Lecture 7
Artificial Intelligence

Khola Naseem
khola.naseem@uet.edu.pk

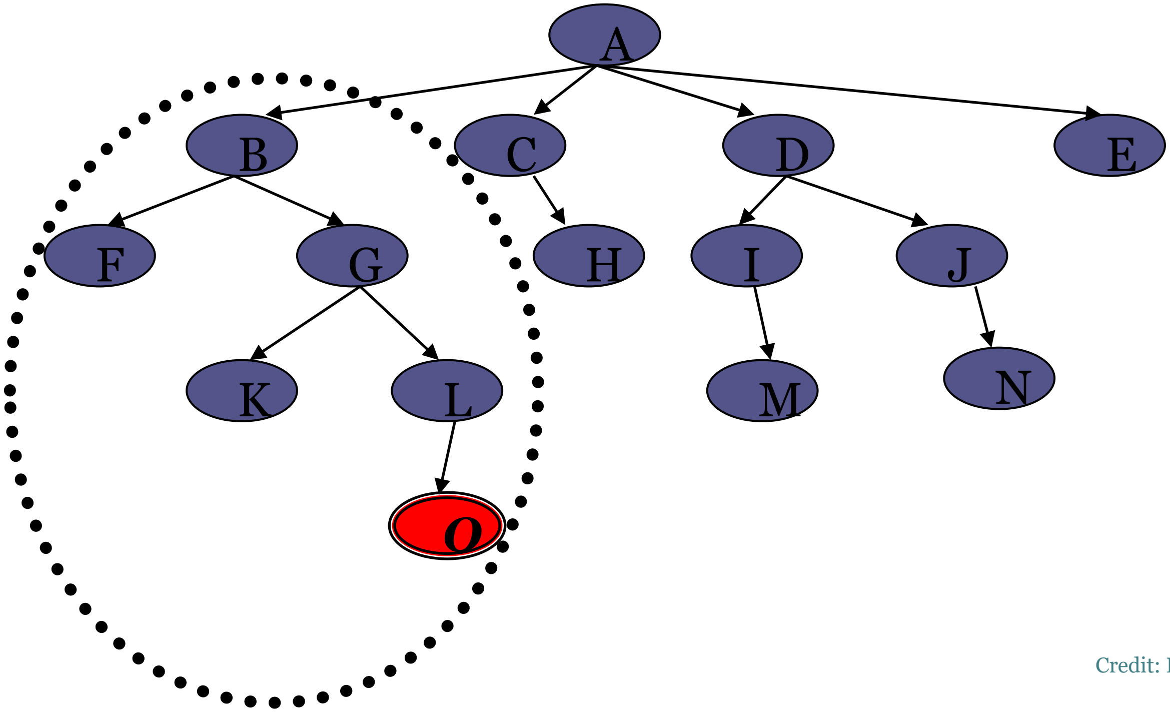
Search strategies

- Informed search, or heuristic search
  - a cleverer strategy that searches toward the goal,
  - based on the information from the current state so far
  - E.g. A*, Heuristic DFS, Best first search

Informed search, or heuristic search

- With knowledge, one can search the state space as if he was given "hints" when exploring a maze.
- Heuristic information in search = Hints
- Leads to dramatic speed up in efficiency.

Search only in this subtree!!



Credit: Khola Naseem

Informed search, or heuristic search

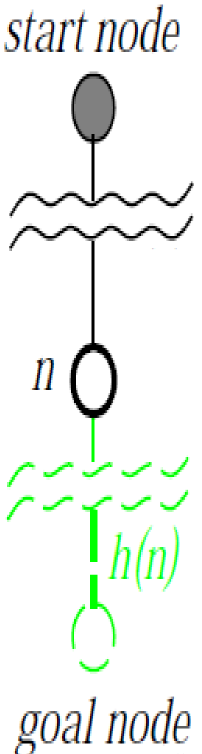
- Previous: at most b choices at each node and a depth of d at the goal node
  - An uninformed search algorithm would have to, in the worst case, search around O(b^d) nodes before finding a solution (Exponential Time Complexity).
- Heuristics improve the efficiency of search algorithms by reducing the effective branching factor from b to (ideally) a lower constant b* such that
  - $1 \leq b^* << b$

Evaluation Function

- An evaluation function $f(n)$ gives an estimation on the "cost" of node n in a tree/graph
- So that the node with the least cost among all possible choices can be selected for expansion first

One Approach to calculate heuristic function

- Evaluation function f measures the estimated cost of getting to the goal from the current state:
  - $f(n) = h(n)$
  - where
    - $h(n)$ = an estimate of the cost to get from state n to a goal state



Credit: Khola Naseem

Another way to find the cost function

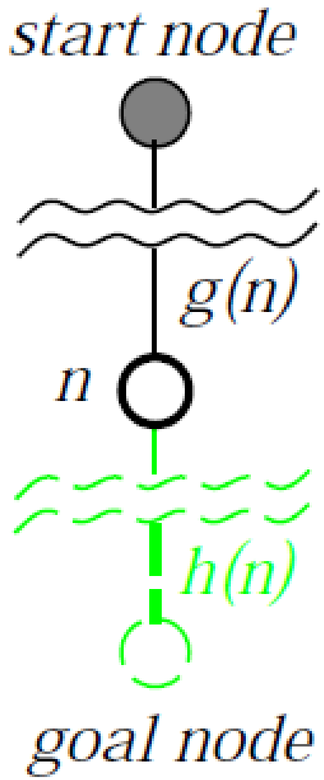
- Evaluation function f measures the estimated cost of getting to the goal state from the current state and the cost of the existing path to it:

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

where

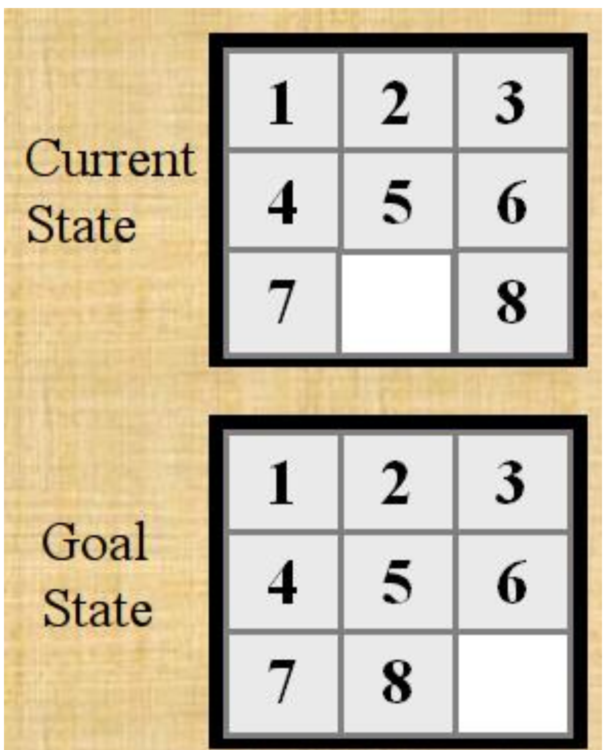
$g(n)$ = an *exact* cost to get to n (from initial state)

$h(n)$ = an estimate of the cost to get from state n to a goal state

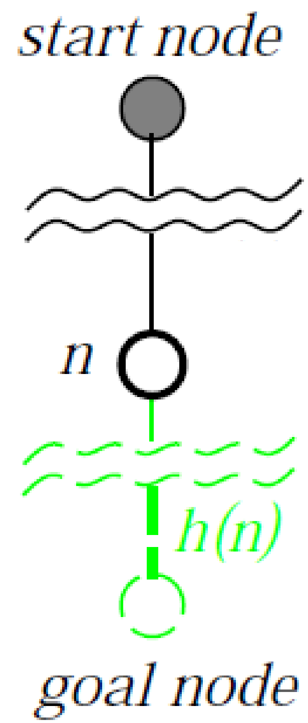


Example 8-Puzzle tree

- h1: The number of misplaced tiles (squares with number).
- h2: The sum of the distances of the tiles from their goal positions.
  - h1: The number of misplaced tiles (not including the blank)

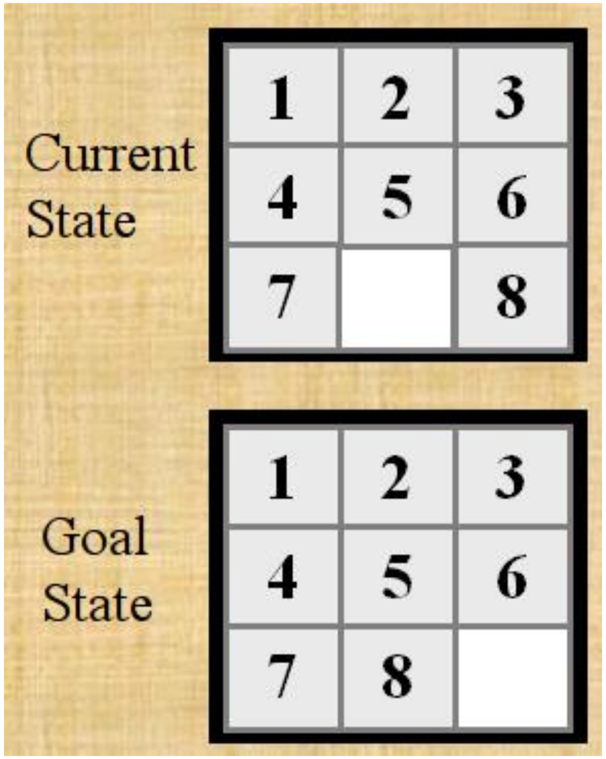


- *Notation:* $f(n) = h(n)$
- *h(current state) = 1*
- Because this state has one Y



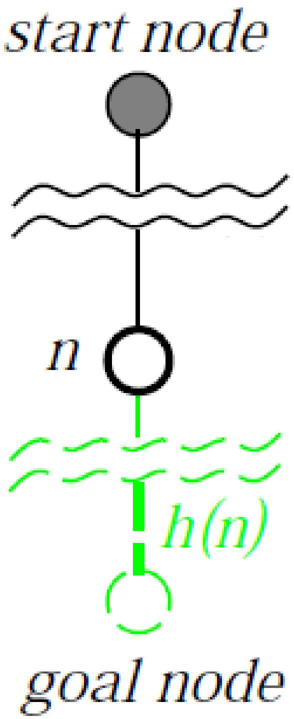
Example 8-Puzzle tree

- h1: The number of misplaced tiles (not including the blank)



N	N	N
N	N	N
N	Y	

- *Notation:* $f(n) = h(n)$
- *h(current state)* = *1*
- Because this state has one Y

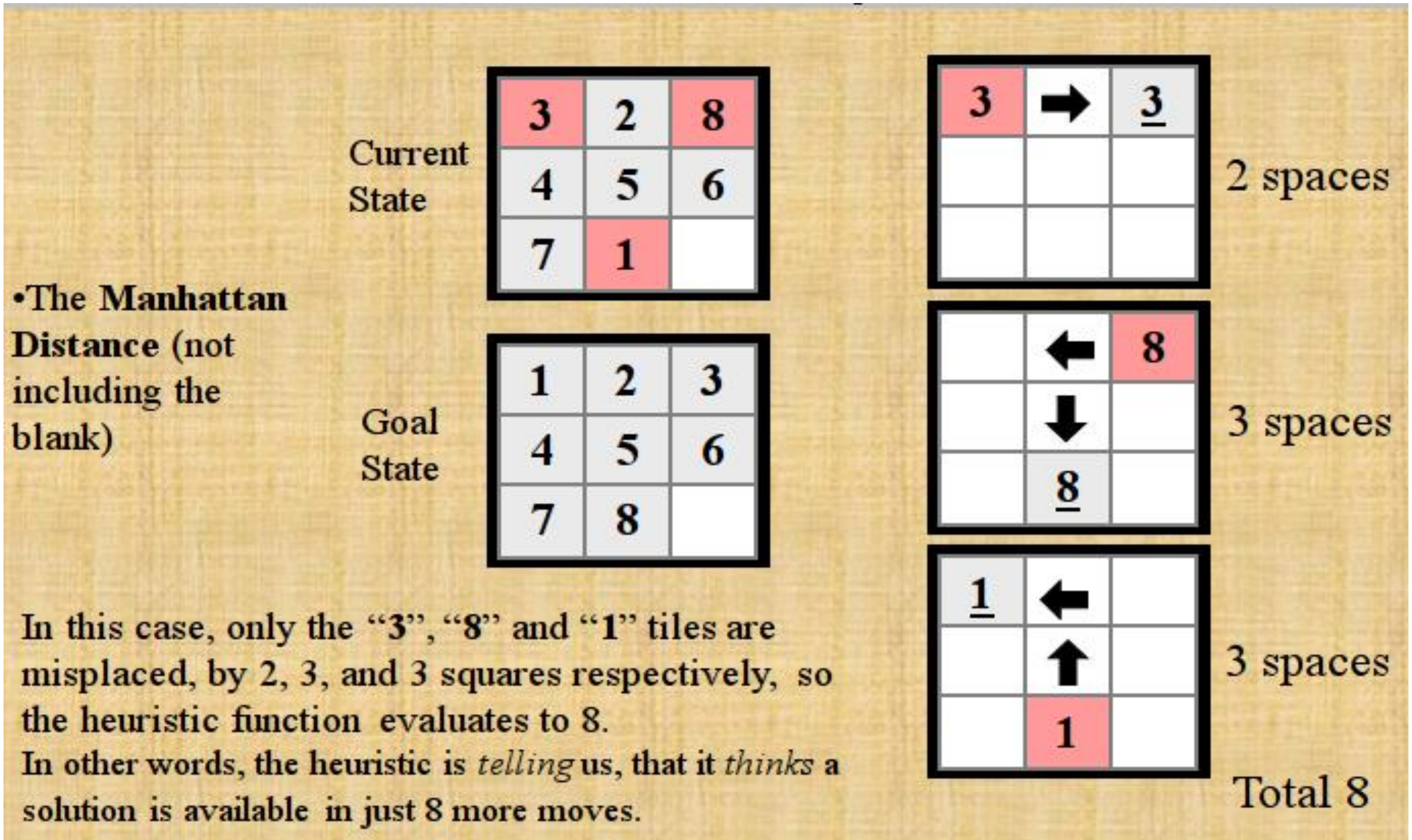


- Only "**8**" is misplaced
- So the heuristic function evaluates to 1.
- The heuristic is *telling* us, that it *thinks* a solution might be available in just 1 more move.

