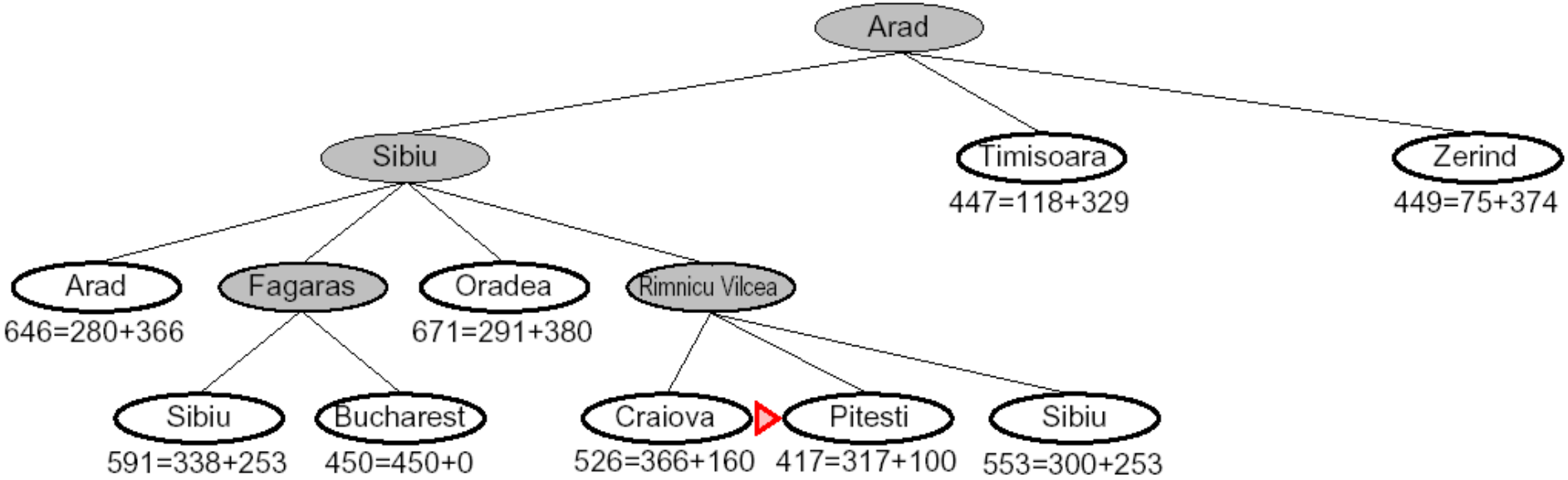
A* for Romanian Shortest Path

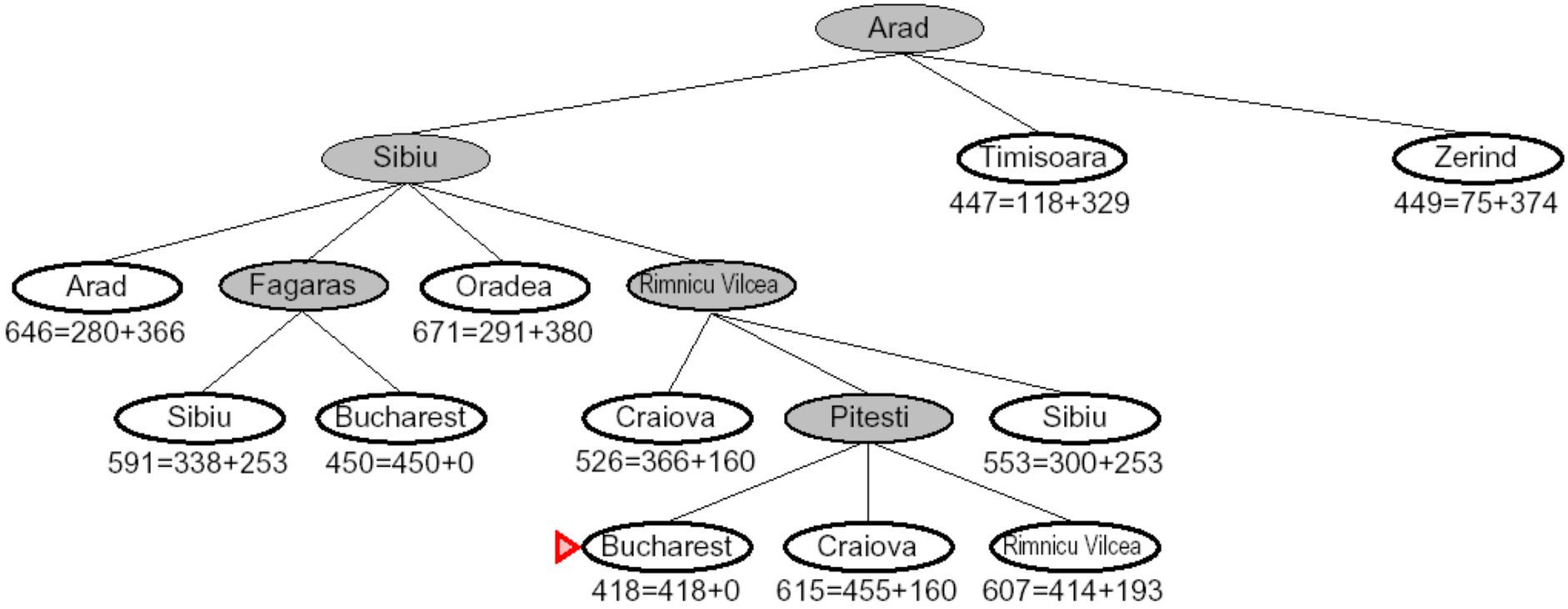



Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

Admissible heuristics

- A heuristic *h(n)* is admissible if for every node *n*,
  $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from *n*.

