Chapter 3

Lecture 3

Informed Search Algorithms (Ch 3.5)

Jim Rehg

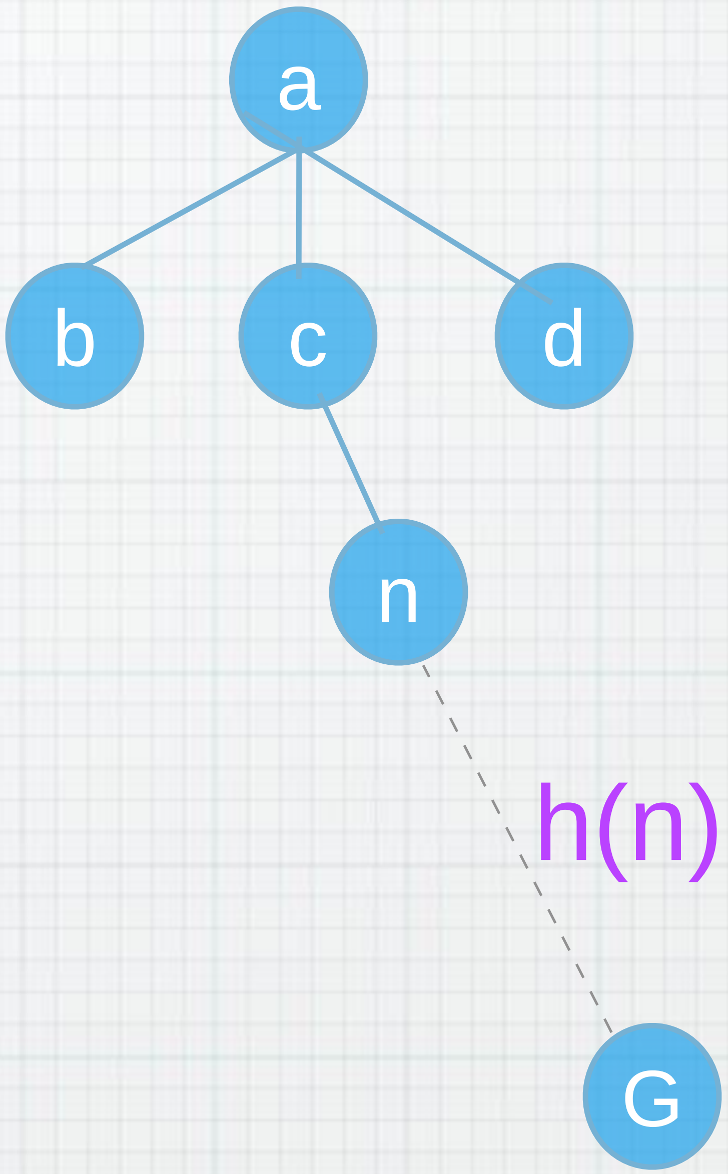
Evaluation Function

* f(n) = desirability of node n
* Best-First Search:
  Tree search + Evaluation Function f(n)

Search Strategy: How to define eval function

Heuristic Function

* Key to BFS algorithms is the heuristic h(n)
    * estimated cheapest path, n to goal
    * estimated future path cost from n



Greedy Best-First

* $f(n) = h(n)$: expand node that appears to be closest from here
* Example — Route planning — a common heuristic is straight line distance to goal

Greedy Best-First

- Complete?
  - No, can get stuck in loops
  - Yes, if graph-search version
- Optimal?
  - No, only pays attention to future not how costly it was to get here