Credit: Khola Naseem

Example 8-Puzzle tree

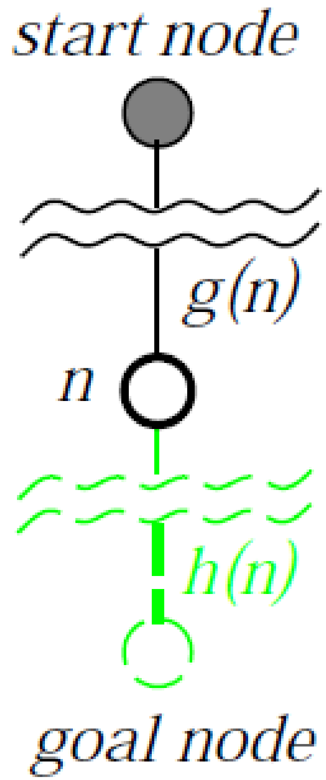
- h2: The sum of the distances of tiles from goal positions)



Notation: $f(n) = h(n)$, *h(current state) = 8*

Another way to find the cost function

- Evaluation function f measures the estimated cost of getting to the goal state from the current state and the cost of the existing path to it:

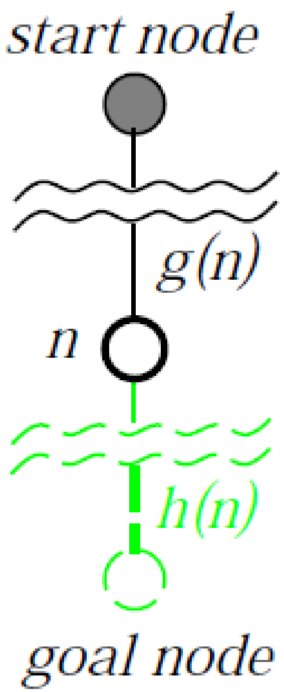


Summary of Evaluation / Heuristic Function

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

Special cases:

- Greedy (best-first) Search: $f(n) = h(n)$
- A* Search: $f(n) = g(n) + h(n)$



Credit: Khola Naseem

Informed Search Strategies

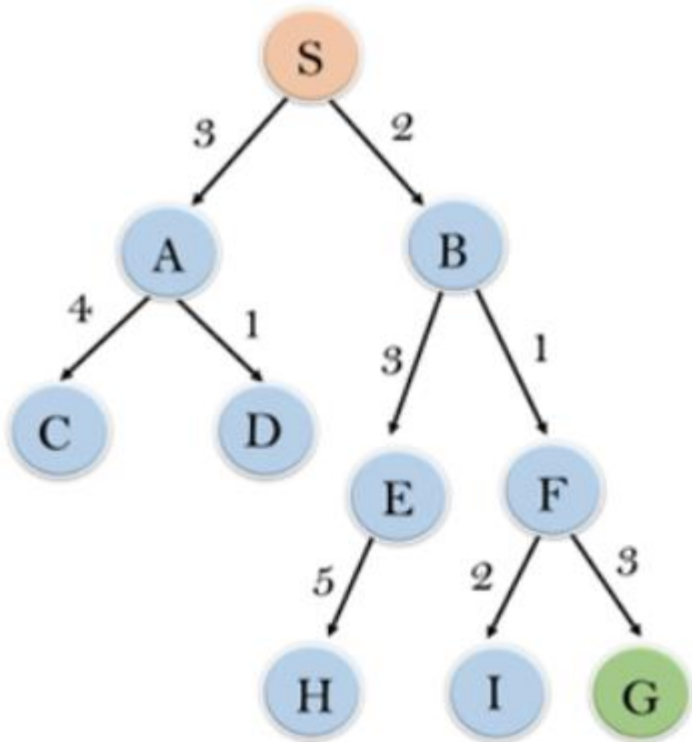
- **Greedy Best First Search :**eval-fn: $f(n) = h(n)$
- Greedy best-first search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly.
- Greedy best-first search algorithm always selects the path which appears best at that moment.
- It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search.
- In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function

Informed Search Strategies

- **Greedy Best First Search :**eval-fn: $f(n) = h(n)$
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms

Informed Search Strategies

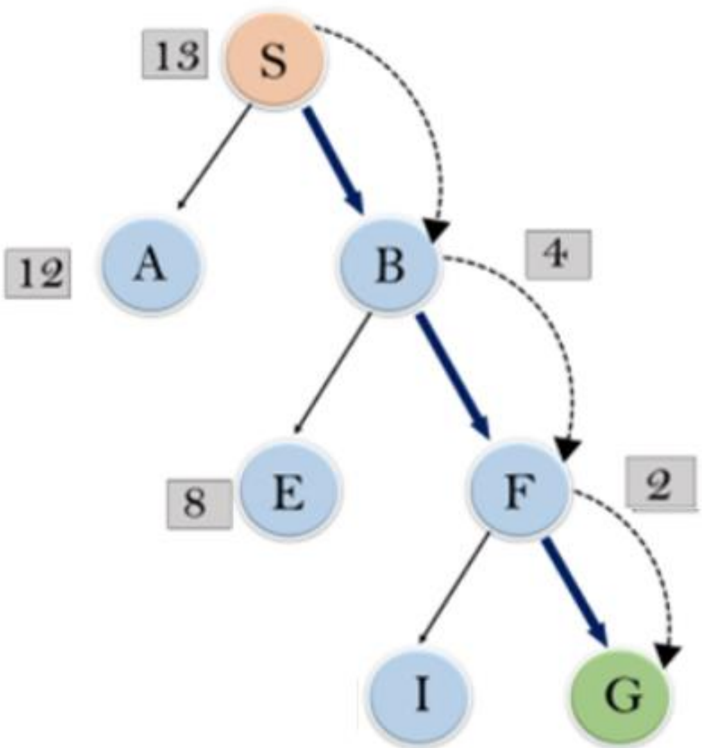
- **Greedy Best First Search :**eval-fn: $f(n) = h(n)$
- Example:



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

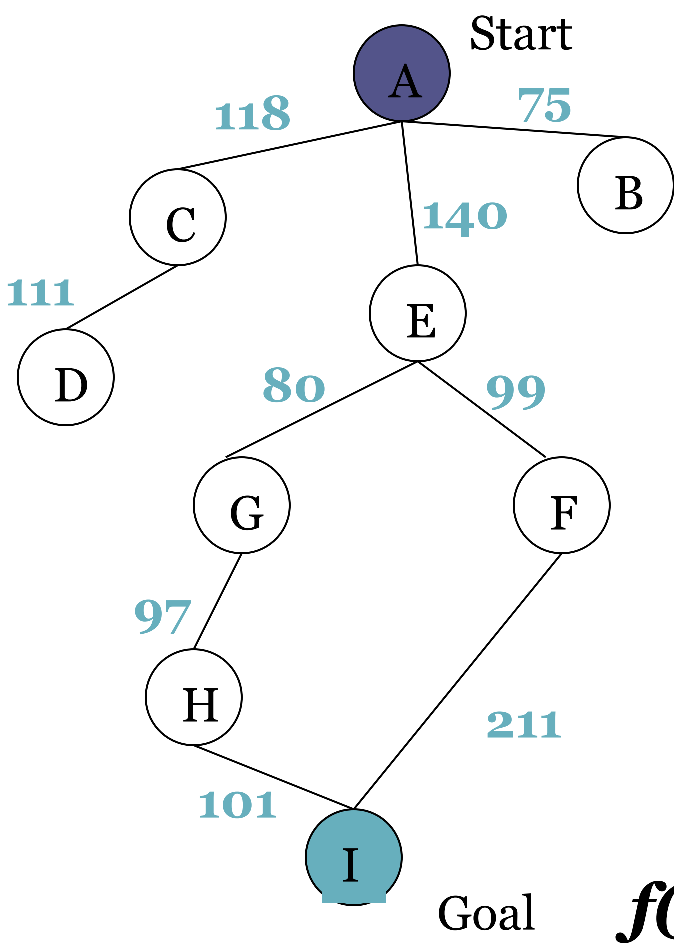
Informed Search Strategies

- **Greedy Best First Search :**eval-fn: $f(n) = h(n)$
- Example:



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

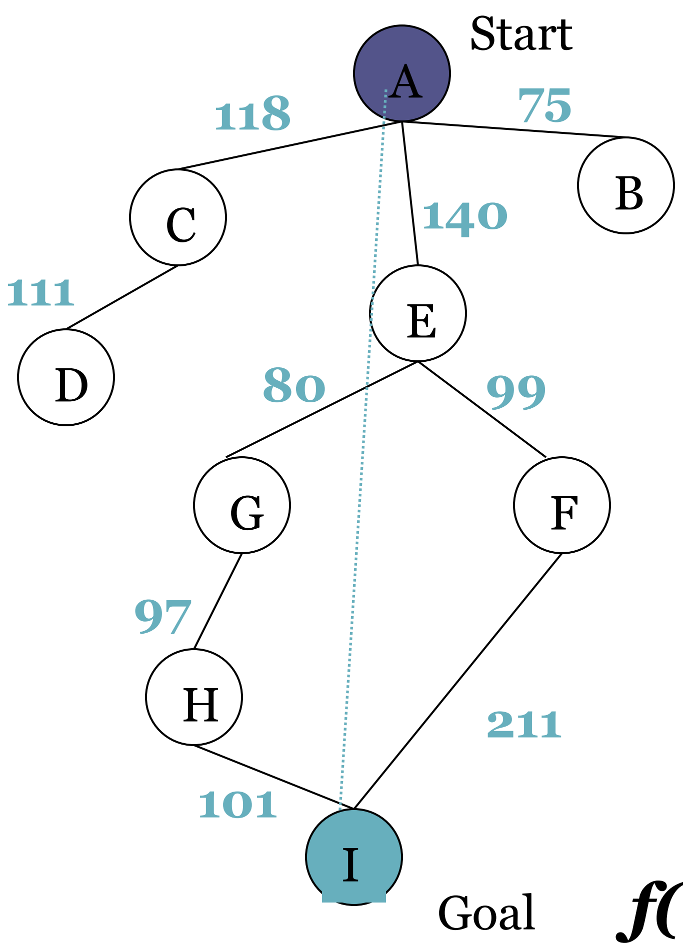
Greedy Search Example 2:



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$f(n) = h(n)$ = straight-line distance heuristic

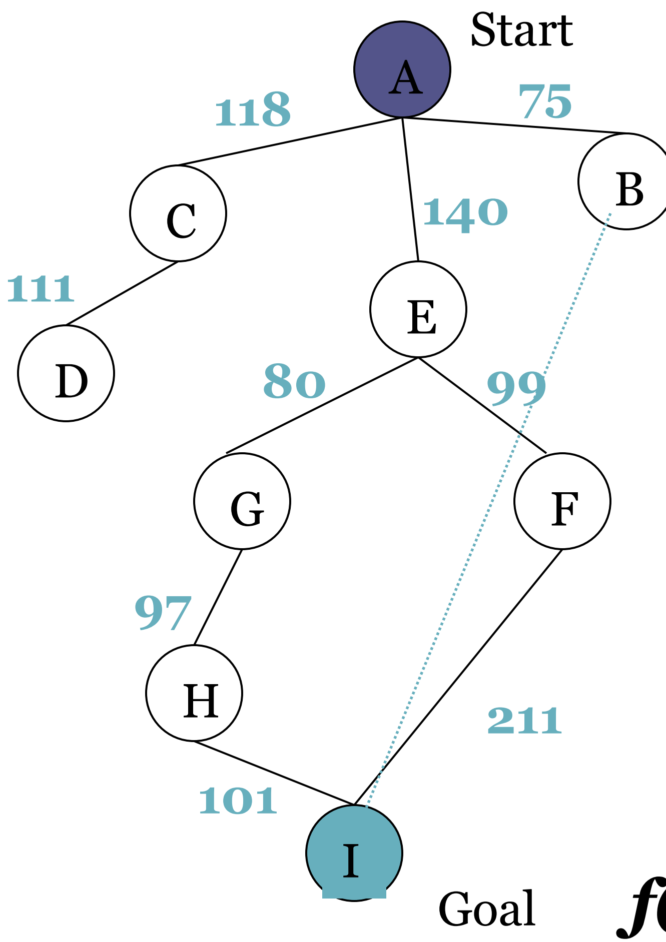
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$f(n) = h(n)$ = straight-line distance heuristic

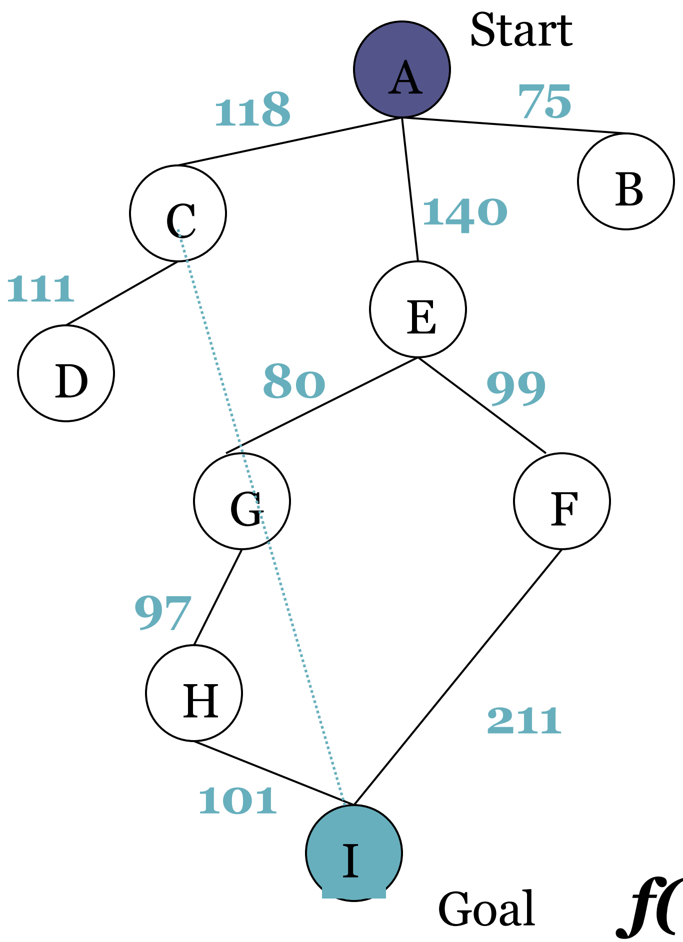
Greedy Search



State	Heuristic: h(n)
A	366
B	**374**
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$f(n) = h(n)$ = straight-line distance heuristic