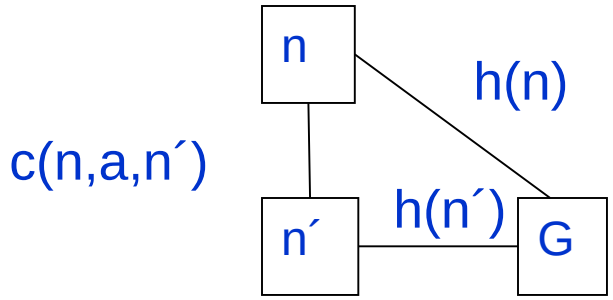
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic

- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

- Theorem: If *h(n)* is admissible, A* using `TREE-SEARCH` is optimal

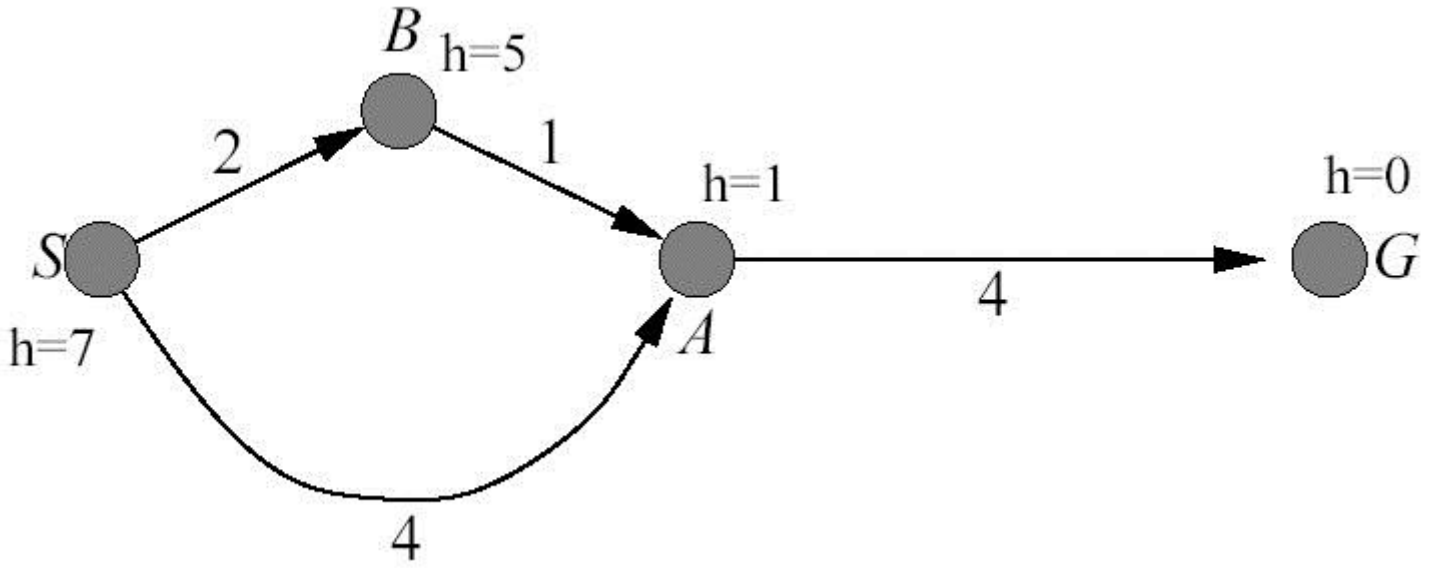
Consistent Heuristics

- h(n) is consistent if
  - for every node n
  - for every successor n´ due to legal action a
  - h(n) <= c(n,a,n´) + h(n´)



- Every consistent heuristic is also admissible.
- Theorem: If *h(n)* is consistent, A* using GRAPH–SEARCH is optimal

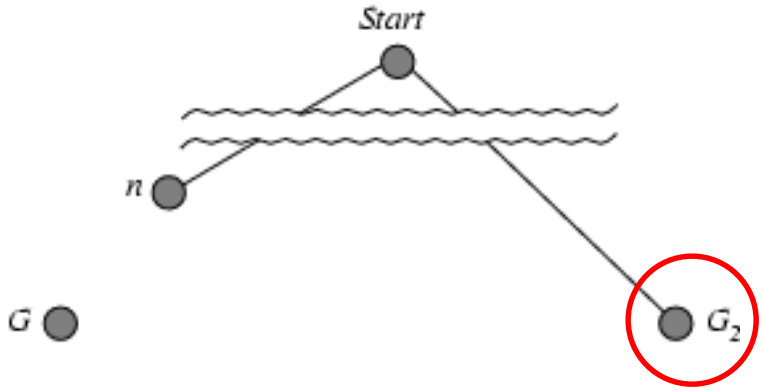
Example



Source: http://stackoverflow.com/questions/25823391/suboptimal-solution-given-by-a-search

Proof of Optimality of (Tree)A*

- Suppose some sub-optimal goal state G_2 has been generated and is on the frontier. Let *n* be an unexpanded state on the agenda such that *n* is on a shortest (optimal) path to the optimal goal state *G*.
  Assume *h*() is admissible.



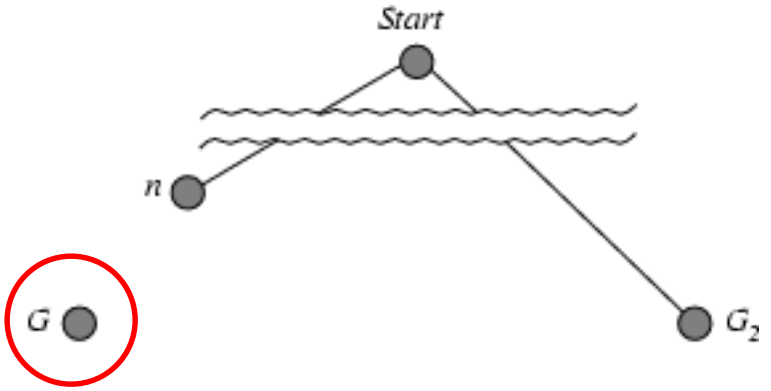
Focus on G_2:

$f(G_2) = g(G_2)$ since $h(G_2) = 0$

$g(G_2) > g(G)$ since G_2 is suboptimal

Proof of Optimality of (Tree)A*

- Suppose some sub-optimal goal state G_2 has been generated and is on the frontier. Let *n* be an unexpanded state on the agenda such that *n* is on a shortest (optimal) path to the optimal goal state *G*.
  Assume *h*() is admissible.