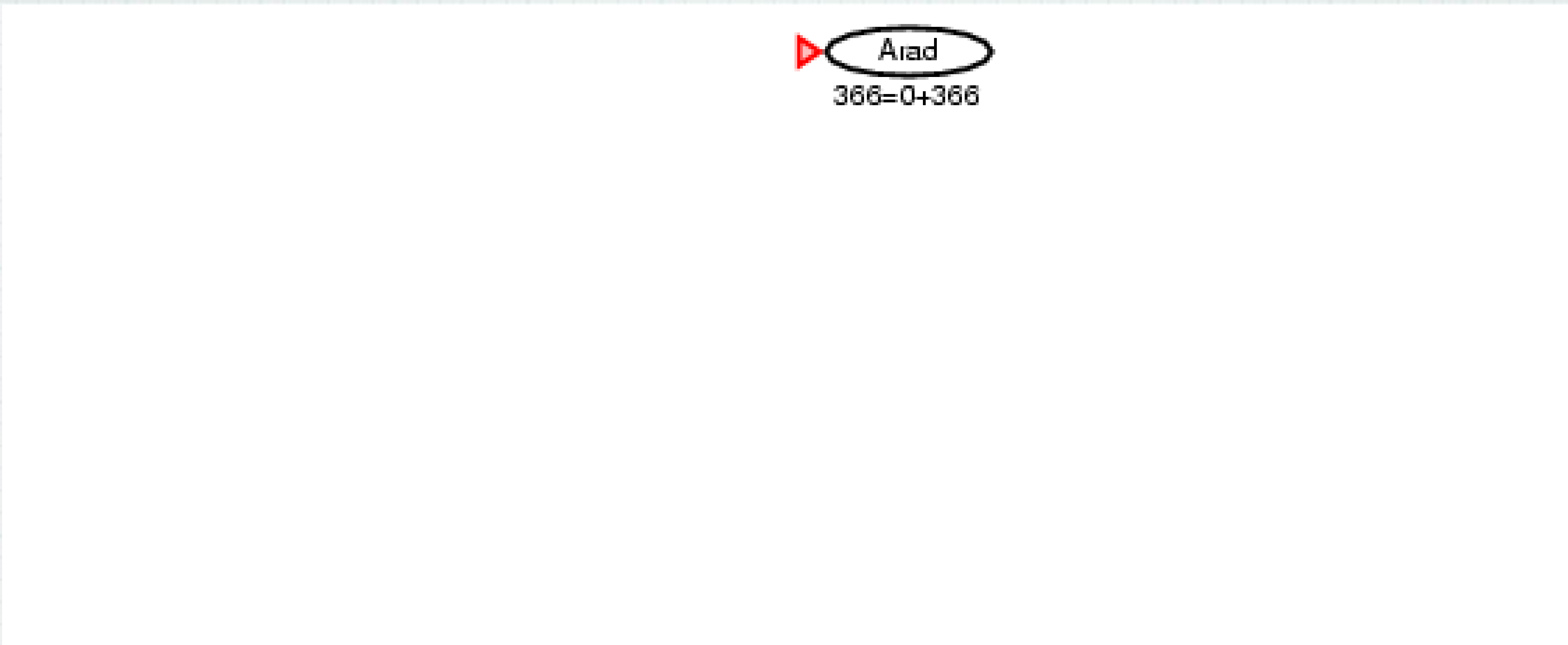
A* Search

* Most widely known Best-First alg
* $f(n) = g(n) + h(n)$
  * $g(n)$ = cost to get to this node
  * $h(n)$ = estimated cost from here
* Minimizes total solution cost
* With an admissible h(n) A* is both complete and optimal

Admissible Heuristic

* h(n) =
    * under-estimate of cost to goal
    * zero for any goal state
    * non-zero for all others
* Makes A* Optimal & Complete!

