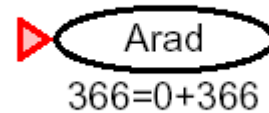
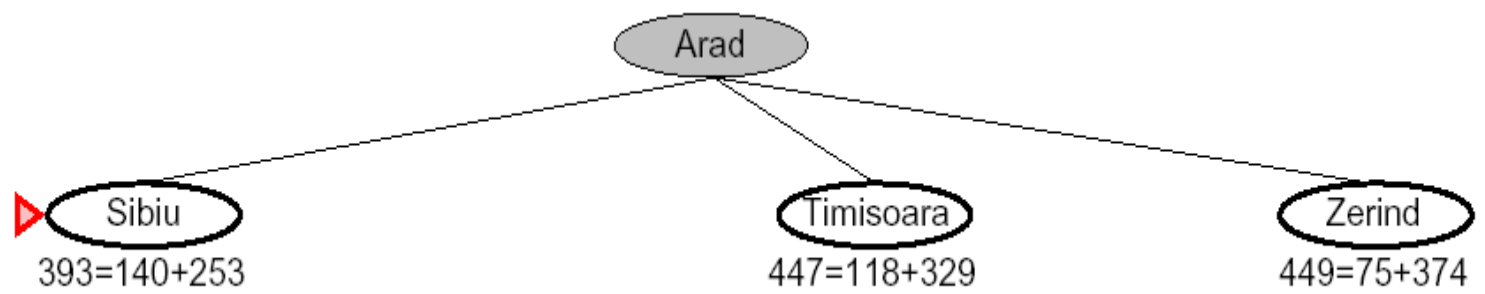