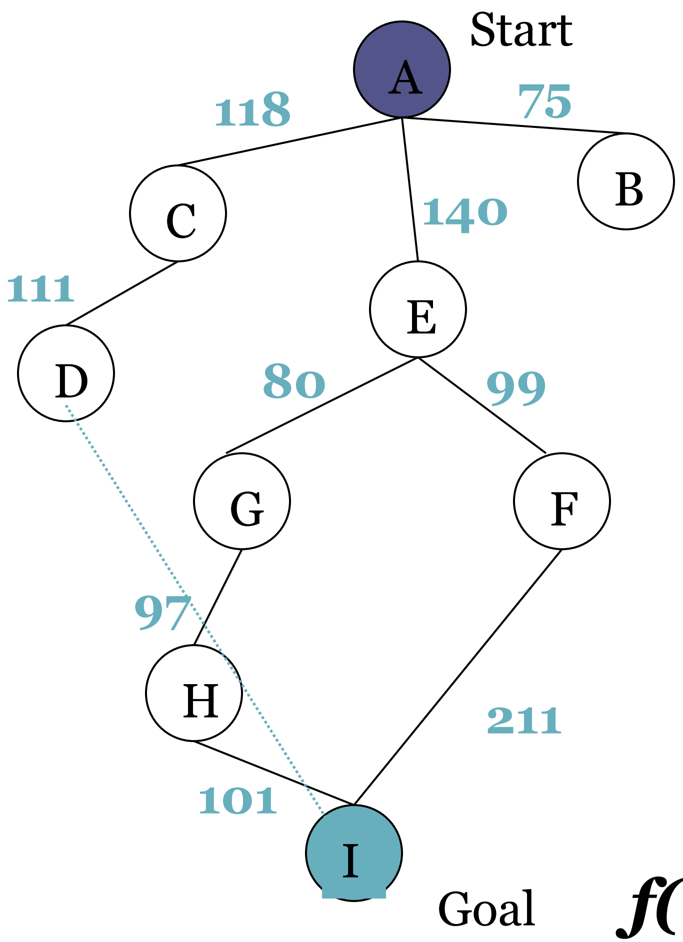
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$f(n) = h(n)$ = **straight-line distance heuristic**

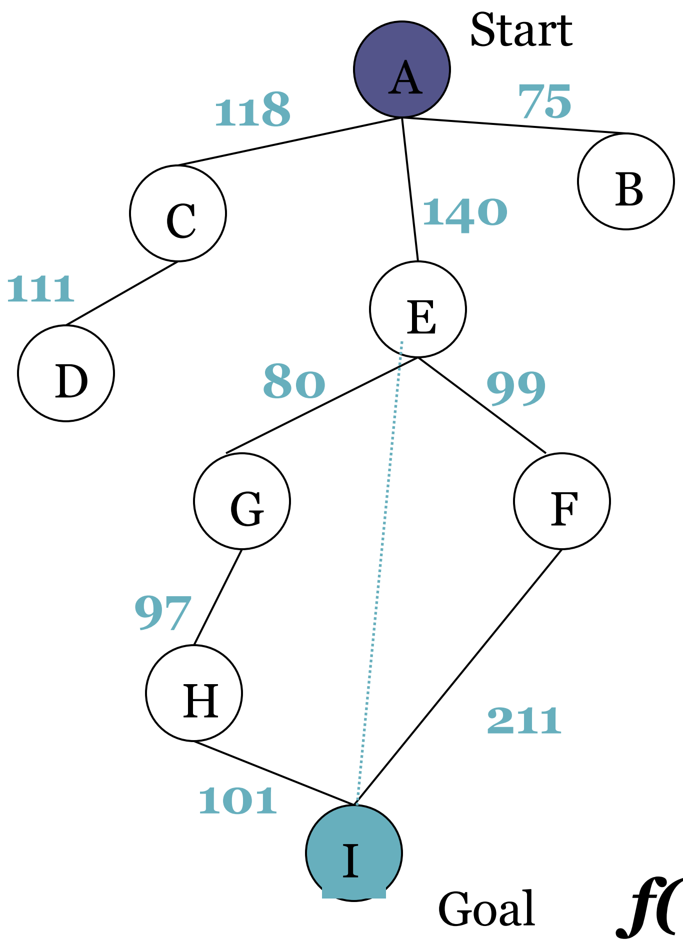
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	**244**
E	253
F	178
G	193
H	98
I	0

$f(n) = h(n)$ = straight-line distance heuristic

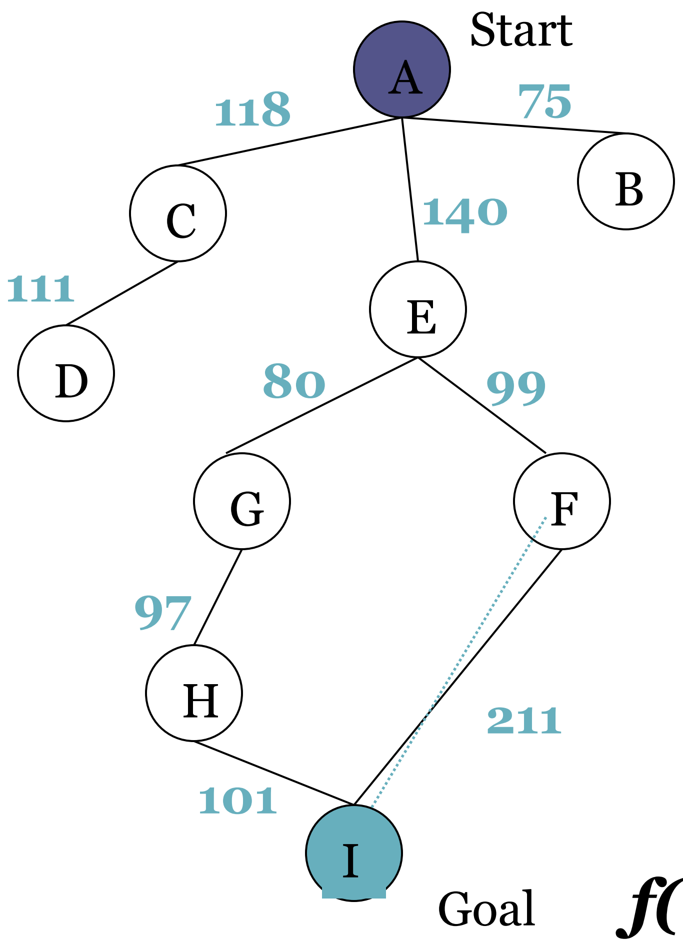
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	**253**
F	178
G	193
H	98
I	0

$f(n) = h\ (n)$ = straight-line distance heuristic

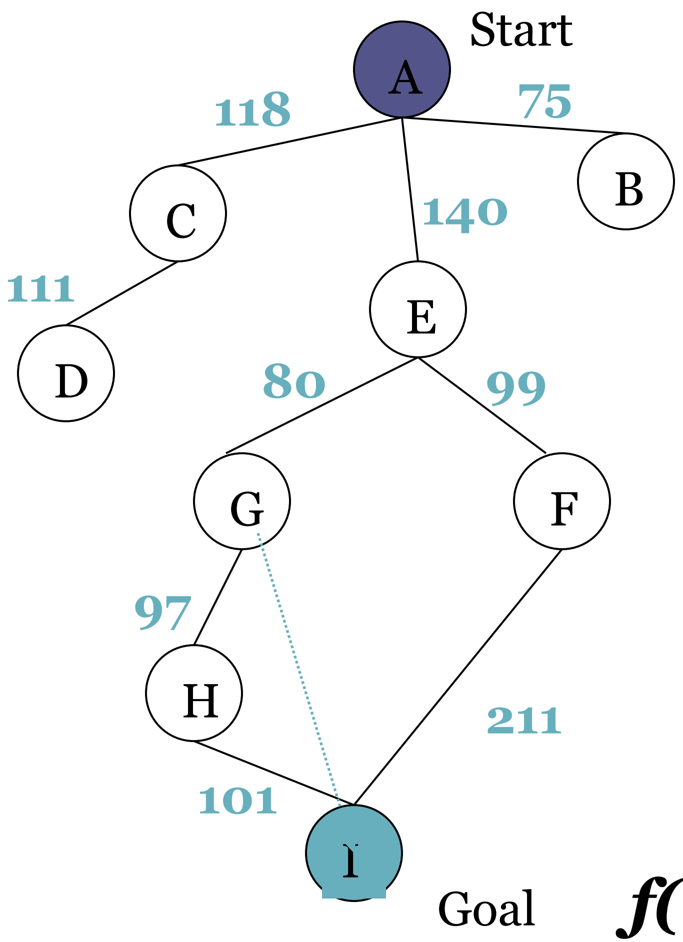
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	**178**
G	193
H	98
I	0

$f(n) = h(n)$ = straight-line distance heuristic

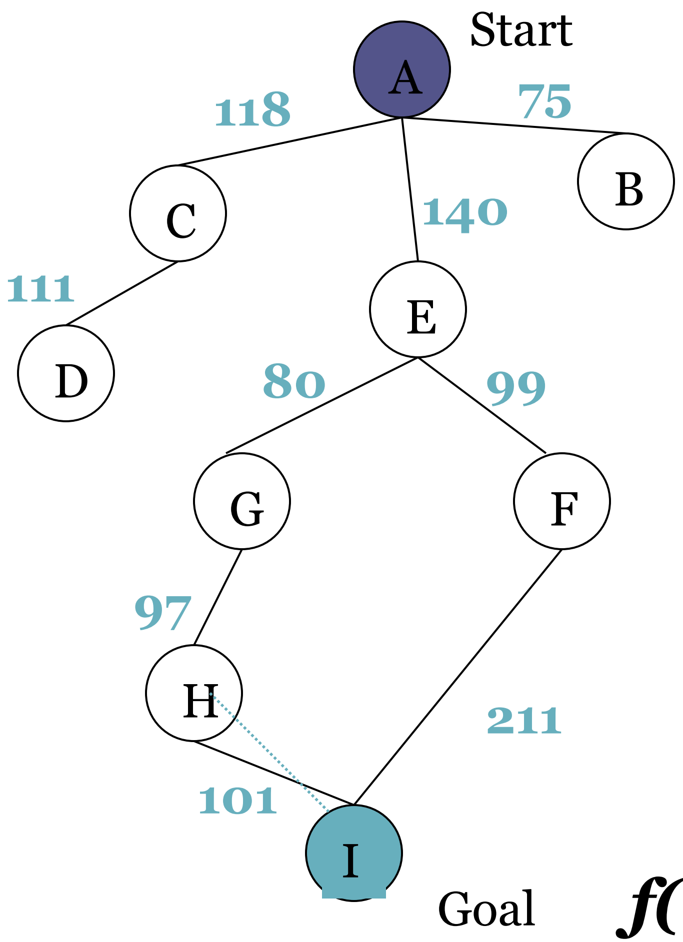
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$f(n) = h\,(n)$ = **straight-line distance heuristic**

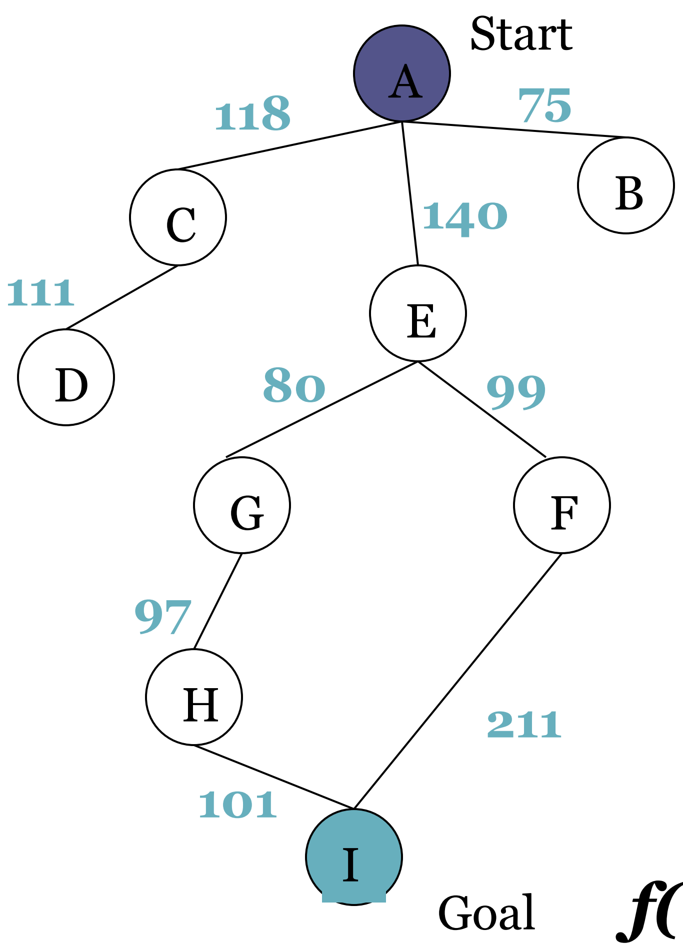
Greedy Search



State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	**98**
I	0

***f(n)* = *h (n)* = straight-line distance heuristic**

Greedy Search

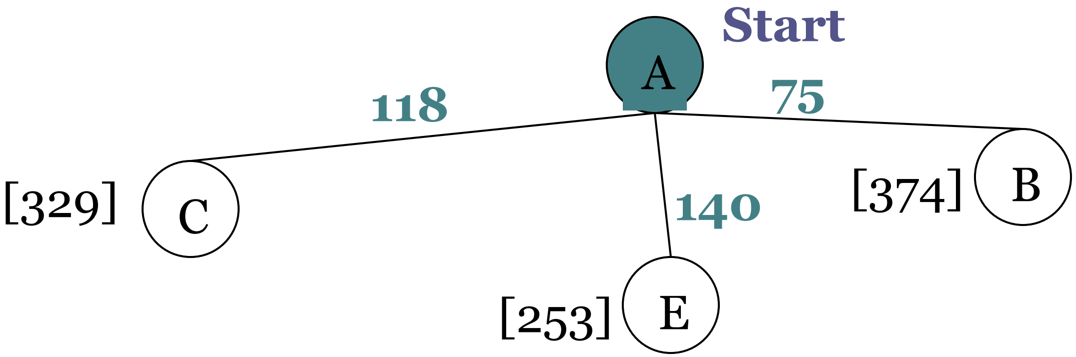


State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

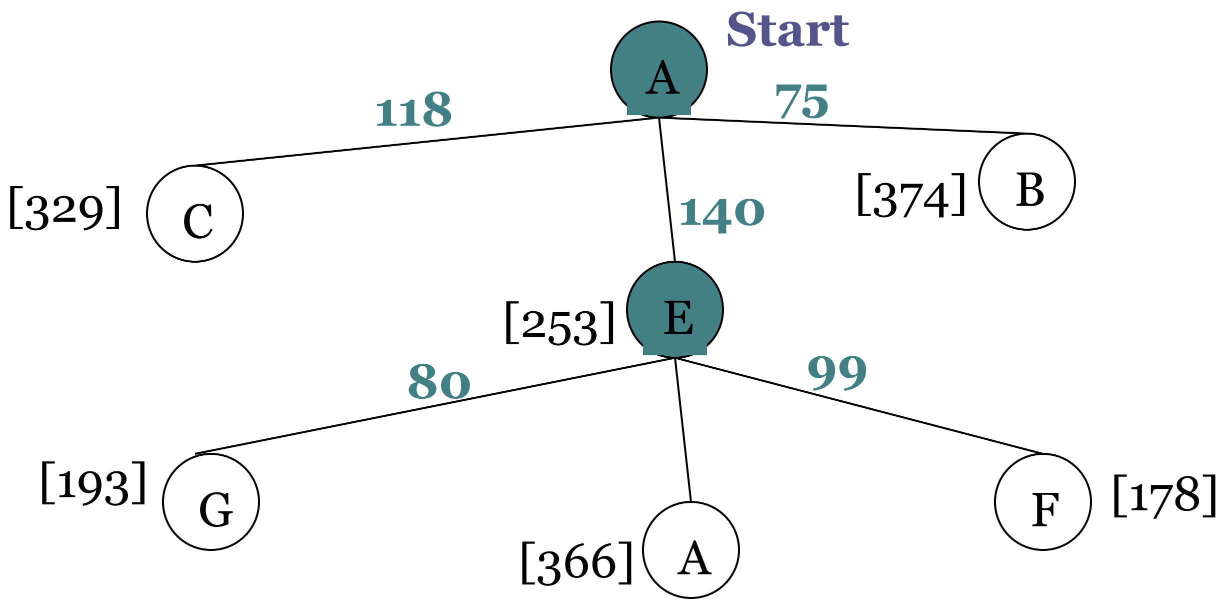
$f(n) = h\ (n)$ = straight-line distance heuristic