A* search example

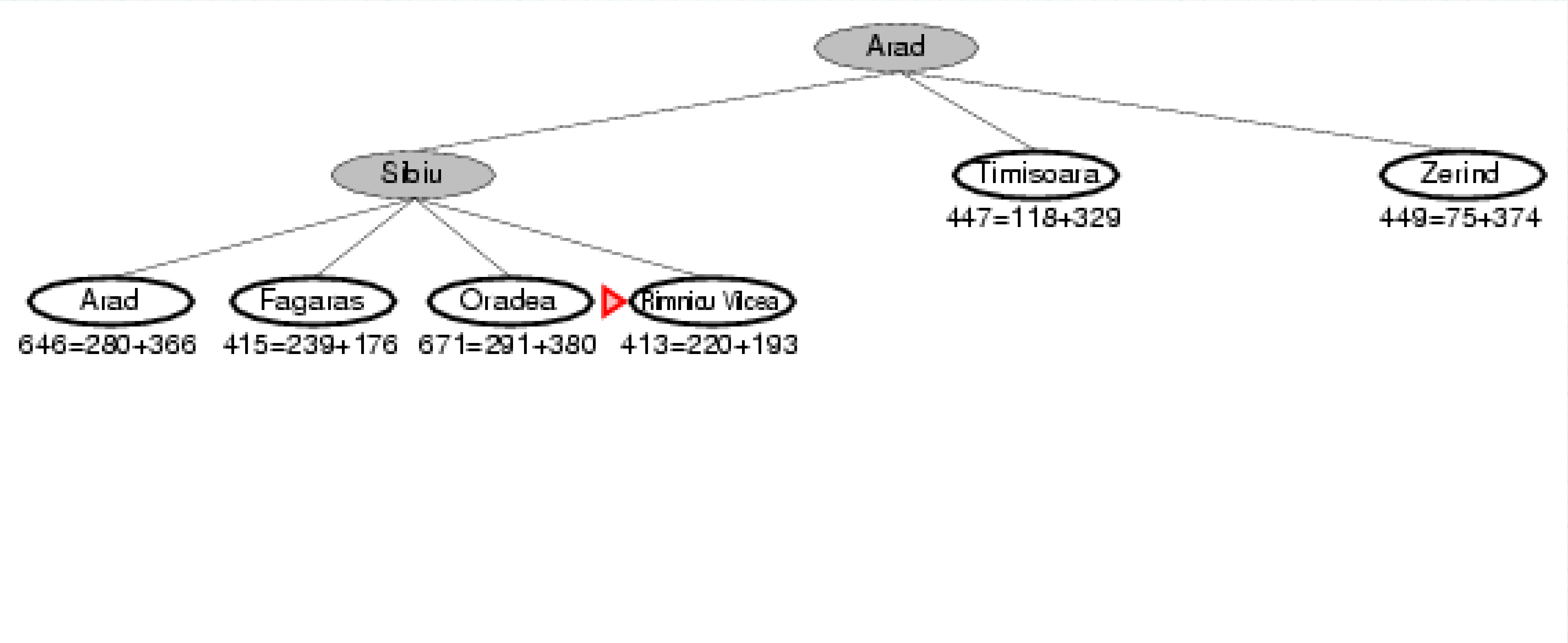


* $f(n) = g(n) + h(n)$

A* search example

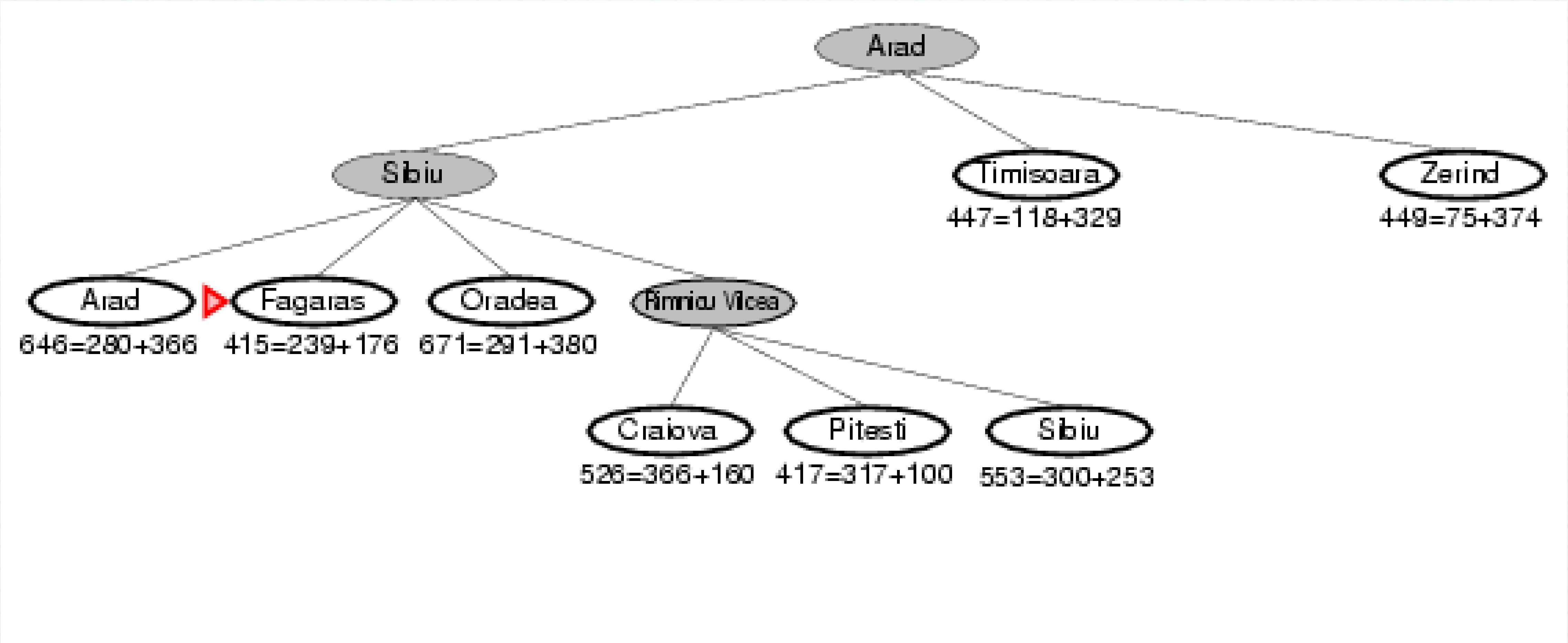
- $f(n) = g(n) + h(n)$

A* search example



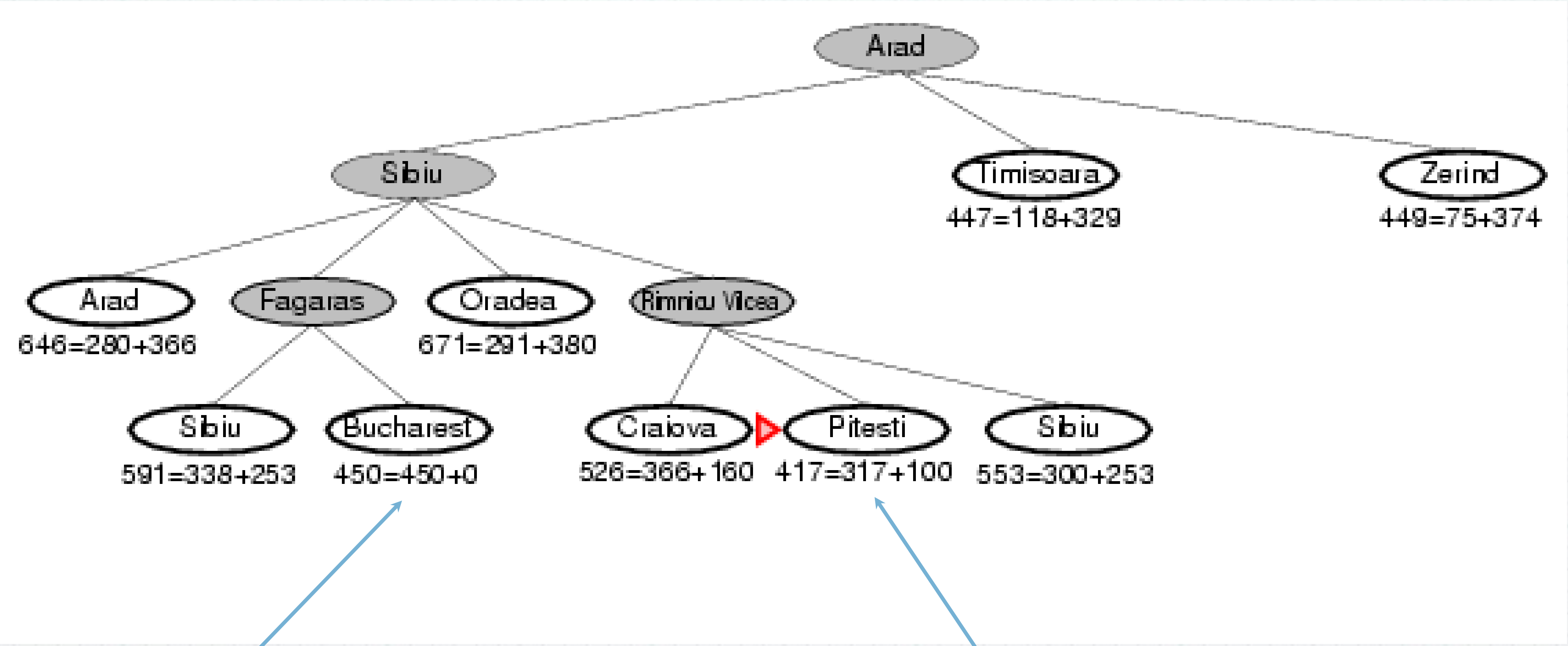
* $f(n) = g(n) + h(n)$

A* search example



- $f(n) = g(n) + h(n)$