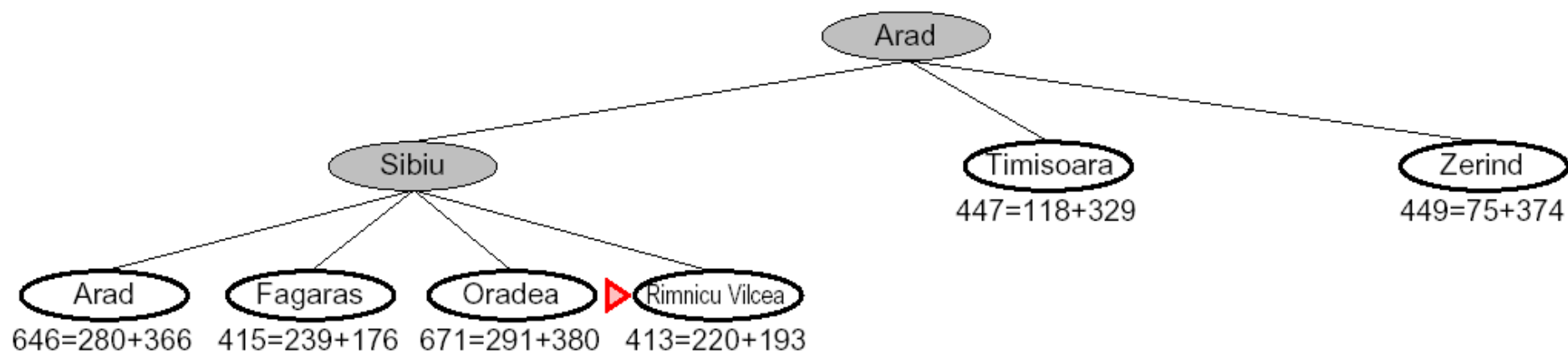
A* search

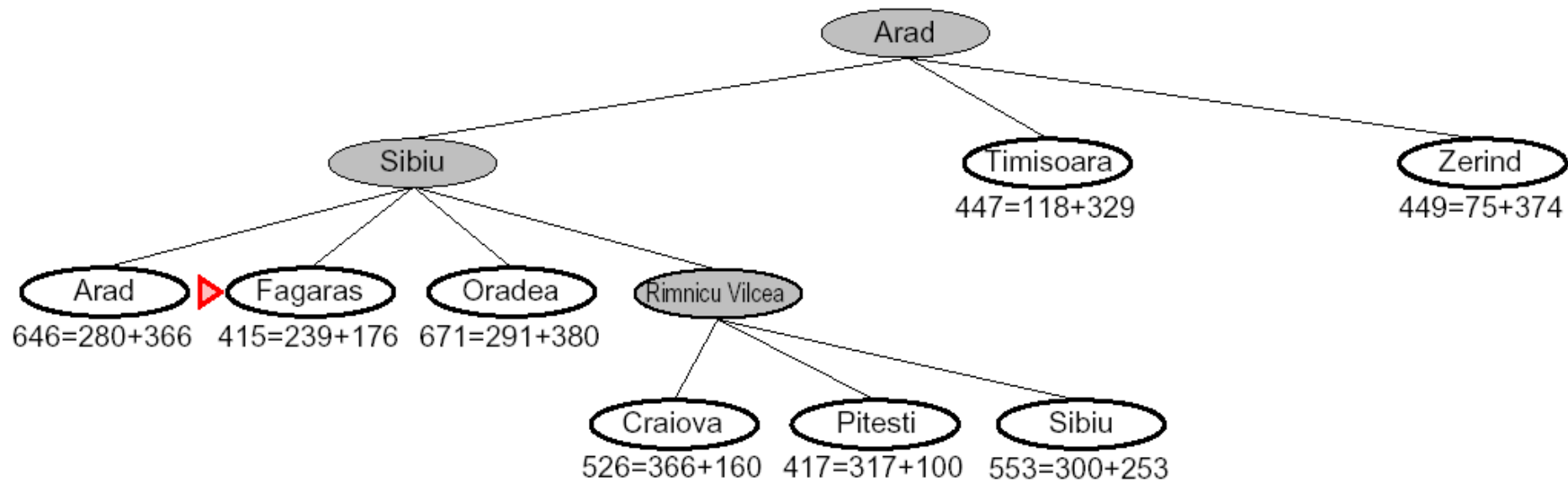
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