$f(G_2) = g(G_2)$ since $h(G_2) = 0$

$g(G_2) > g(G)$ since G_2 is suboptimal

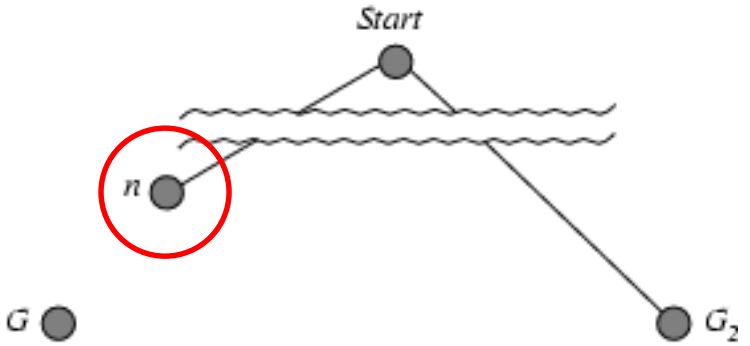
Focus on G:

$f(G) = g(G)$ since $h(G) = 0$

$f(G_2) > f(G)$ substitution

Proof of Optimality of (Tree)A*

- Suppose some sub-optimal goal state G_2 has been generated and is on the frontier. Let *n* be an unexpanded state on the agenda such that *n* is on a shortest (optimal) path to the optimal goal state *G*.
  Assume *h*() is admissible.



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

Now focus on n:

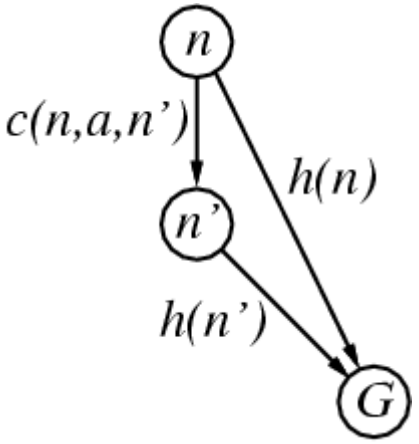
$h(n) \leq h^*(n)$	since h is admissible
$g(n) + h(n) \leq g(n) + h^*(n)$	algebra
$f(n) = g(n) + h(n)$	definition
$f(G) = g(n) + h^*(n)$	by assumption
$f(n) \leq f(G)$	substitution

Hence $f(G_2) > f(n)$, and A^* will never select G_2 for expansion.

Proof of Optimality of (Graph)A*

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

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



- If *h* is consistent, we have

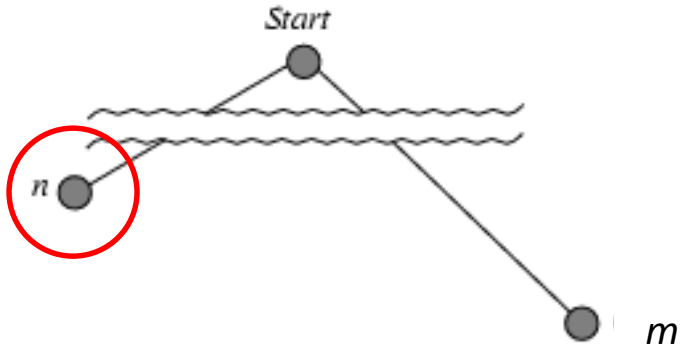
$$
\begin{aligned}
f(n') &= g(n') + h(n') \\
&= g(n) + c(n,a,n') + h(n') \\
&\geq g(n) + h(n) \\
&= f(n)
\end{aligned}
$$

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

Proof of Optimality of (Graph)A*

- Theorem: Whenever A* selects a node m for expansion,
  - the optimal path to m is found

- Assume NOT
- There exists a node n on frontier
  on optimal path s to m



- f(n) < f(m) because of non-decreasing property

Hence A* will never select m for expansion.

Properties of A*

- Complete?

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

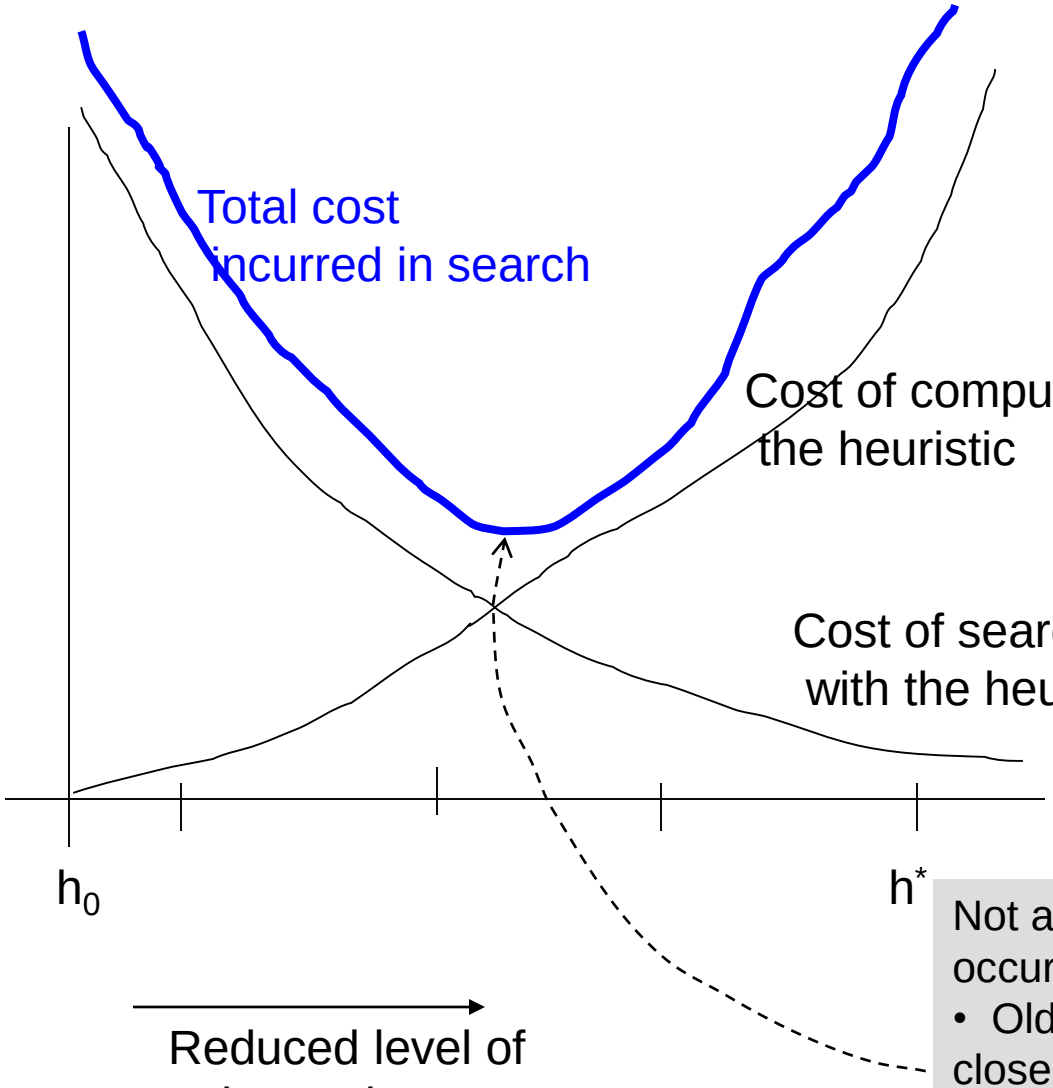
- Time? Exponential

- Space? Keeps all nodes in memory

- Optimal?