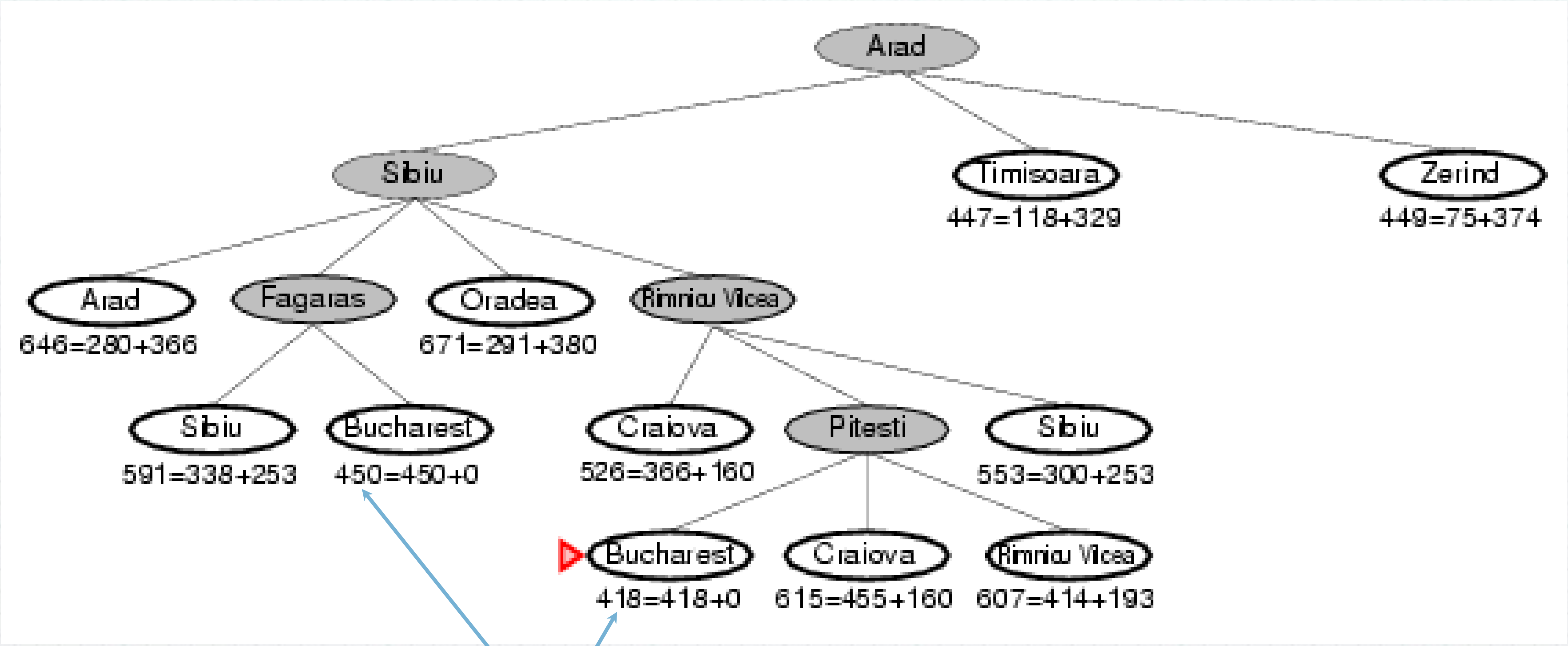
A* search example



Goal appears in the frontier

But we keep searching cheaper paths

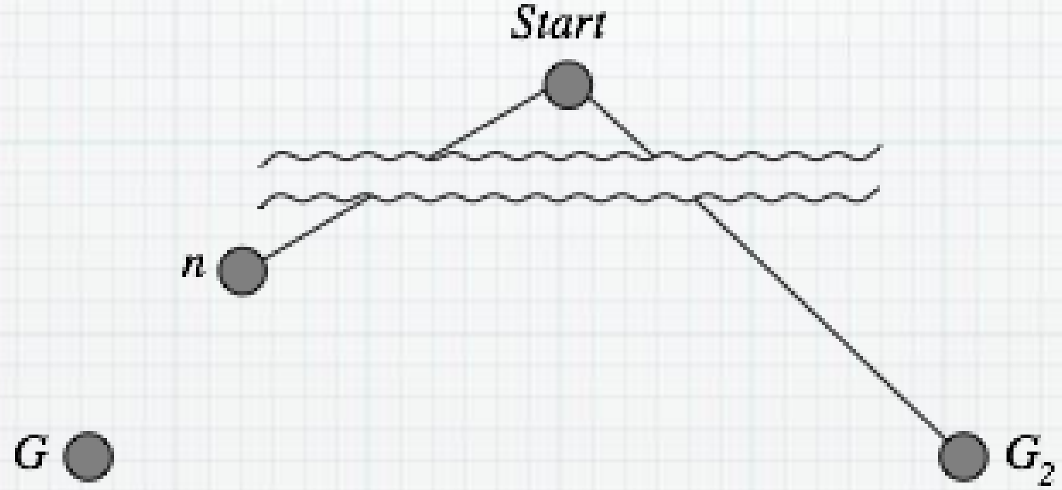
A* search example



And we find the cheapest path instead of first path to the goal...