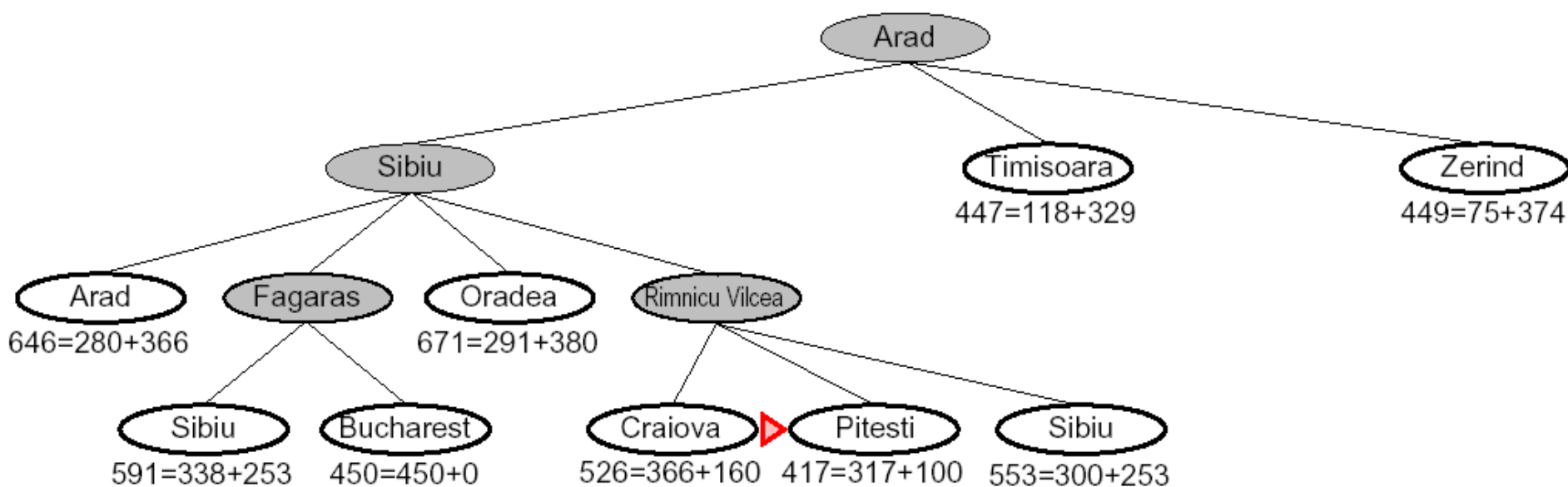
A* for Romanian Shortest Path

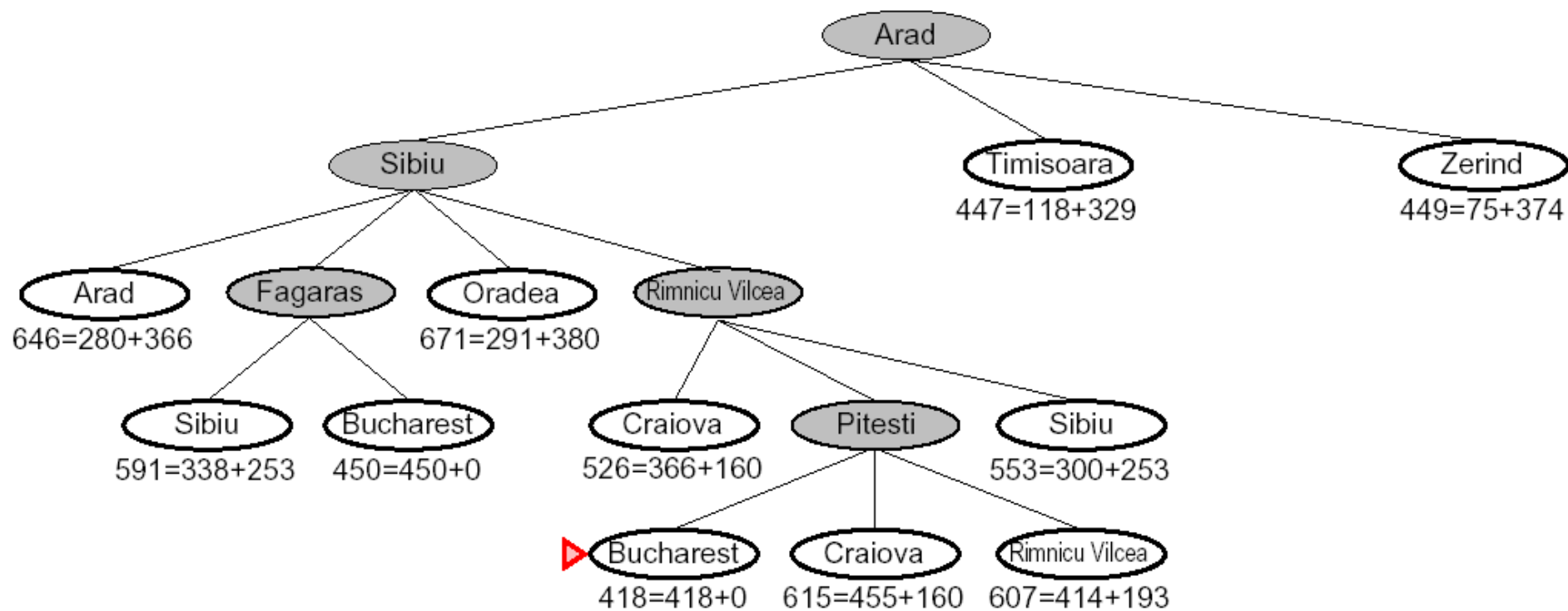










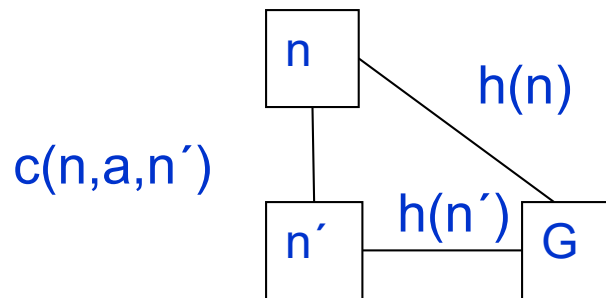


Admissible heuristics

- A heuristic function $h(n)$ 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