  Yes (depending upon search algo and heuristic property)

http://www.youtube.com/watch?v=huJEgJ82360



How informed should the heuristic be?

Not always clear where the total minimum occurs

- Old wisdom was that the global min was closer to cheaper heuristics
- Current insights are that it may well be far from the cheaper heuristics for many problems
  - E.g. Pattern databases for 8-puzzle
  - Plan graph heuristics for planning

Memory Problem?

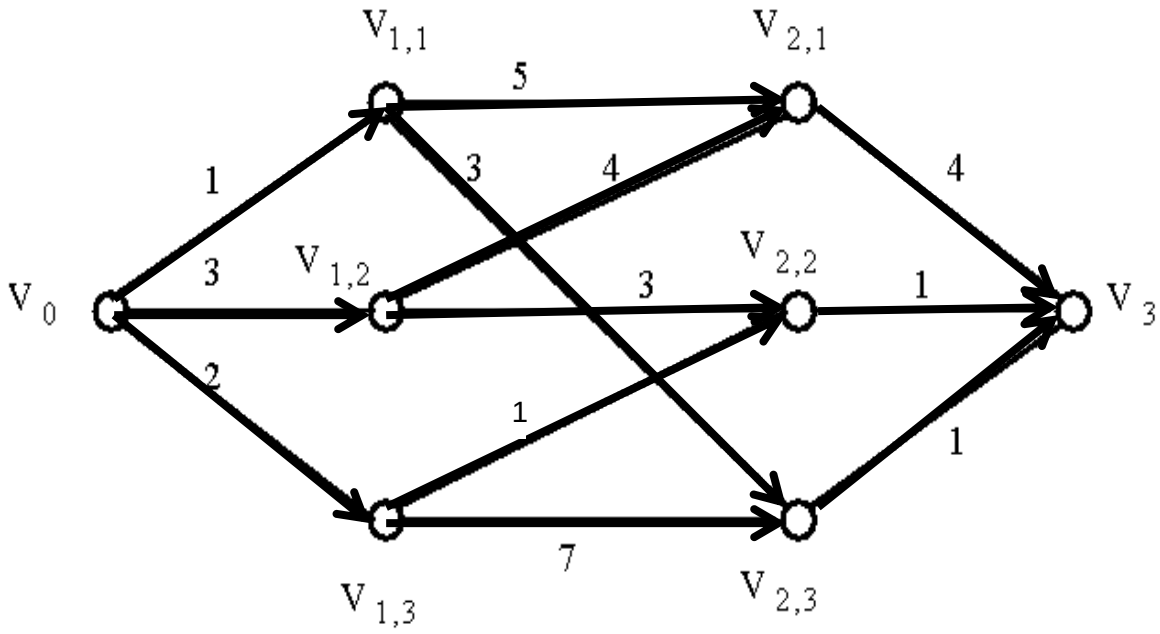
- Iterative deepening A*
  - Similar to ID search
  - While (solution not found)
    - Do DFS but prune when cost (f) > current bound
    - Increase bound

Depth First Branch and Bound

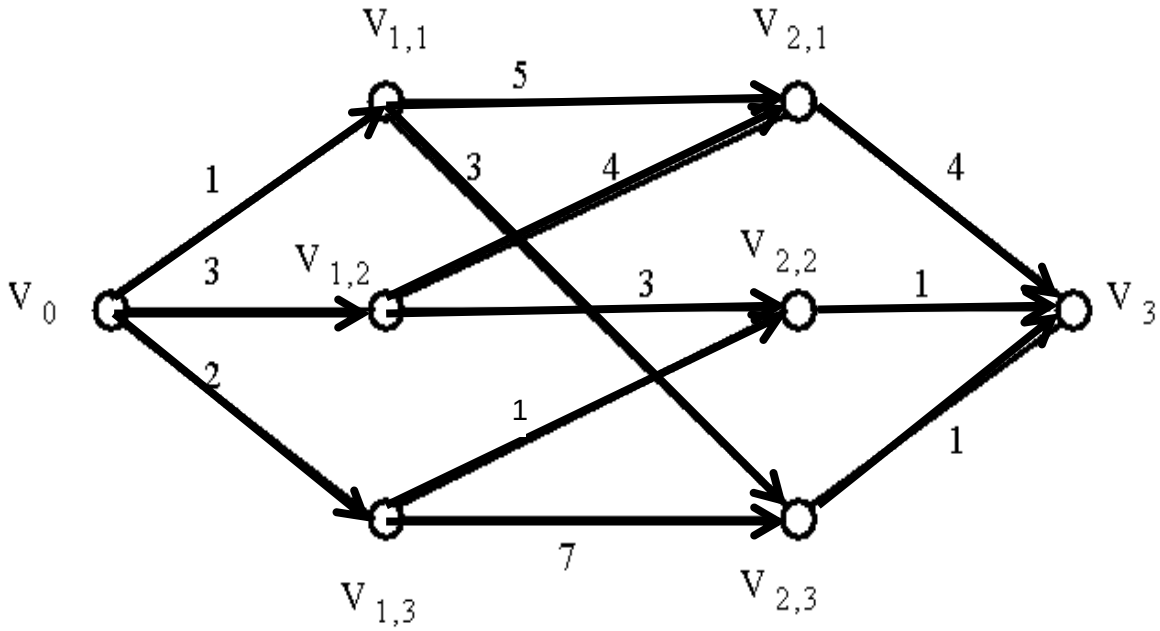
- 2 mechanisms:
  - BRANCH: A mechanism to generate branches when searching the solution space
    - Heuristic strategy for picking which one to try first.
  - BOUND: A mechanism to generate a bound so that many branches can be terminated