A* Optimality 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.



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

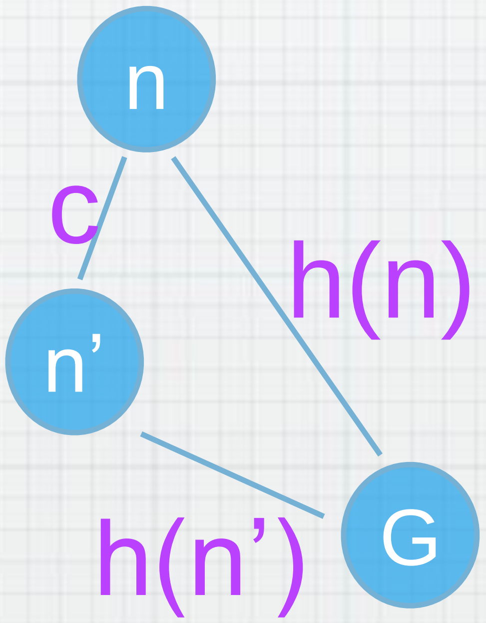
Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

A* + Graph-Search

* Basic A* can have repeated states
* Graph search more efficient
    * Not optimal, may cut off opt. path
    * Fix with extra book keeping to make sure you repeat a state only if it's more optimal

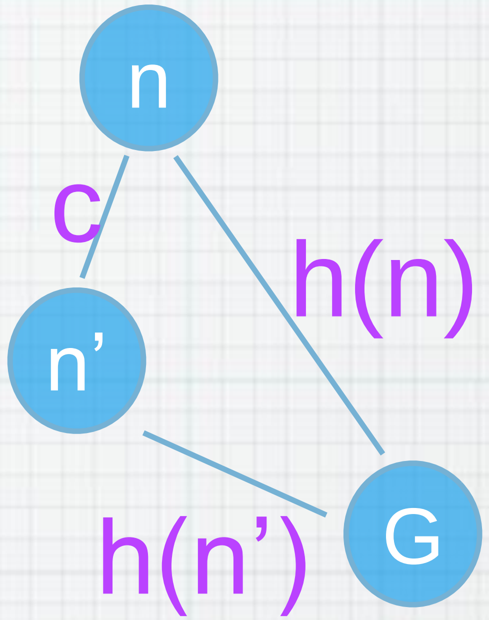
Consistent Heuristics

- An extra requirement on h(n) can make A* graph search optimal
- consistent: $h(n) <= c(n,a,n') + h(n')$