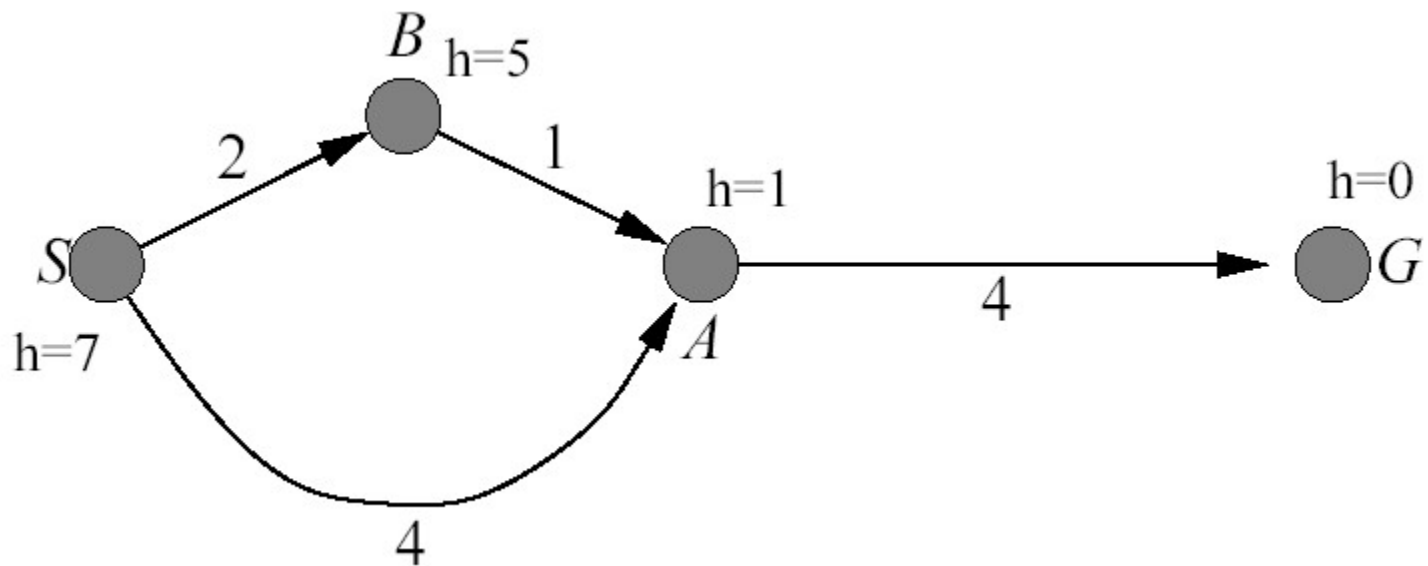
Consistent Heuristics

- $h(n)$ is **consistent** if
 - for every node n
 - for every successor n' due to legal action a
 - $h(n) \leq c(n,a,n') + h(n')$



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

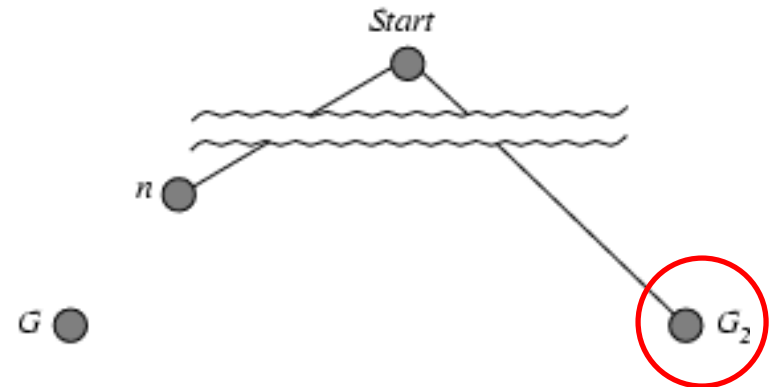
Example



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

Proof of Optimality of (Tree) A^*

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

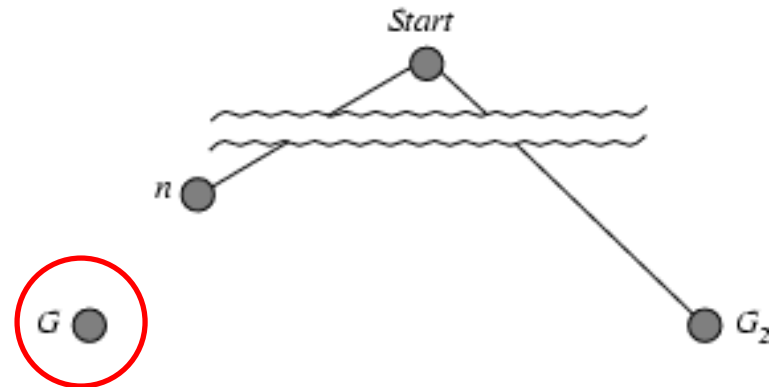


Focus on G_2 :