Greedy Search: Tree Search

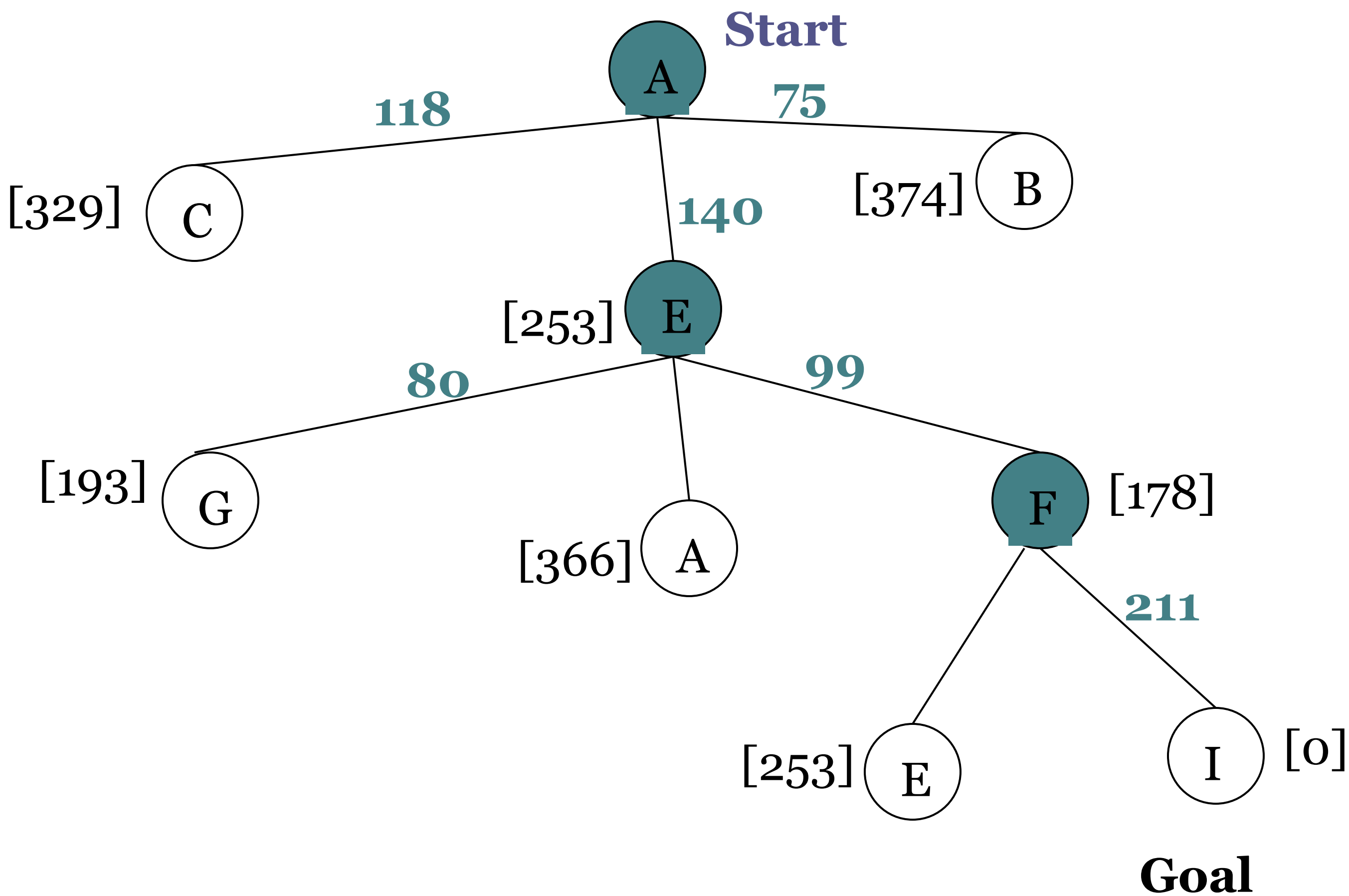
Greedy Search: Tree Search



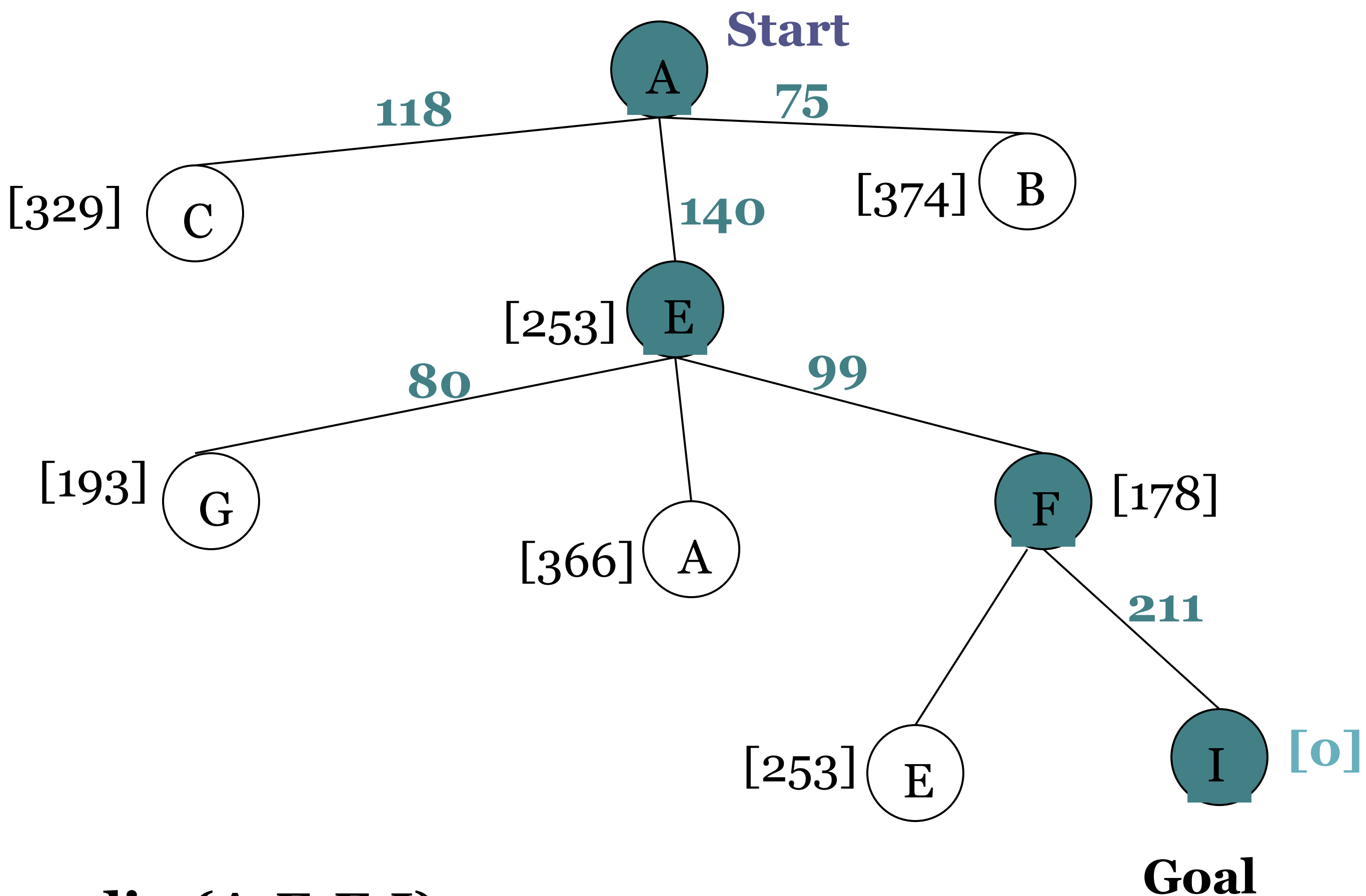
Greedy Search: Tree Search



Greedy Search: Tree Search

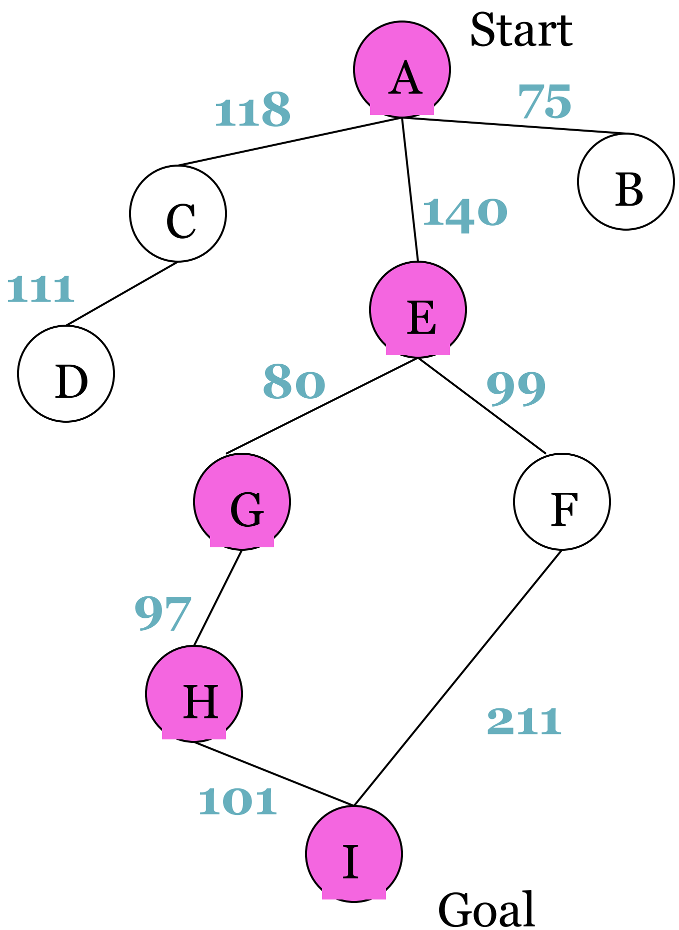


Greedy Search: Tree Search



Path cost=dist(A-E-F-I) = 140 + 99 + 211
= 450

Greedy Search: Optimal?



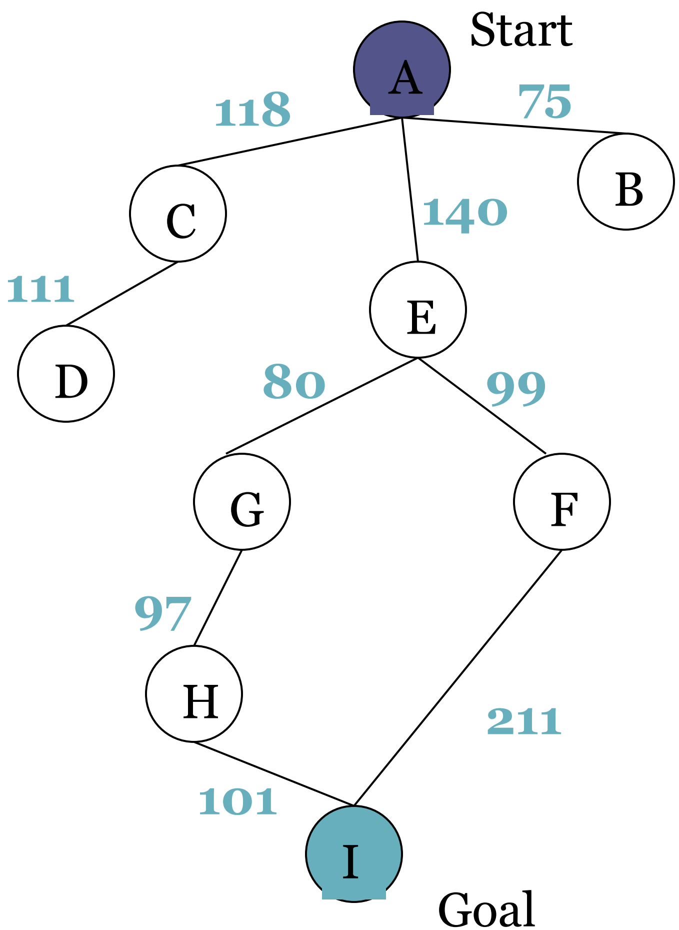
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Not optimal!

$f(n) = h(n)$ = straight-line distance heuristic

dist(A-E-G-H-I) =140+80+97+101=418

Greedy Search: Complete ?