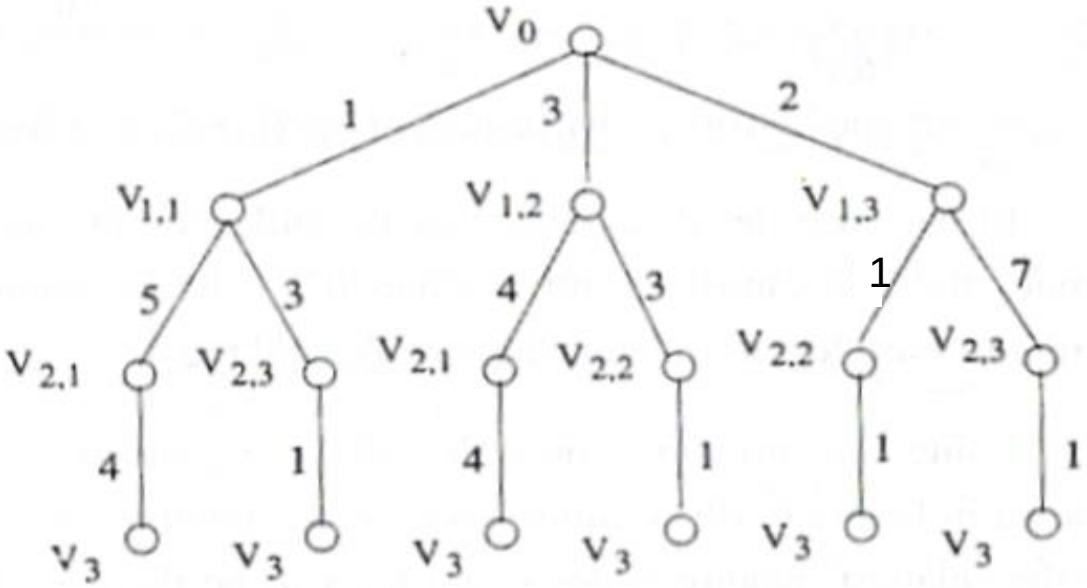
A Multi-Stage Graph Searching Problem.

Find the shortest path from V_0 to V_3

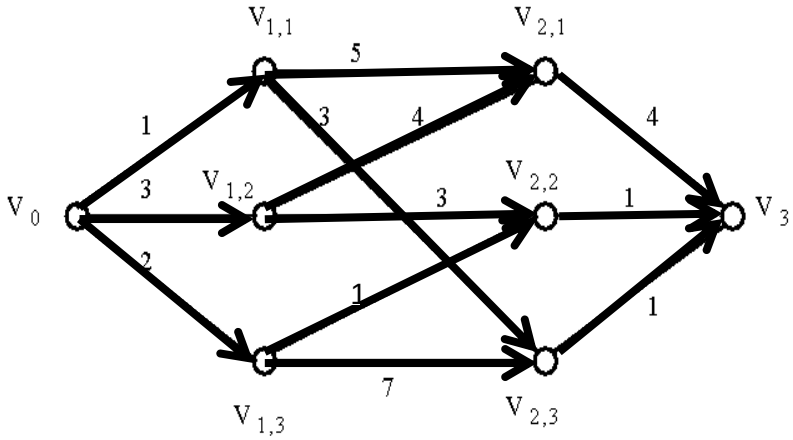


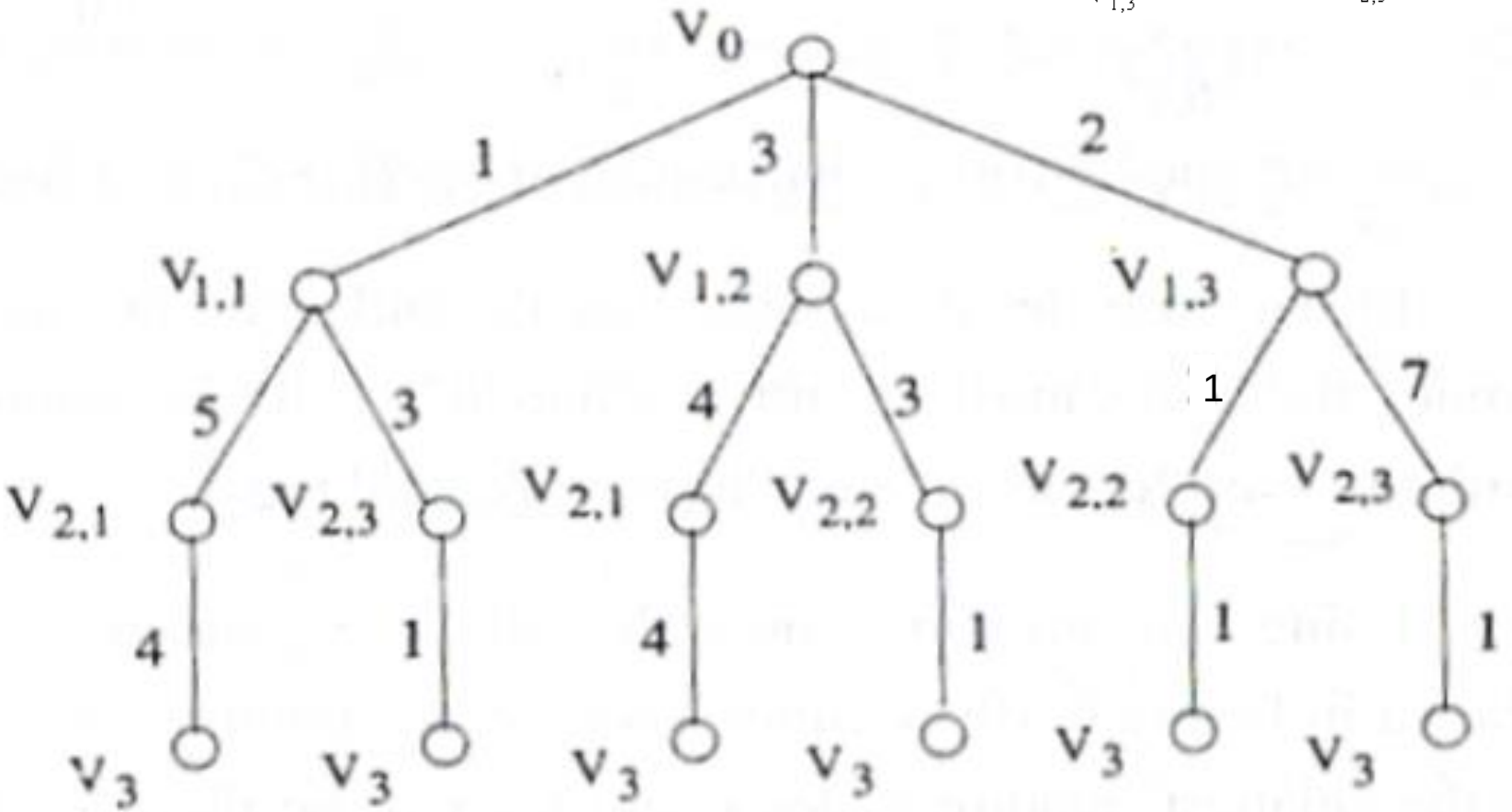
E.G.:A Multi-Stage Graph Searching Problem