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

Proof of Optimality of (Tree) A^*

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



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

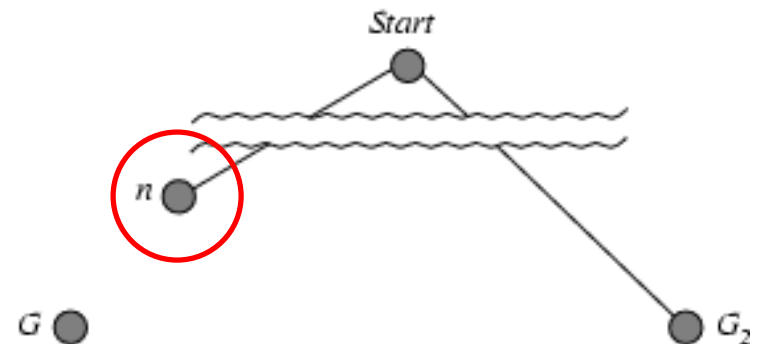
Focus on G :

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

Proof of Optimality of (Tree) A^*

- Assume $h()$ is admissible.

Say some sub-optimal goal state G_2 has been generated and is on the frontier. Let n be an unexpanded state such that n is on an optimal path to the optimal goal G .



$$\begin{array}{ll} f(G_2) = g(G_2) & \text{since } h(G_2) = 0 \\ g(G_2) > g(G) & \text{since } G_2 \text{ is suboptimal} \end{array}$$

$$\begin{array}{ll} f(G) = g(G) & \text{since } h(G) = 0 \\ f(G_2) > f(G) & \text{substitution} \end{array}$$

Now focus on n :

$$\begin{array}{ll} h(n) \leq h^*(n) & \text{since } h \text{ is admissible} \\ g(n) + h(n) \leq g(n) + h^*(n) & \text{algebra} \\ f(n) = g(n) + h(n) & \text{definition} \\ f(G) = g(n) + h^*(n) & \text{by assumption} \\ f(n) \leq f(G) & \text{substitution} \end{array}$$

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

Properties of A*

- Complete?

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

- Time? Exponential (worst case all nodes are added)

- Space? Keeps all nodes in memory

- Optimal?

Yes (depending upon search algo and heuristic property)

A^*



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

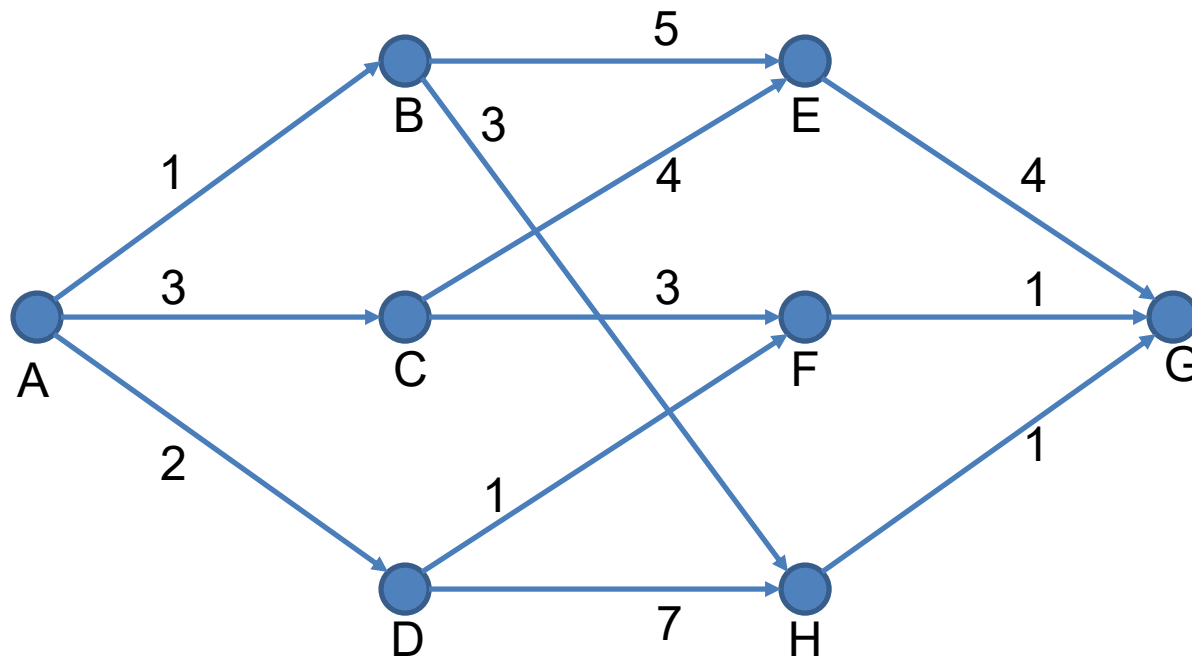
Memory Problem?