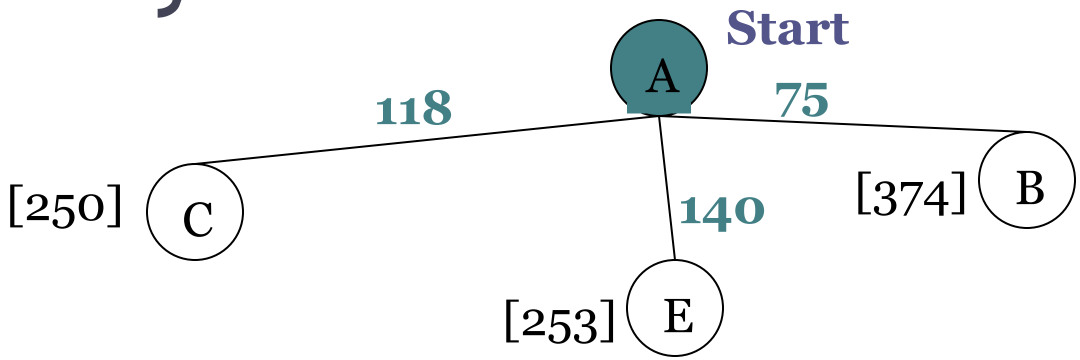


State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

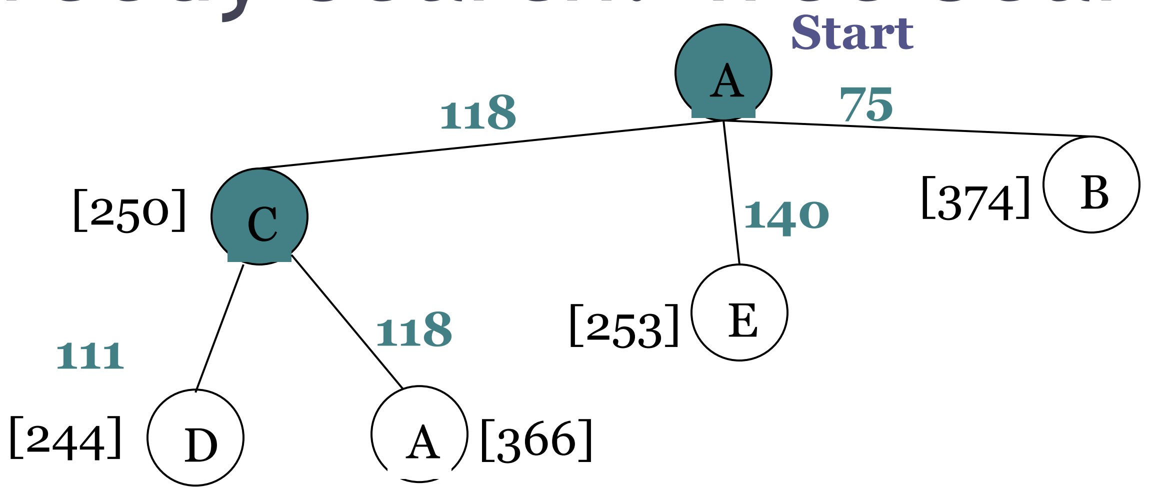
***f(n)* = *h (n)* = straight-line distance heuristic**

Greedy Search: Tree Search

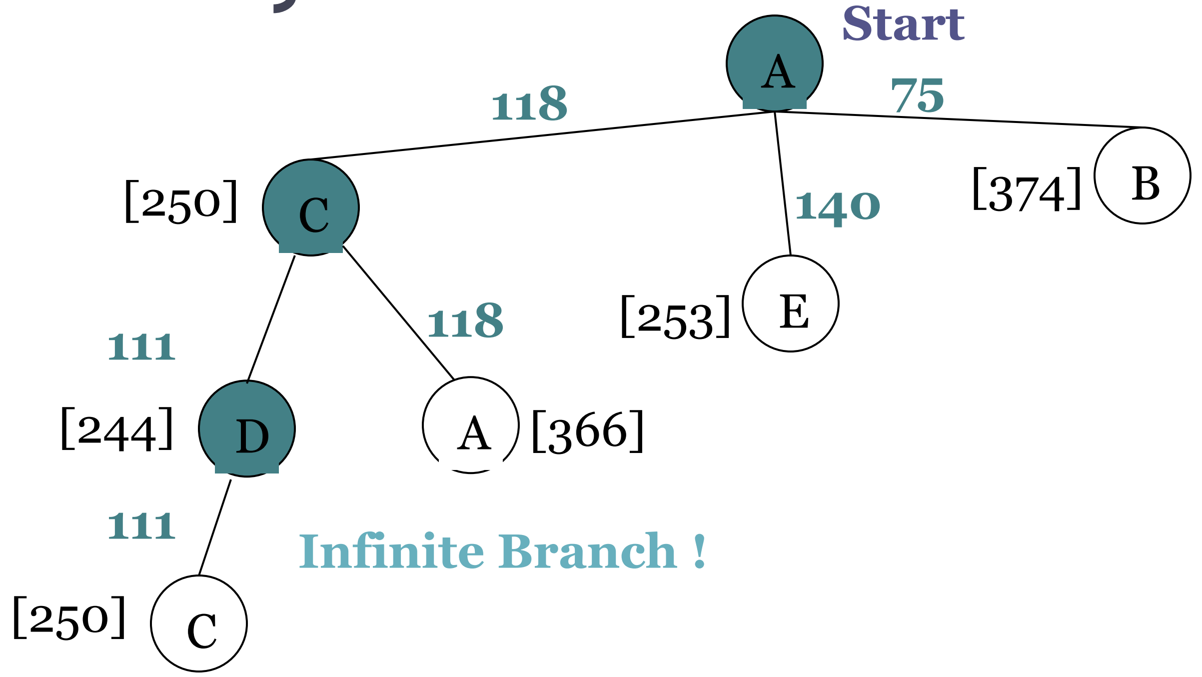
Greedy Search: Tree Search



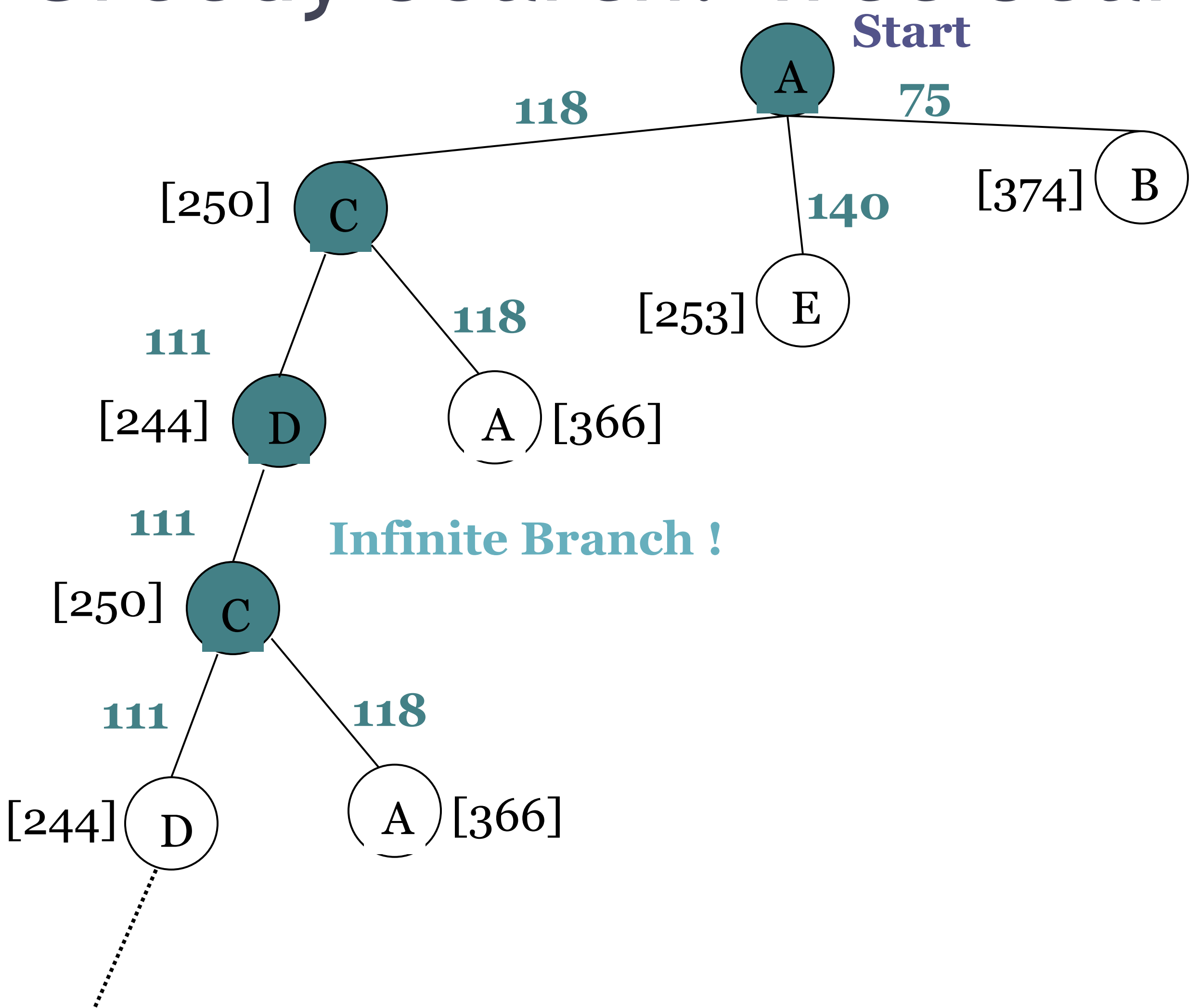
Greedy Search: Tree Search



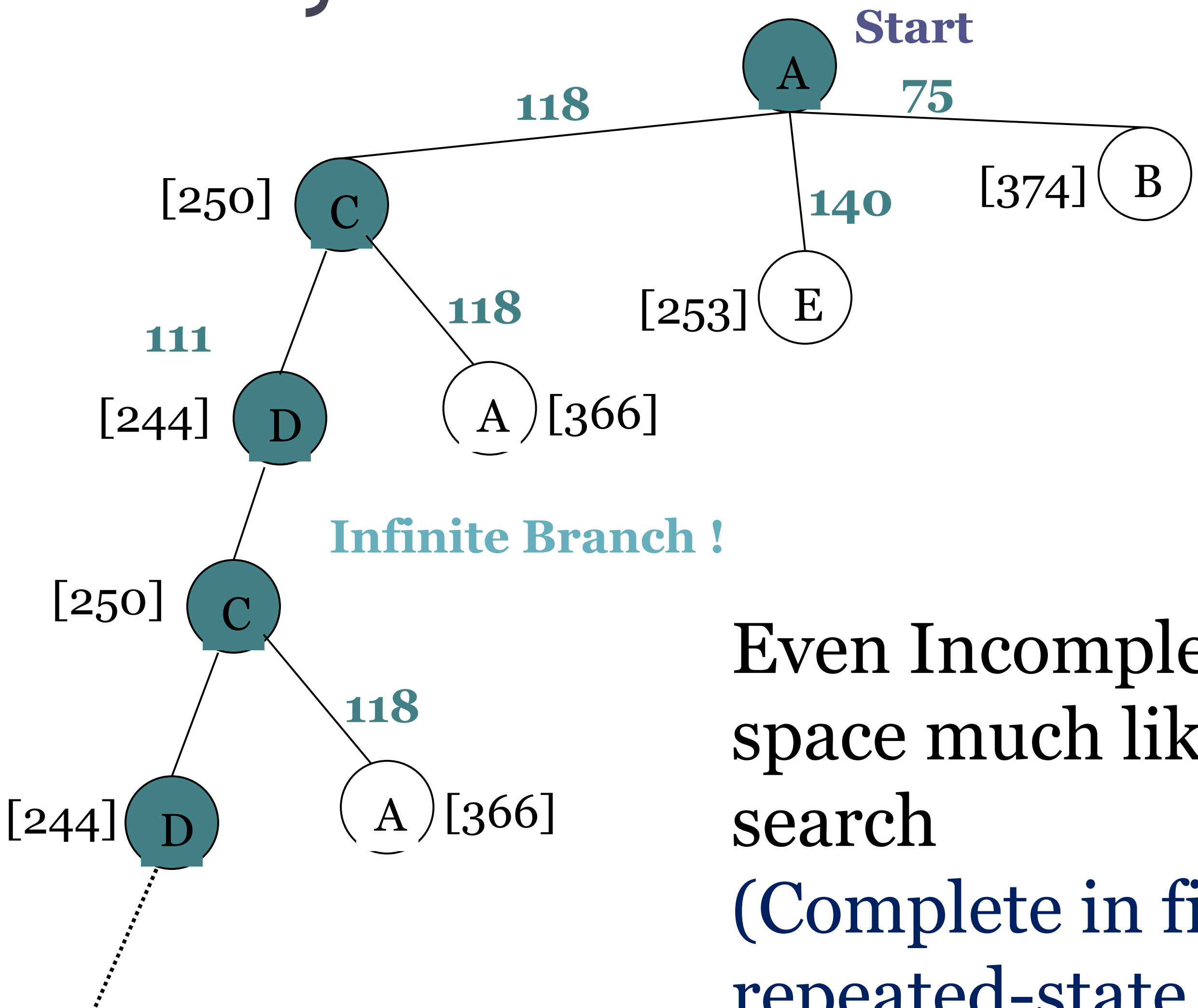
Greedy Search: Tree Search



Greedy Search: Tree Search

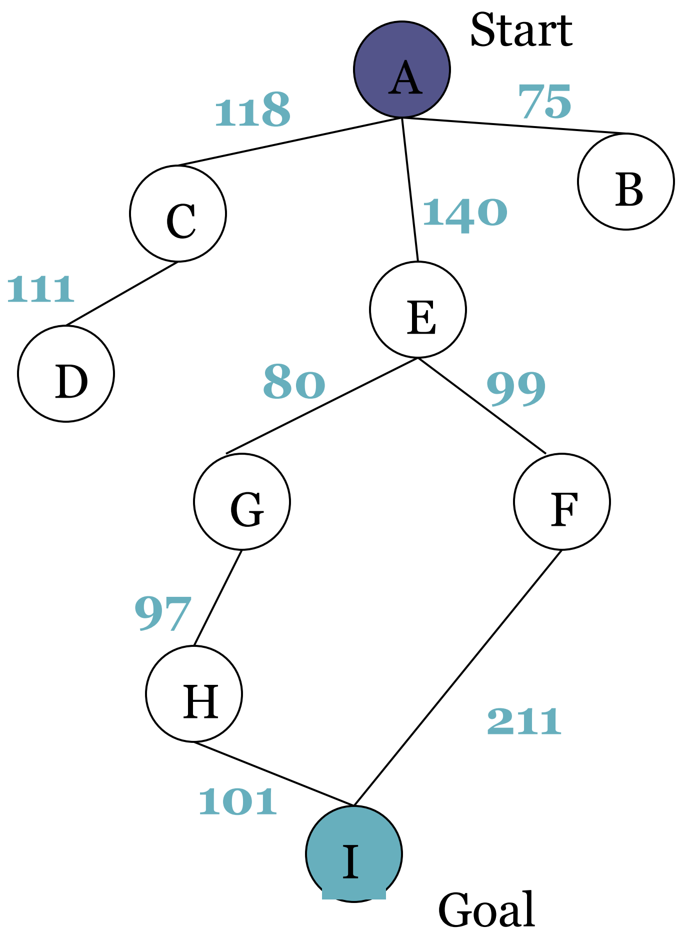


Greedy Search: Tree Search



Even Incomplete in finite state space much like depth first search
(Complete in finite space with repeated-state checking)

Greedy Search: Time and Space Complexity ?