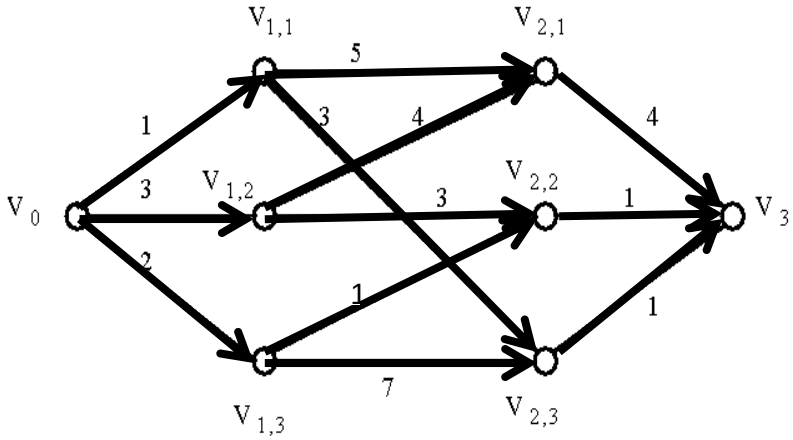


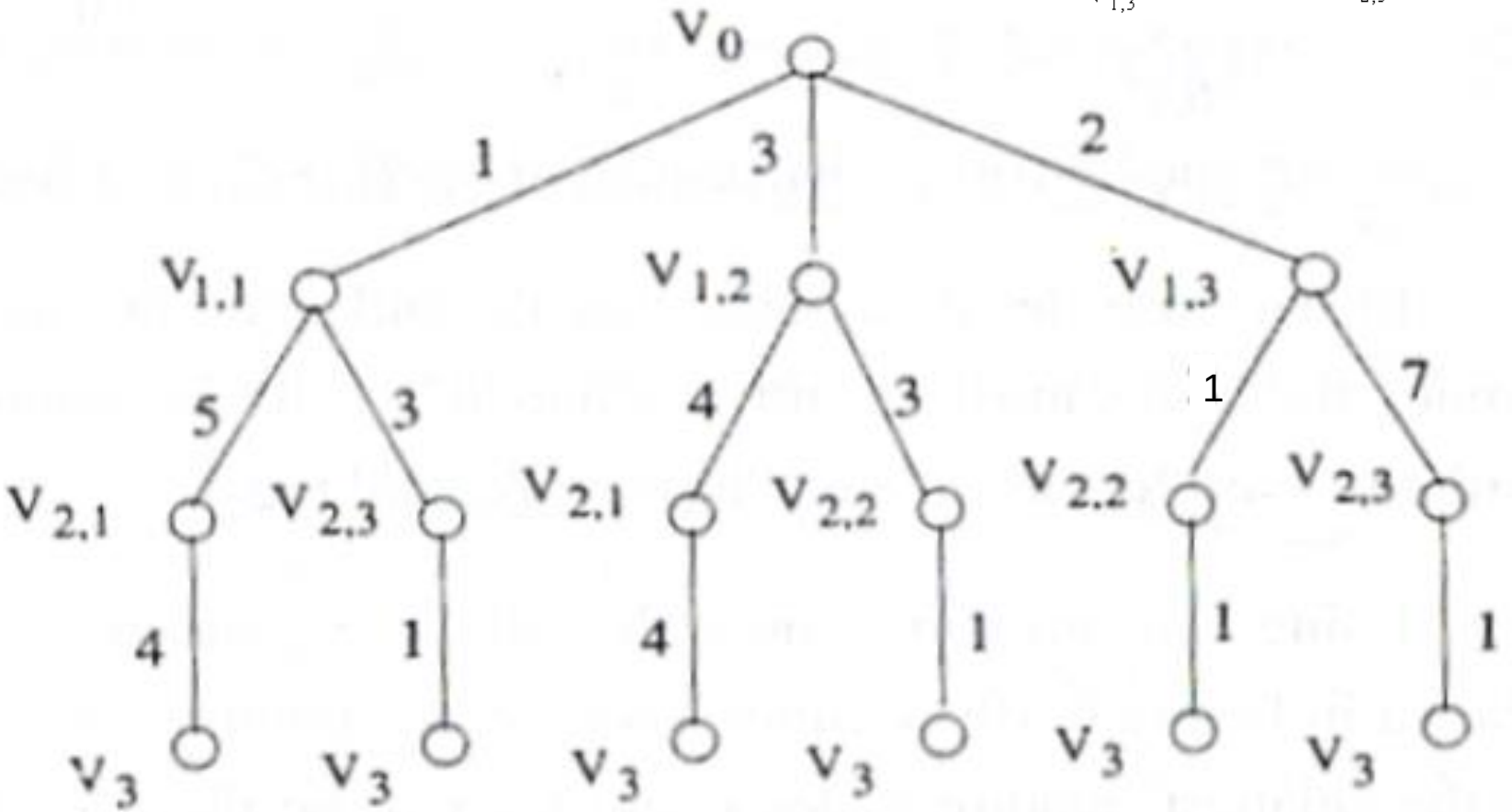
Dfs-B&B


V1,1
V2,1
5
1
3
4
4
V0
3
V1,2
V2,2
3
1
V3
2
1
1
7
V1,3
V2,3

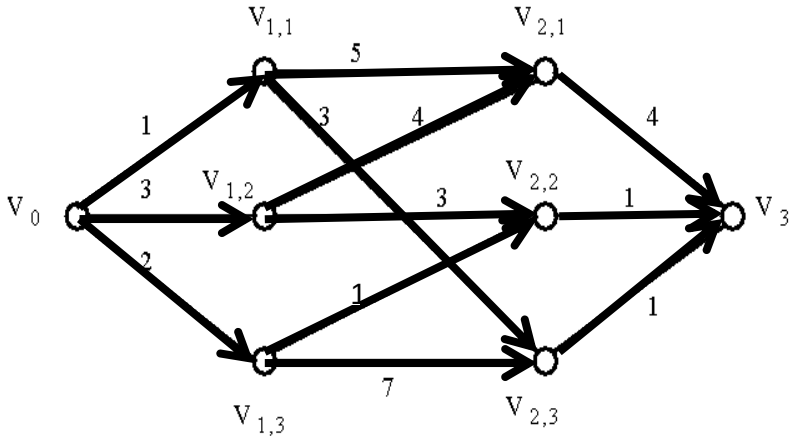

V0
1
3
2
V1,1
V1,2
V1,3
5
3
4
3
1
7
V2,1
V2,3
V2,1
V2,2
V2,2
V2,3
4
1
4
1
1
1
V3
V3
V3
V3
V3
V3

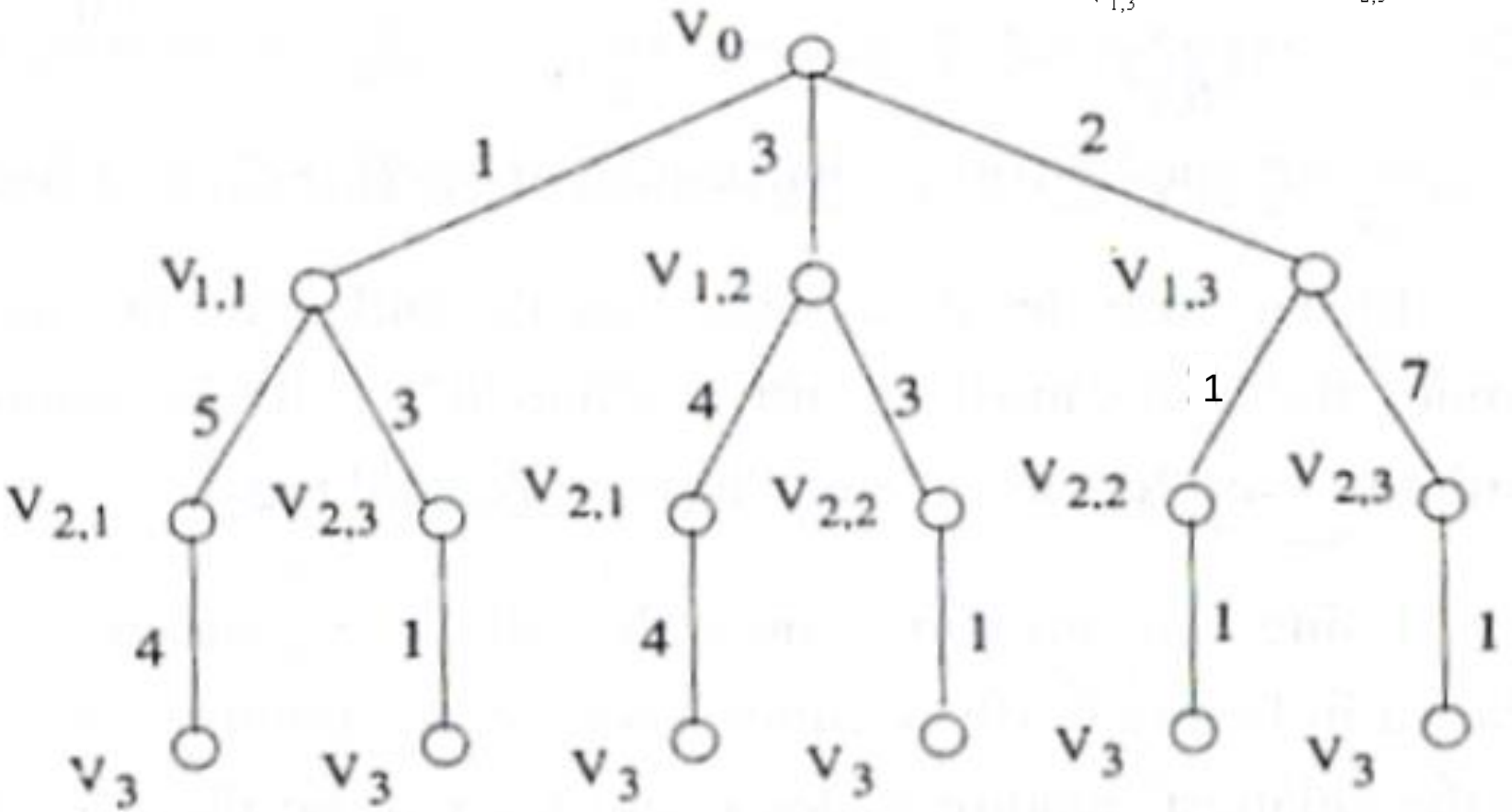
Dfs-B&B


V1,1
V2,1
5
1
3
4
4
V0
3
V1,2
V2,2
3
1
V3
2
1
1
7
V1,3
V2,3

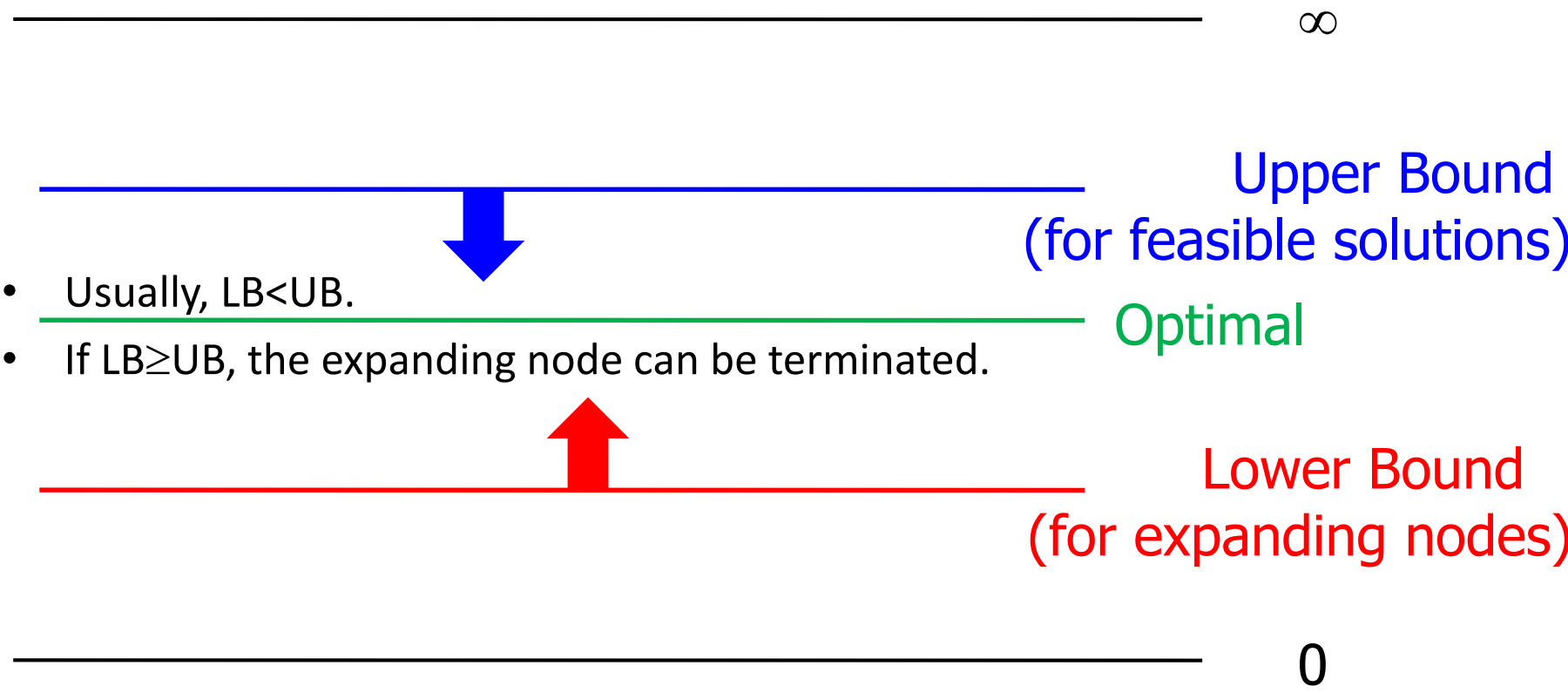

V0
1
3
2
V1,1
V1,2
V1,3
5
3
4
3
1
7
V2,1
V2,3
V2,1
V2,2
V2,2
V2,3
4
1
4
1
1
1
V3
V3
V3
V3
V3
V3

Dfs-B&B


V1,1
V2,1
5
1
3
4
4
V0
3
V1,2
V2,2
3
1
V3
2
1
1
7
V1,3
V2,3


V0
1
3
2
V1,1
V1,2
V1,3
5
3
4
3
1
7
V2,1
V2,3
V2,1
V2,2
V2,2
V2,3
4
1
4
1
1
1
V3
V3
V3
V3
V3
V3

For Minimization Problems



- Usually, LB<UB.