- 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

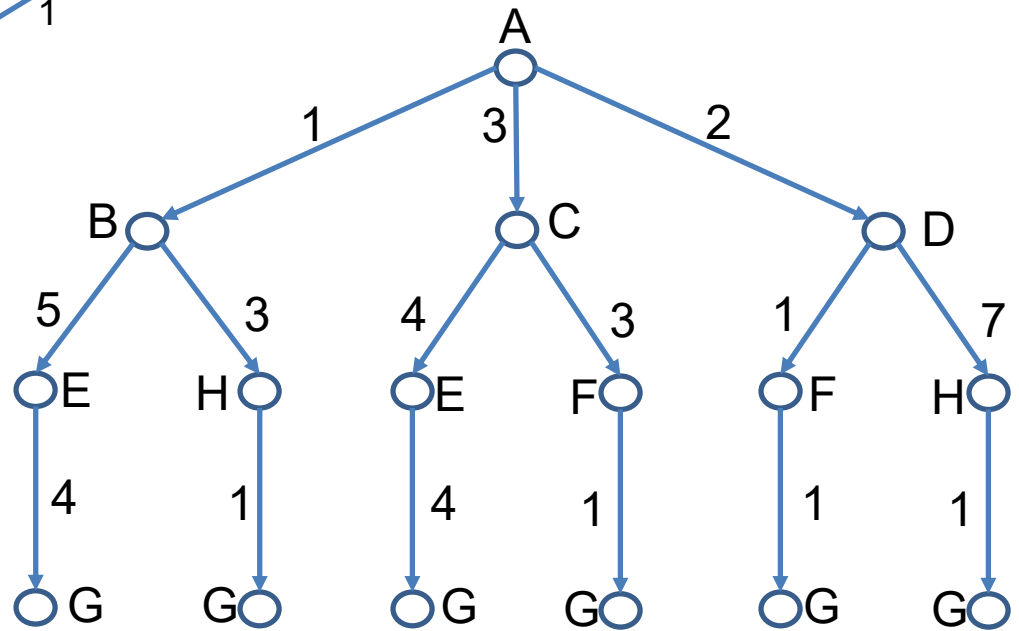
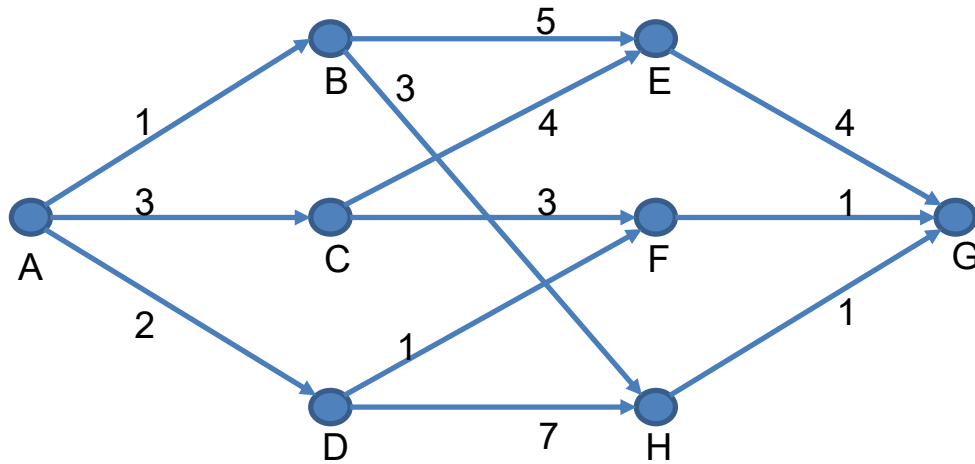
- 2 mechanisms:
 - **BRANCH**: A mechanism to generate branches when searching the solution space
 - Heuristic strategy for picking which one to try first.
 - **BOUND**: A mechanism to generate a bound so that many branches can be terminated