- Greedy search is not optimal.
- Greedy search is incomplete
- In the worst case, the Time and Space Complexity of Greedy Search are both $O(b^m)$

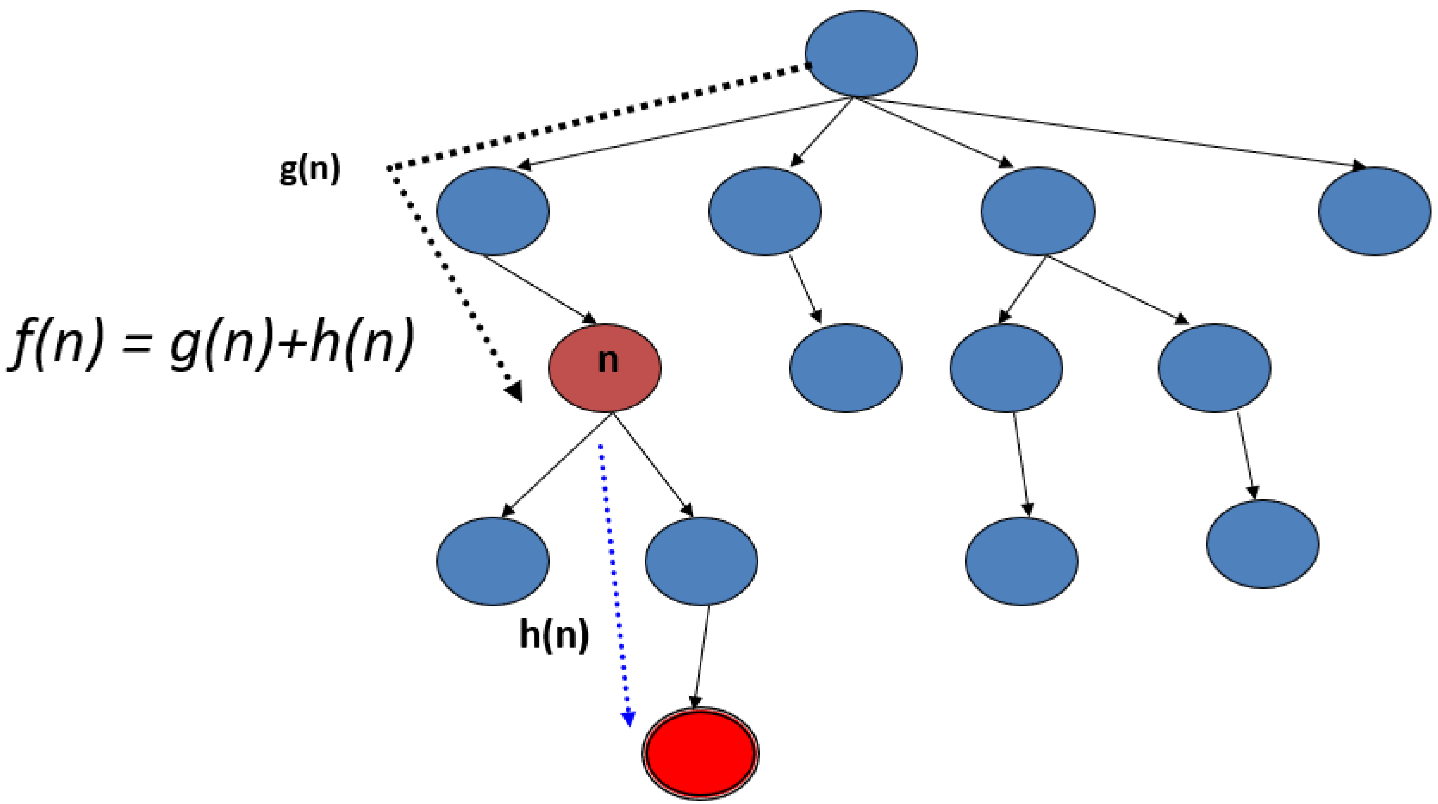
Where b is the branching factor and m the maximum path length

Informed Search Strategies

- **A* Search eval-fn: $\boldsymbol{f(n)=g(n)+h(n)}$**
- Although Greedy Search can considerably cut the search time (efficient), it is neither optimal nor complete.
- A* uses a priority function which combines g(n) and h(n): $f(n) = g(n) + h(n)$
- g(n) is the exact cost to reach node n from the initial state. Cost so far up to node n.
- h(n) is an estimation of the remaining cost to reach the goal.

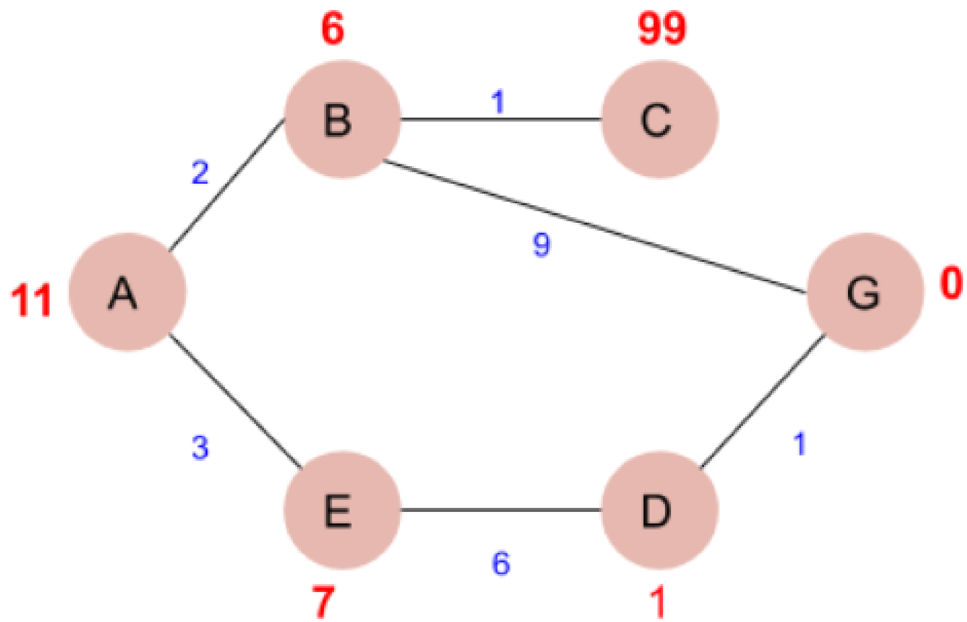
Informed Search Strategies

- **A* Search eval-fn:** $\boldsymbol{f(n)=g(n)+h(n)}$

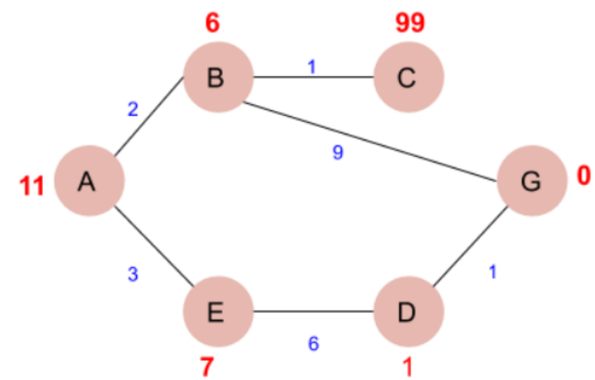


Informed Search Strategies

- **A* Search eval-fn: $f(n)=g(n)+h(n)$**

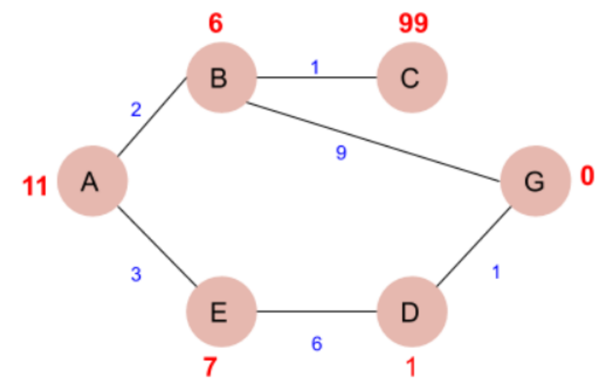


Informed Search Strategies



- **A* Search eval-fn: $f(n)=g(n)+h(n)$**
- Example:
- g(x) + h(x) = f(x)
- 0+ 11 =11 Thus for A, we can write A=11
- Now from A, we can go to point B or point E, so we compute f(x) for each of them
- A → B = 2 + 6 = 8
- A → E = 3 + 7 = 10 (hold)
- Since the cost for A → B is less, we move forward with this path and compute the f(x) for the children nodes of B
- Since there is no path between C and G, the heuristic cost is set to infinity or a very high value
- A → B → C = (2 + 1) + 99= 102
- A → B → G = (2 + 9) + 0 = 11

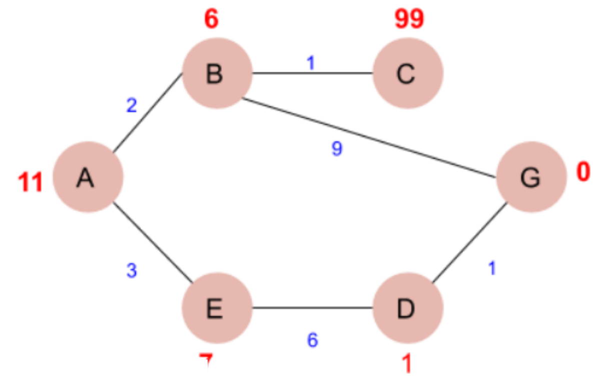
Informed Search Strategies



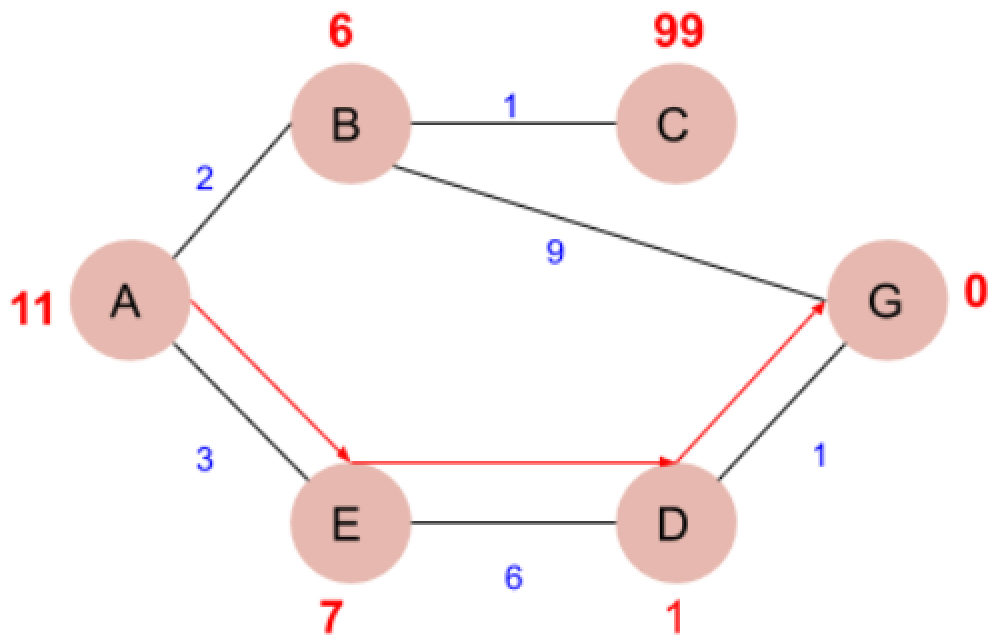
- **A* Search eval-fn: $f(n)=g(n)+h(n)$**
- Example:
- Here the path A → B → G has the least cost but it is still more than the cost of A → E, thus we explore this path further
- A → E → D = (3 + 6) + 1 = 10
- Comparing the cost of A → E → D with all the paths we got so far and as this cost is least of all we move forward with this path.
- And compute the f(x) for the children of D
- A → E → D → G = (3 + 6 + 1) +0 =10
- Now comparing all the paths that lead us to the goal, we conclude that A → E → D → G is the most cost-effective path to get from A to G.

Informed Search Strategies

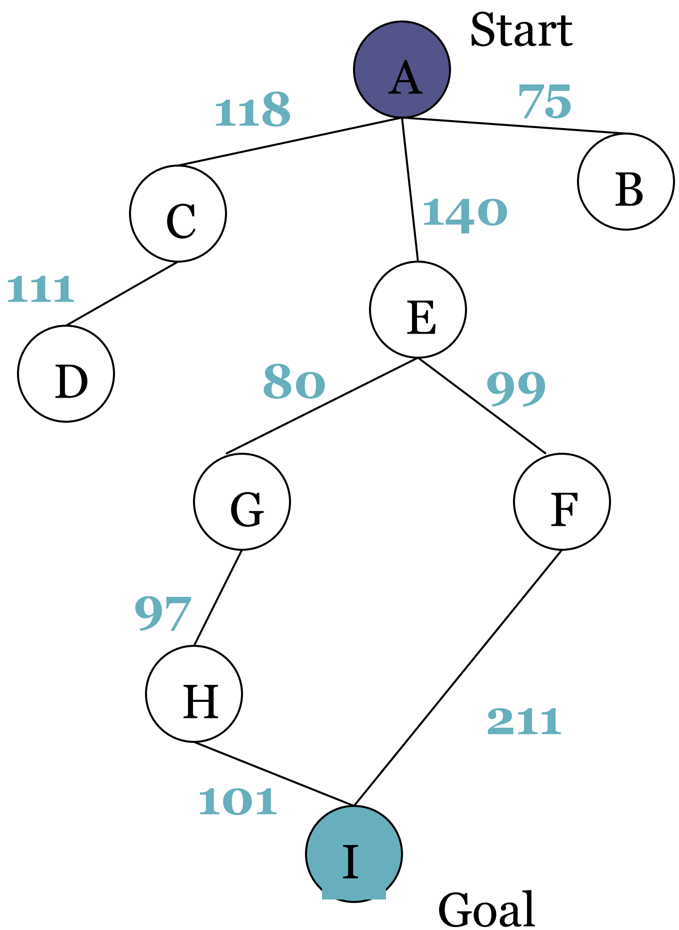
- **A* Search eval-fn: $f(n)=g(n)+h(n)$**



- Example:



A* Search



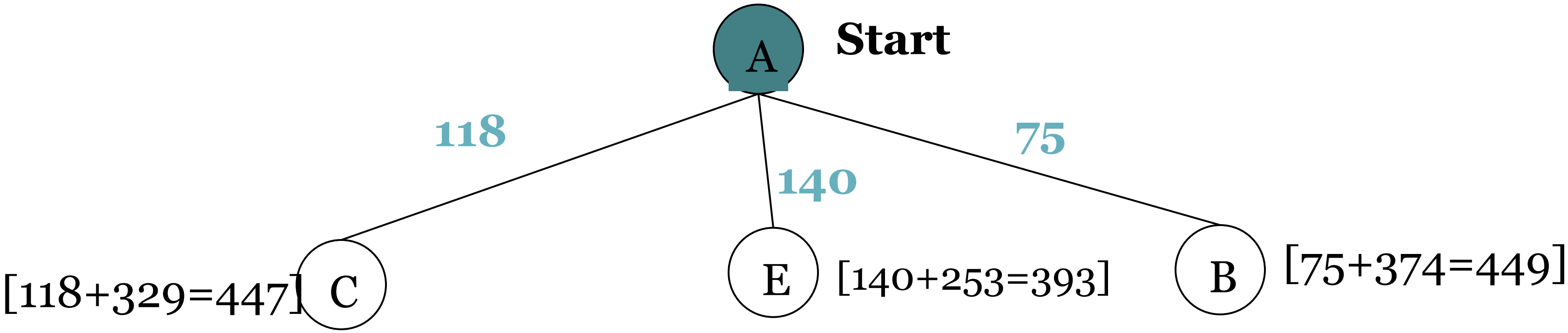
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