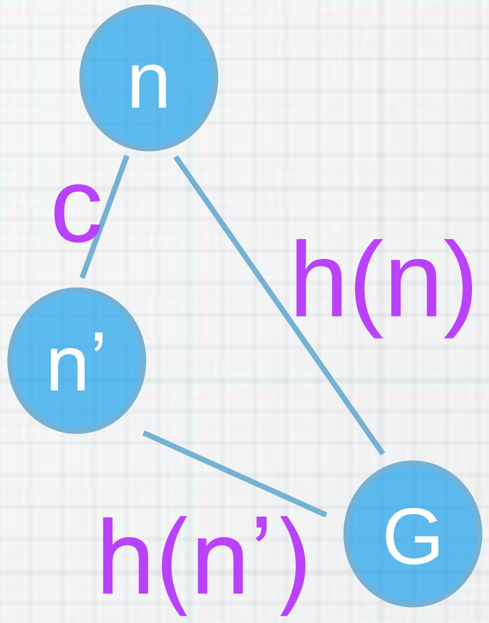
Consistent Heuristics

* h(n) <= c(n,a,n’) + h(n’)



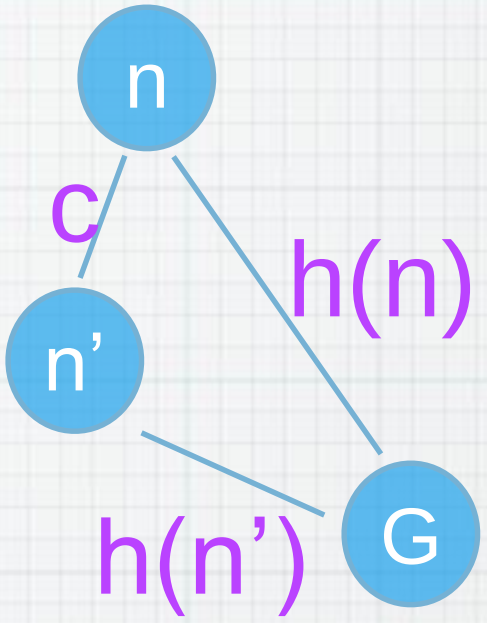
Consistent Heuristics

* $h(n) <= c(n,a,n') + h(n')$
* $f(n') = g(n') + h(n')$

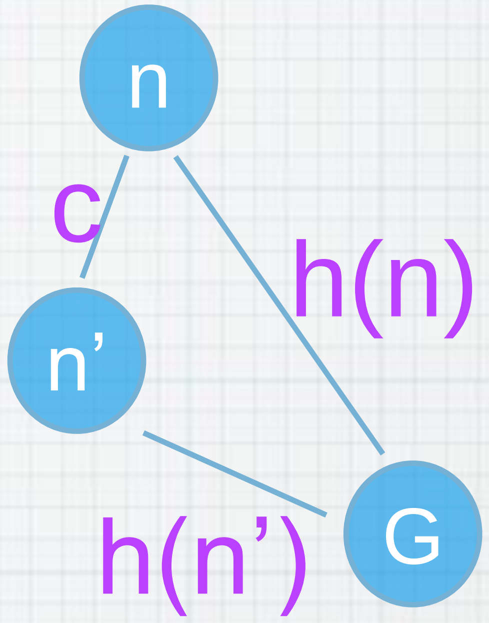


Consistent Heuristics

* $h(n) <= c(n,a,n') + h(n')$
* $f(n') = g(n') + h(n')$



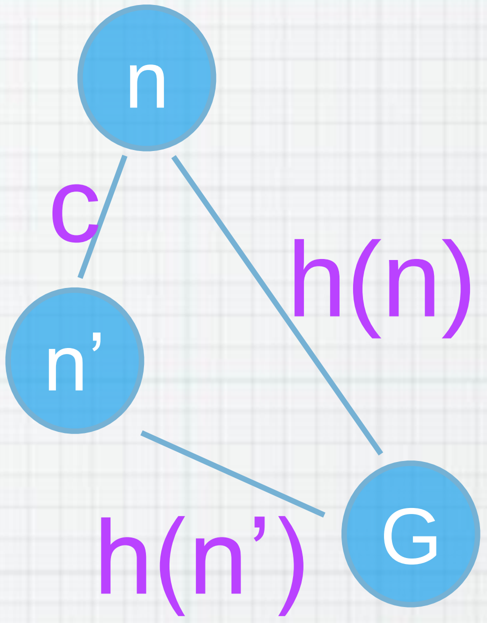
Consistent Heuristics



- h(n) <= c(n,a,n') + h(n')
- f(n') = g(n') + h(n')

  = g(n) + c(n,n') + h(n')