Example



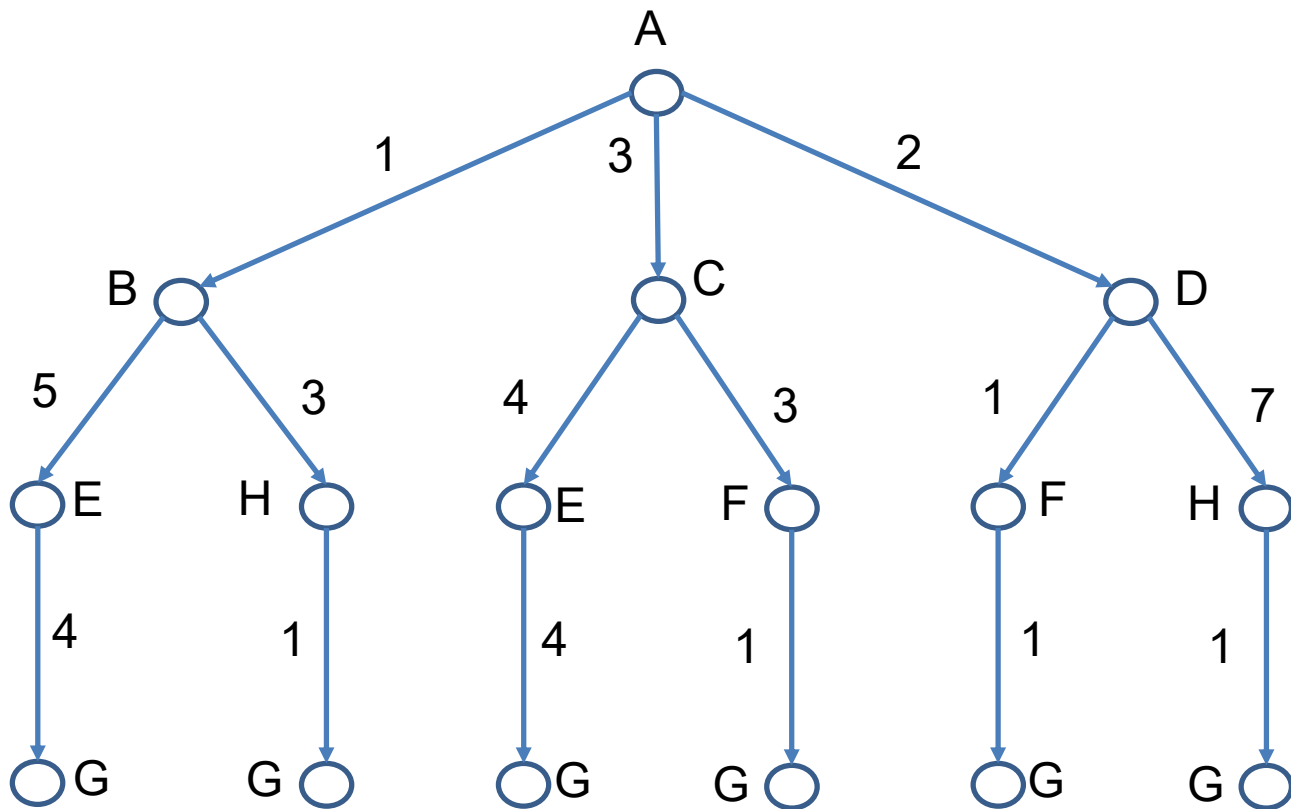
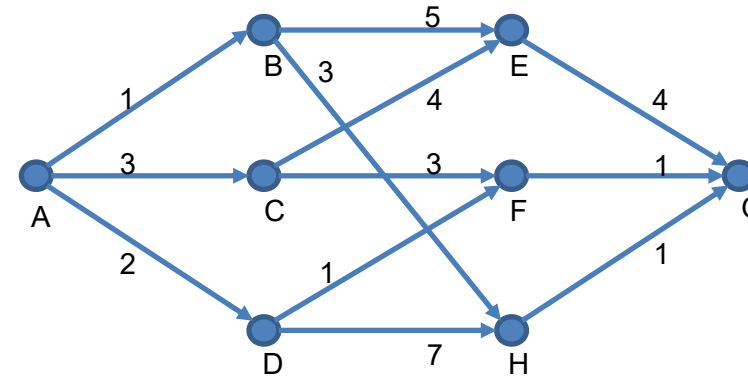
Find optimal path from A to G