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

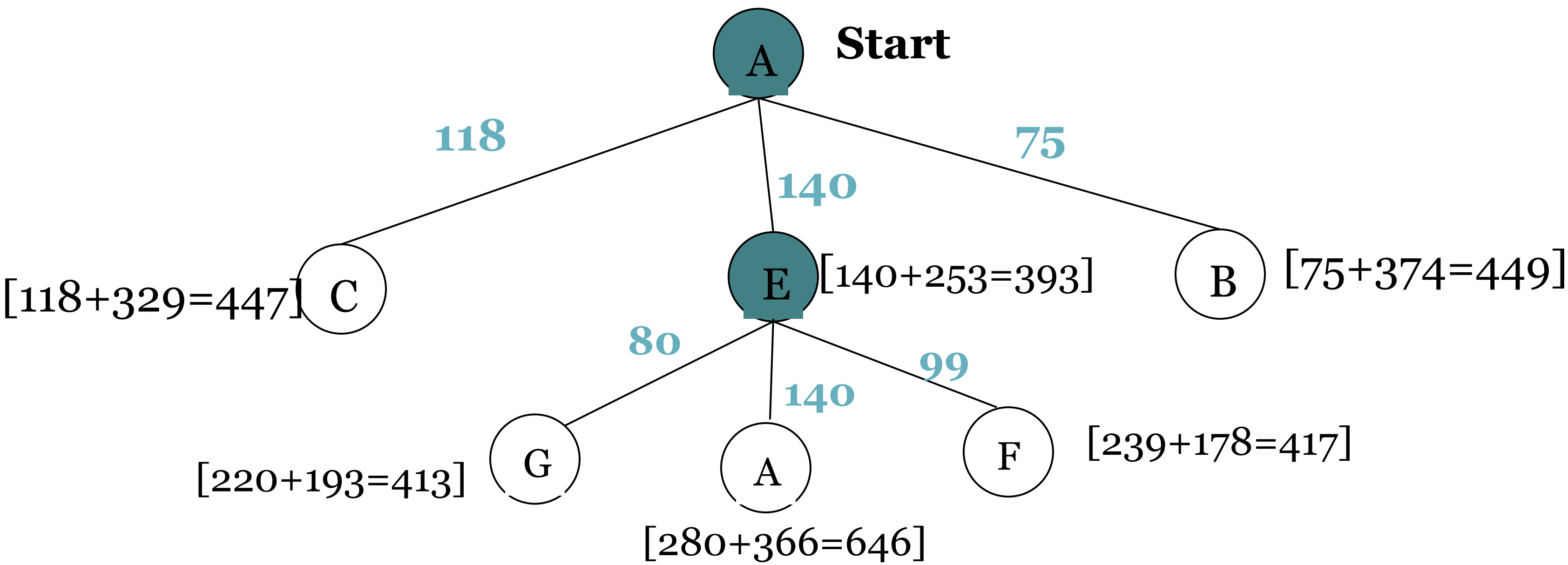
g(n): is the exact cost to reach node *n* from the initial state.