- If LB≥UB, the expanding node can be terminated.

DFS B&B vs. IDA*

- Both optimal
- IDA* never expands a node with f > optimal cost
  - But not systematic
- DFb&b systematic never expands a node twice
  - But expands suboptimal nodes also
- Search tree of bounded depth?
- Easy to find suboptimal solution?
- Infinite search trees?
- Difficult to construct a single solution?

Non-optimal variations

- Use more informative, but inadmissible heuristics

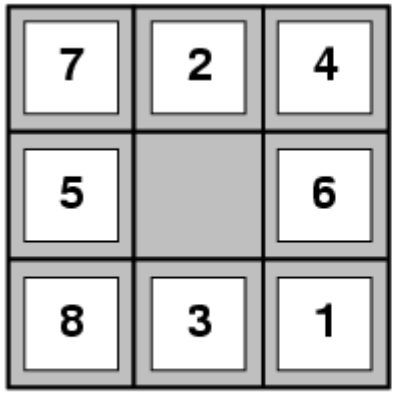
- Weighted A*
  - f(n) = g(n)+ w.h(n) where w>1
  - Typically w=5.
  - Solution quality bounded by w for admissible h

Admissible heuristics

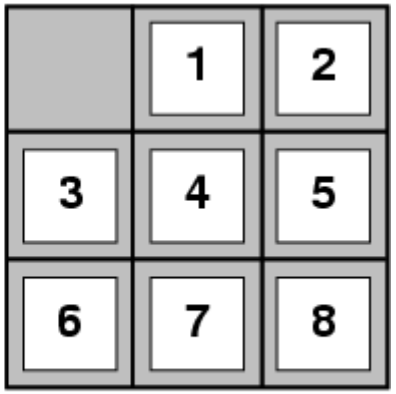
E.g., for the 8-puzzle:

- $h_1(n)$ = 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