Search Tree

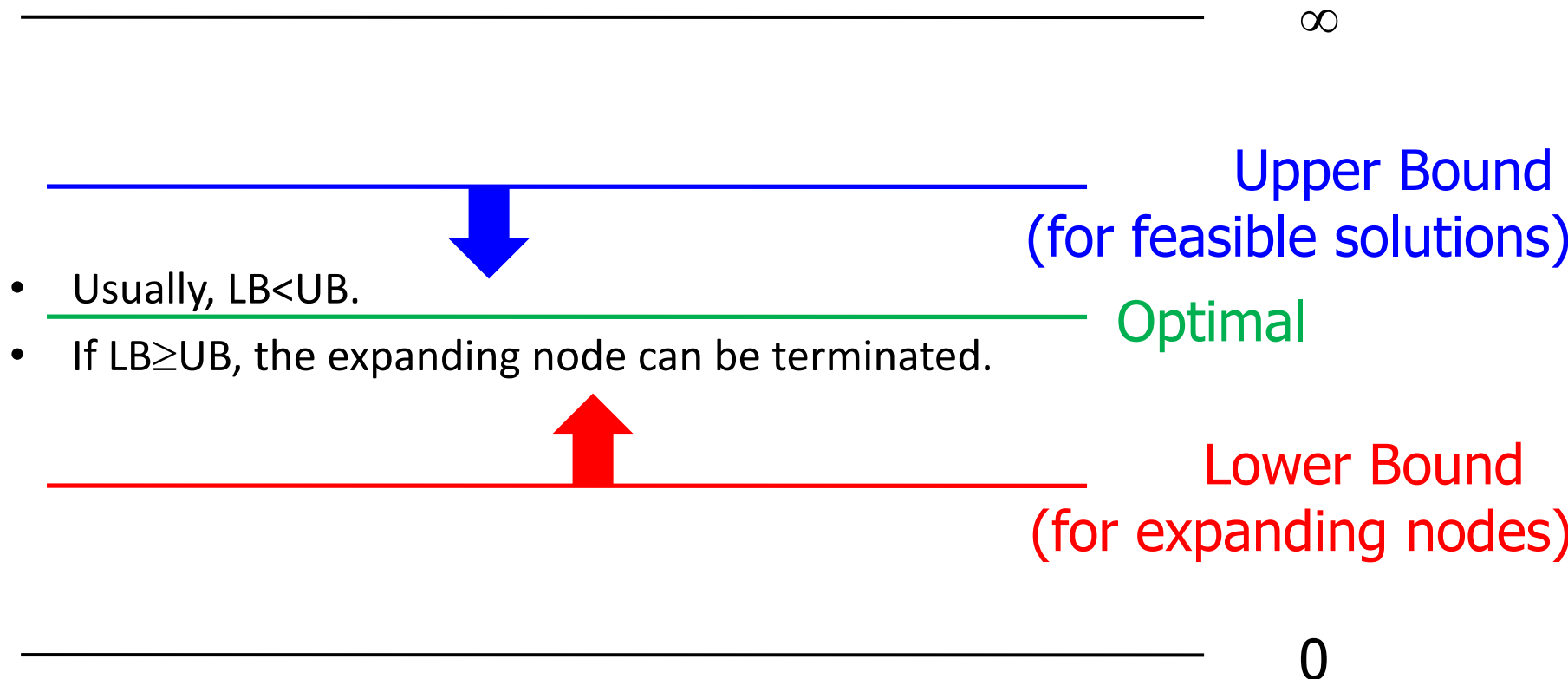


DFS B&B

E.g., Branch policy: take lowest cost edge first



For Minimization Problems



DFS B&B vs. IDA*

- Both optimal
- IDA* never expands a node with $f > \text{optimal cost}$
 - 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^*
 - $f(n) = g(n) + w \cdot h(n)$ 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)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

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)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

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 nodes exactly once

What is a good admissible heuristic for $(a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_k)$

- length of the cheapest edge leaving a_k +
length of cheapest edge entering a_1
- length of shortest path from a_k to a_1
- length of minimum spanning tree of rest of the nodes