Consistent Heuristics



* $h(n) <= c(n,a,n') + h(n')$
* $f(n') = g(n') + h(n')$

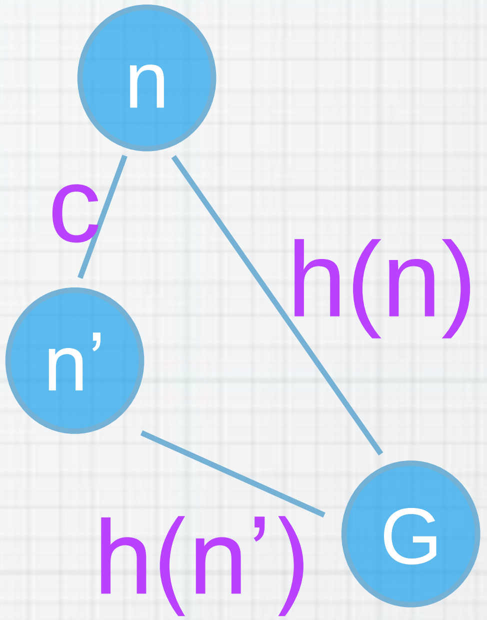
  $= g(n) + c(n,n') + h(n')$

Consistent Heuristics

* h(n) <= c(n,a,n’) + h(n’)
* f(n’) = g(n’) + h(n’)

  = g(n) + c(n,n’) + h(n’)

  >= g(n) + h(n)

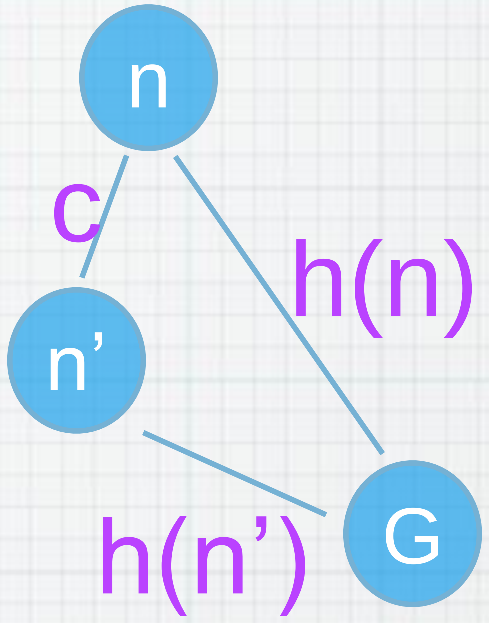


Consistent Heuristics

- $h(n) <= c(n,a,n') + h(n')$
- $f(n') = g(n') + h(n')$

  $= g(n) + c(n,n') + h(n')$

  $>= g(n) + h(n)$
- $f(n') >= f(n)$



Consistent Heuristics

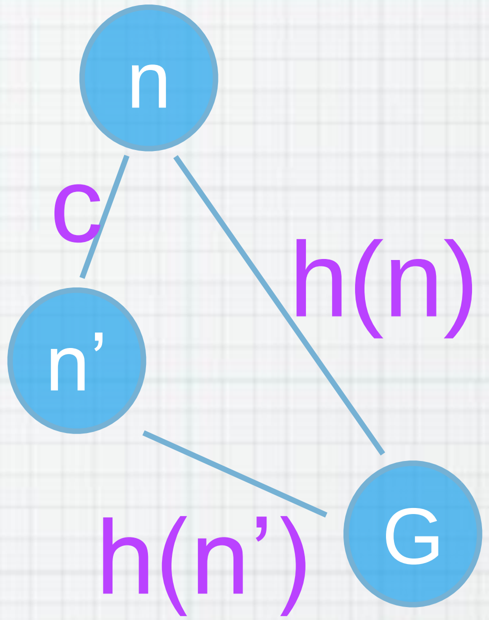
* $h(n) <= c(n,a,n') + h(n')$

* $f(n') = g(n') + h(n')$

  $= g(n) + c(n,n') + h(n')$

  $>= g(n) + h(n)$

* $f(n') >= f(n)$



The values of f() along any path are non-decreasing

First goal expanded will be optimal since future ones are more expensive

Informed Search Review

- Instead of Breadth/Depth, “Best” first
- Evaluation function = f(n) = desirability
- h(n) = estimated cost of cheapest path
- Greedy, only looks ahead: h(n)
- A*, consider total path cost: g(n)+h(n)
- Admissible heuristic = under-estimate