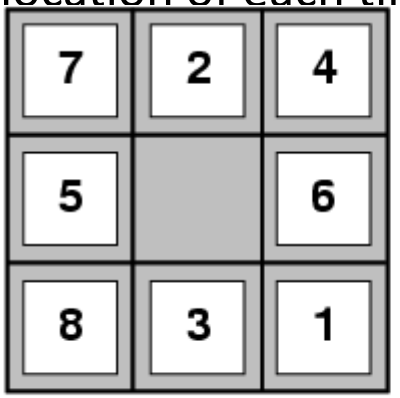
- $h_1(S) = ?$
- $h_2(S) = ?$

Admissible heuristics

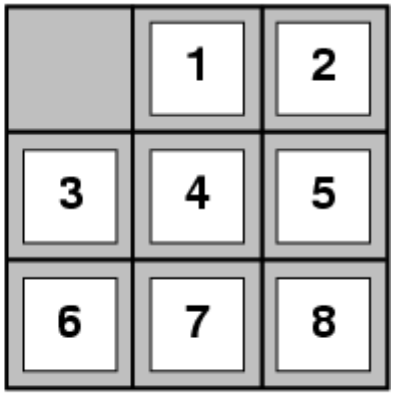
E.g., for the 8-puzzle:

- $h_1(n)$ = 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

- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search

- Typical search costs (average number of node expanded):

- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution

Hamiltonian Cycle Problem

What can be relaxed?

Solution =

1) Each node degree 2
2) Visit all nodes
3) Visit all nodes exactly once

What is a good admissible heuristic for (a1→a2→…→ak)

- length of the cheapest edge leaving ak +
  length of cheapest edge entering a1

- length of shortest path from ak to a1 using other nodes

- length of minimum spanning tree of rest of the nodes