A* Search: Tree Search

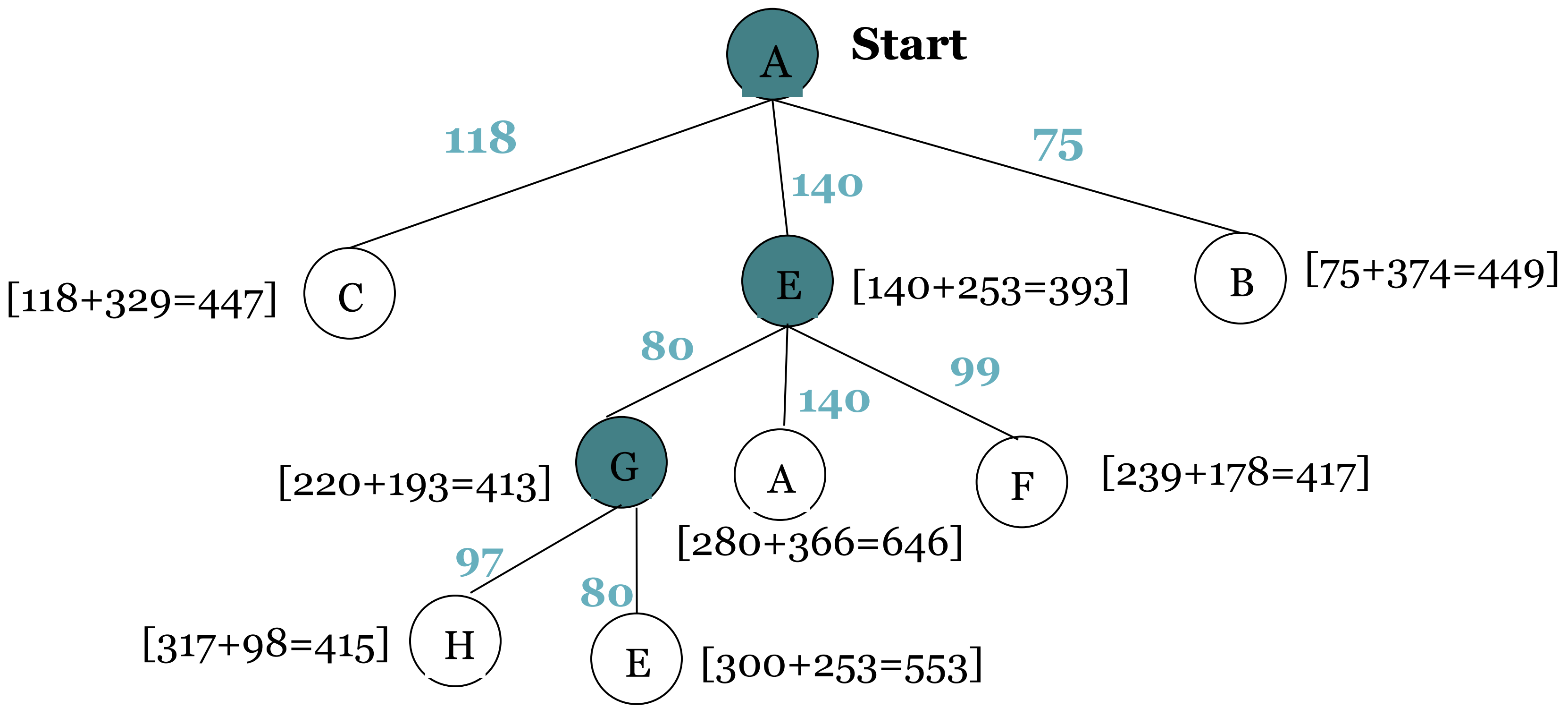
A* Search: Tree Search



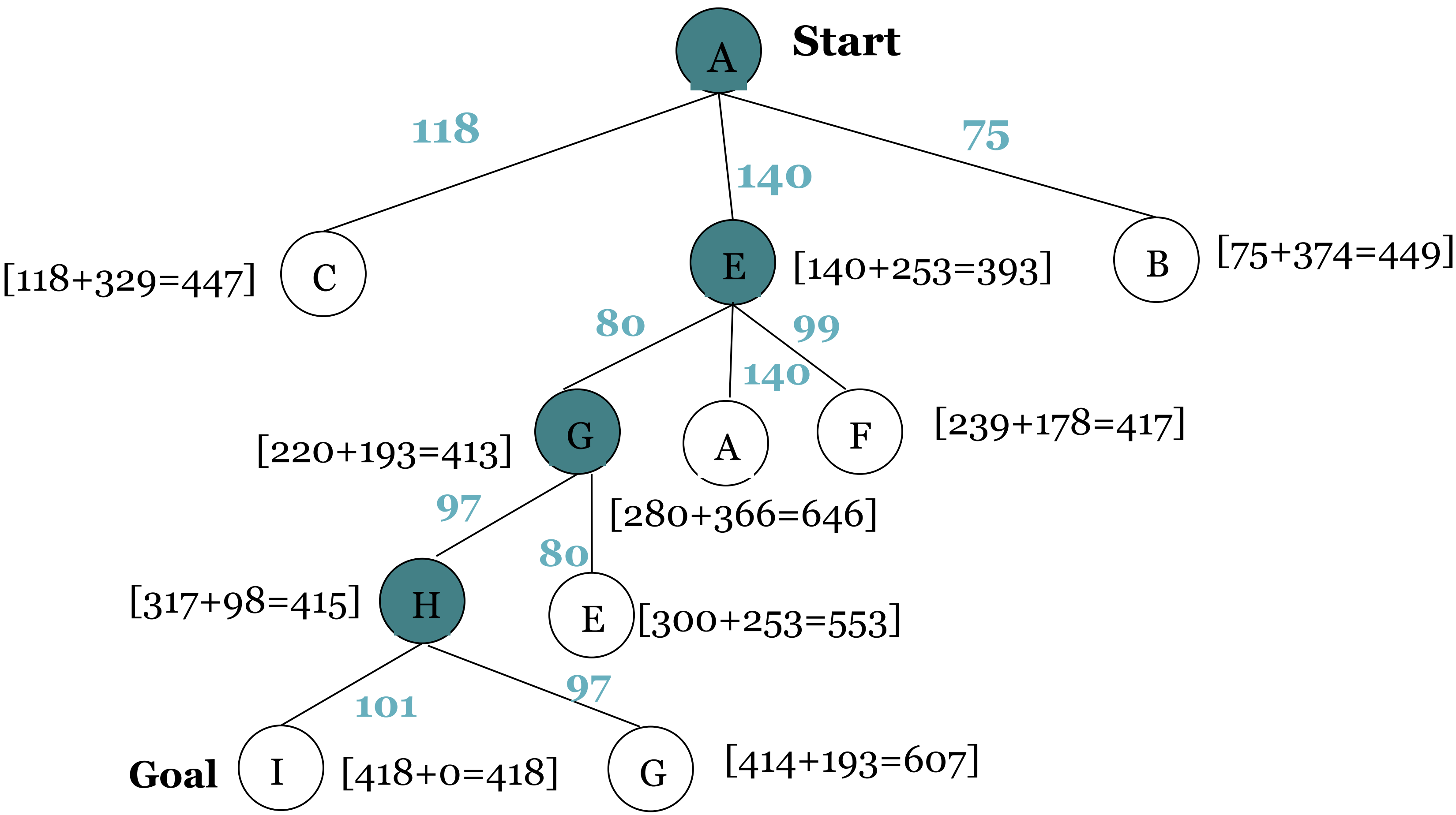
A* Search: Tree Search



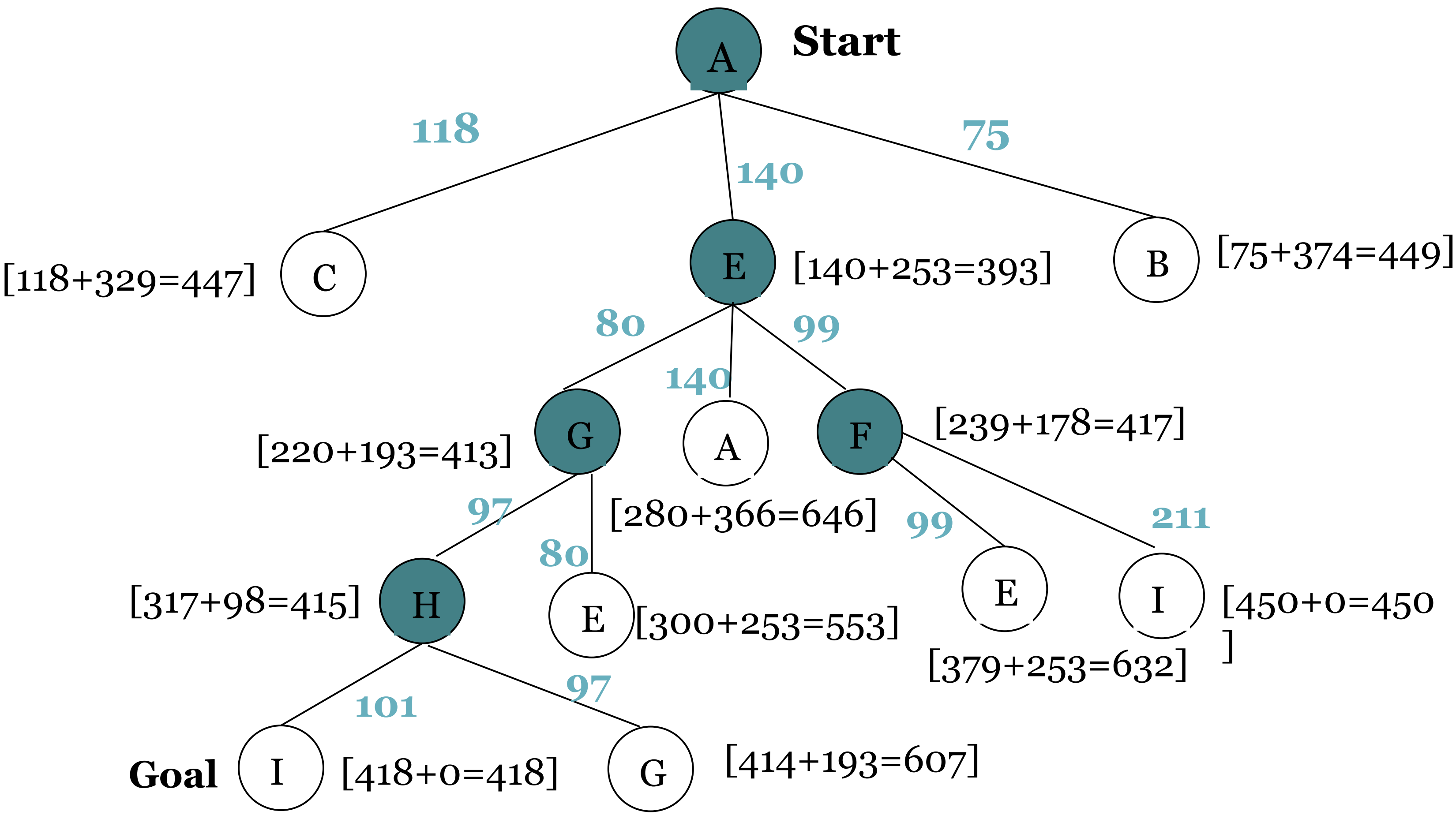
A* Search: Tree Search



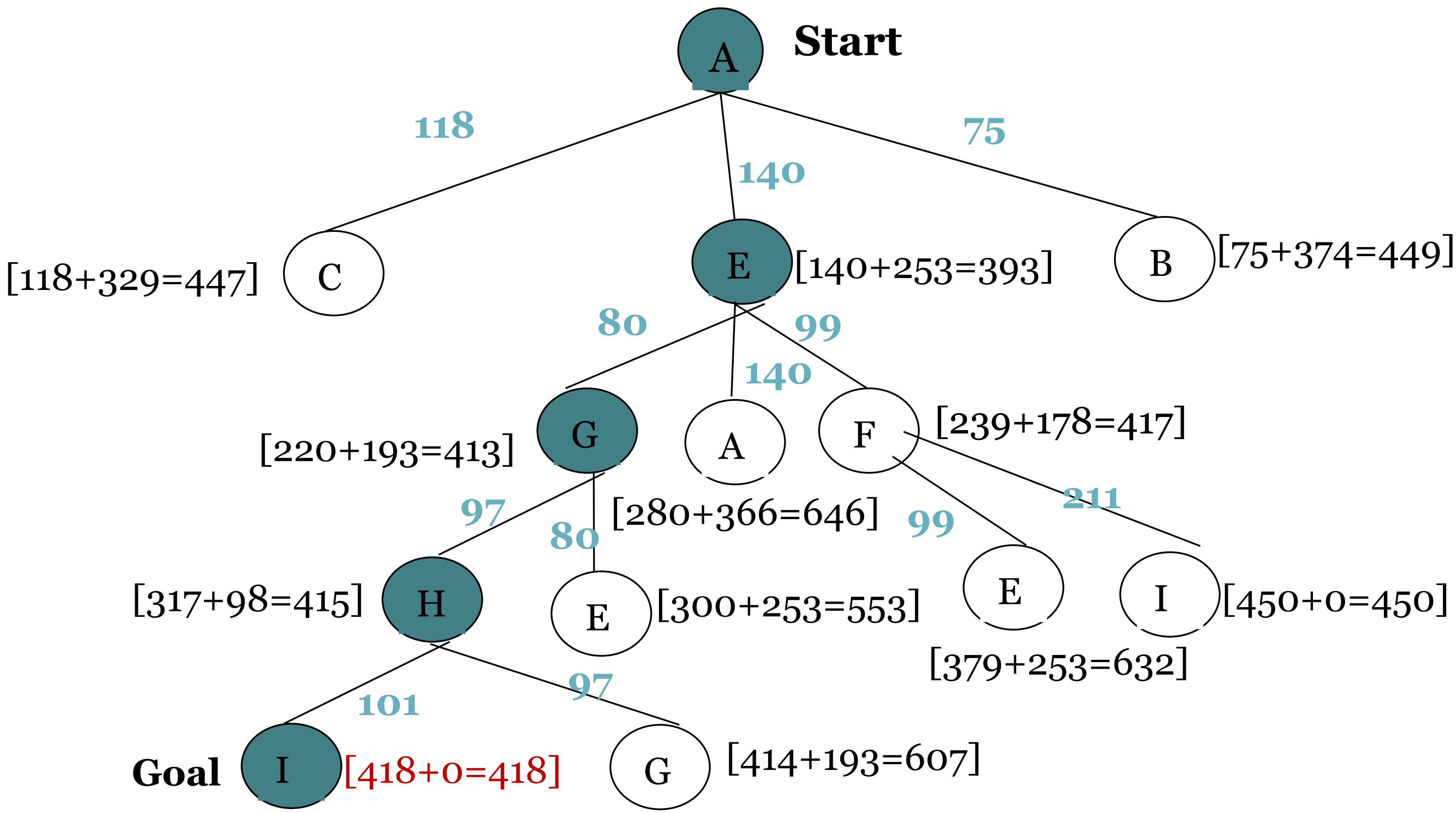
A* Search: Tree Search



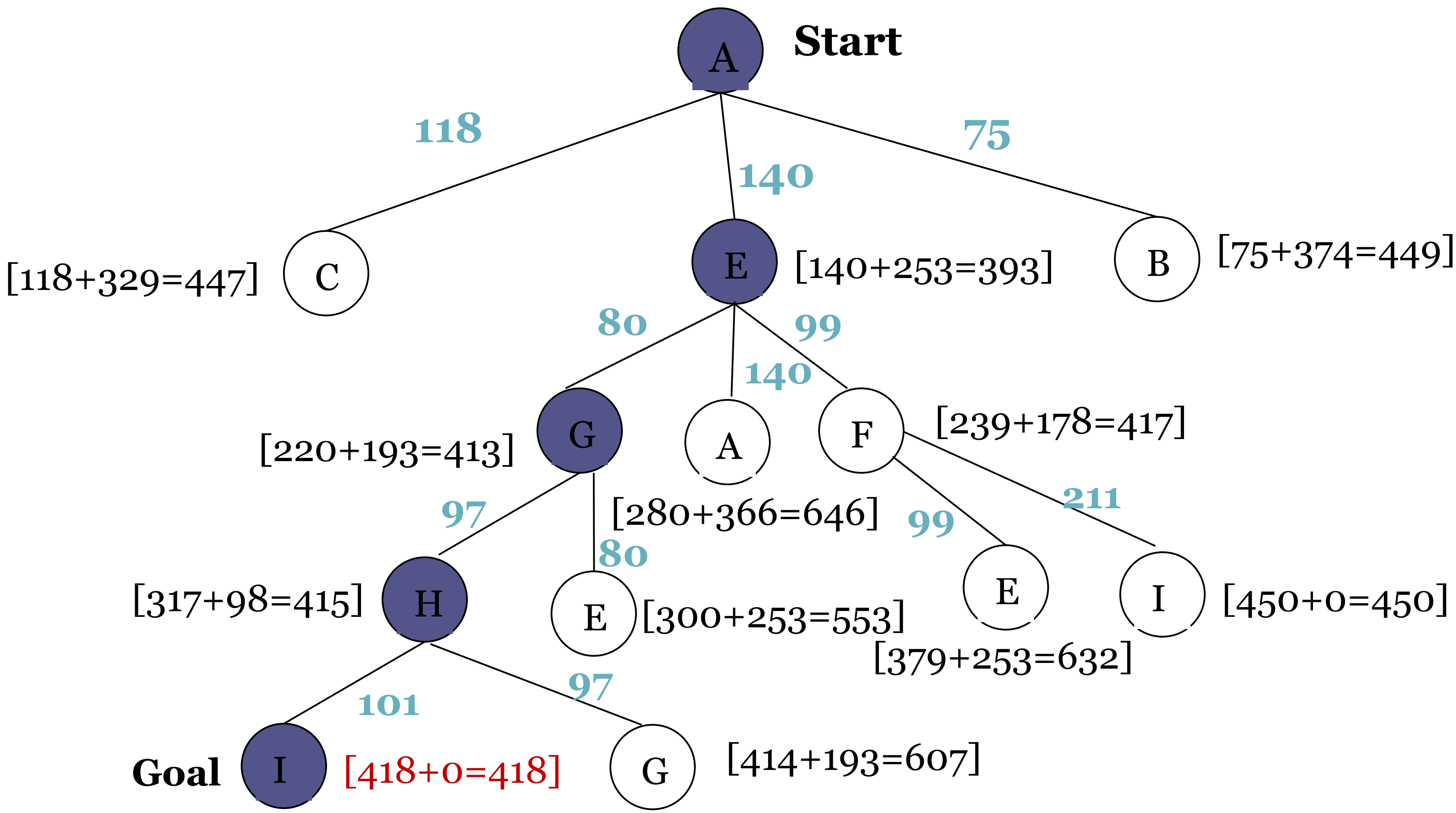
A* Search: Tree Search



A* Search: Tree Search

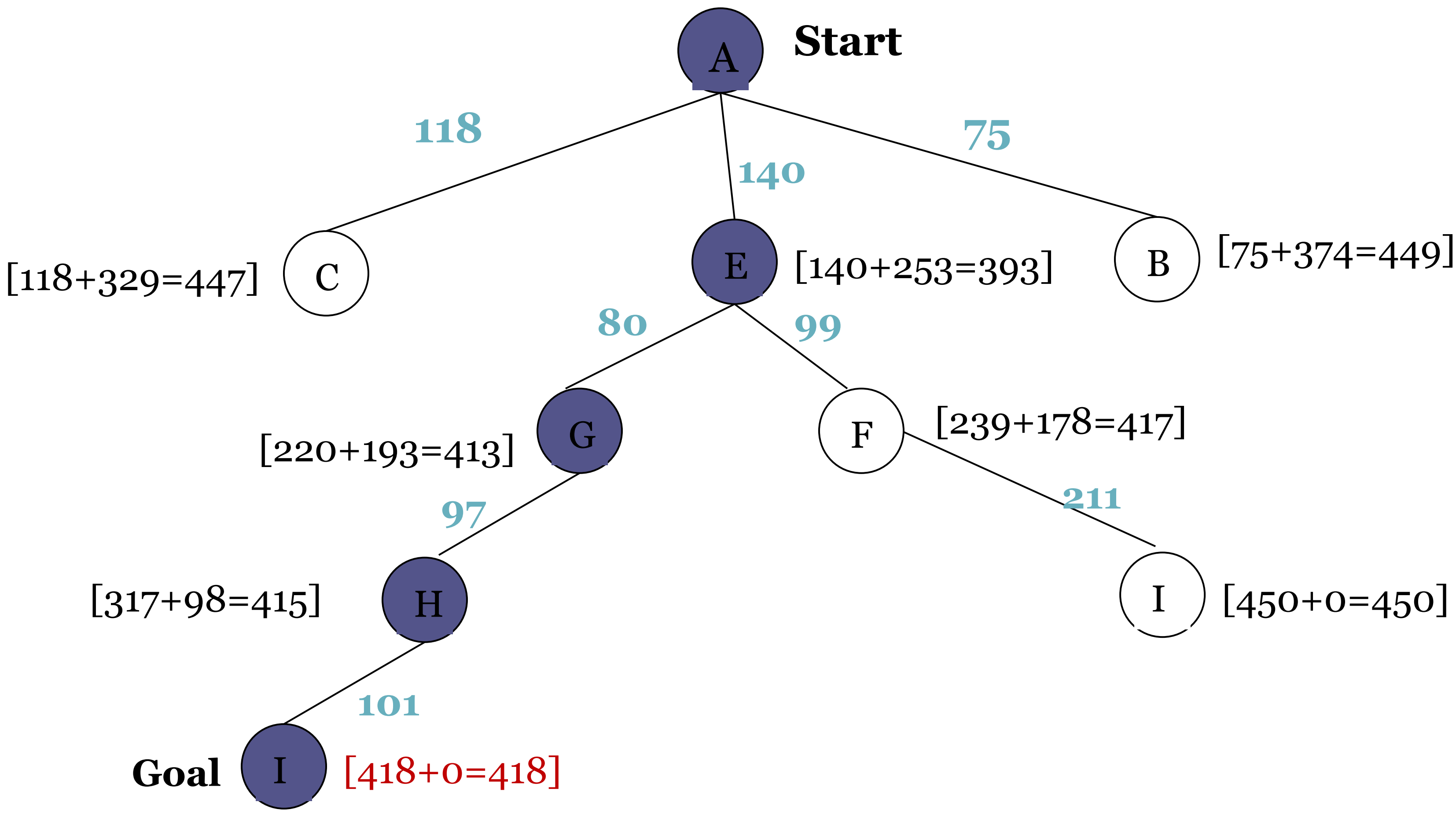


A* Search: Tree Search



dist(A-E-G-H-I) =140+80+97+101=418

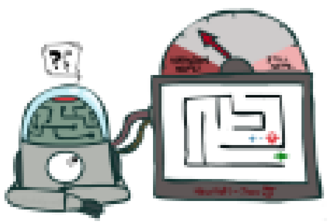
A* Search: Graph Search



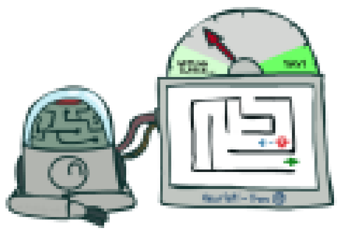
dist(A-E-G-H-I) =140+80+97+101=418

A* Admissible

- An admissible heuristic is a heuristic that is guaranteed to find the shortest path from the current state to the goal state. In other words, it is an optimal heuristic.
- Admissible heuristics are often used in pathfinding algorithms such as A*.



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe



Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

A* Admissible

- If h() overestimates the cost to reach the goal state
- Overestimate: not Admissible(If h() overestimates the cost to reach the goal state)
- Underestimate: Admissible

$h(n) \leq h^*(n)$ ∴ *Underestimation*

$h(n) \geq h^*(n)$ ∴ *Overestimation*

- $1200 > 1000$ i.e. $h(n) \geq h^*(n)$ ∴ Overestimation
- $800 < 1000$ i.e. $h(n) \leq h^*(n)$ ∴ Underestimation

A* is Admissible if

- we need to discover a solution to the problem, the estimated cost must be lower than or equal to the true cost of reaching the goal state.
- The heuristic function h(n) is called admissible if h(n) is never larger than h*(n), namely h(n) is always less or equal to true cheapest cost from n to the goal.

A* Admissible

- *The evaluation function in **A*** looks like this:*

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

f(n) = Actual cost + Estimated cost

here,

n = current node.

f(n) = evaluation function.

g(n) = the cost from the initial node to the current node.

h(n) = estimated cost from the current node to the goal state.

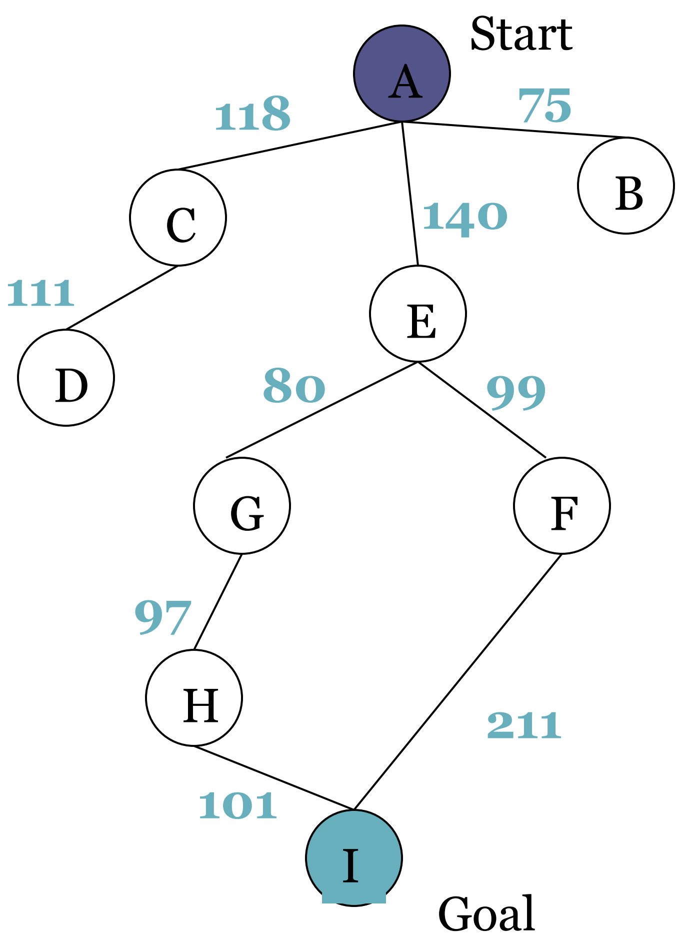
A* Admissible

- Now consider a fringe node n i.e. on an optimal solution path – example P in the example. If h(n) does not over estimate the cost of completing the solution path, then we know that

$$f(n) = g(n) + h(n) \leq C^*$$

A* Search: if h not admissible

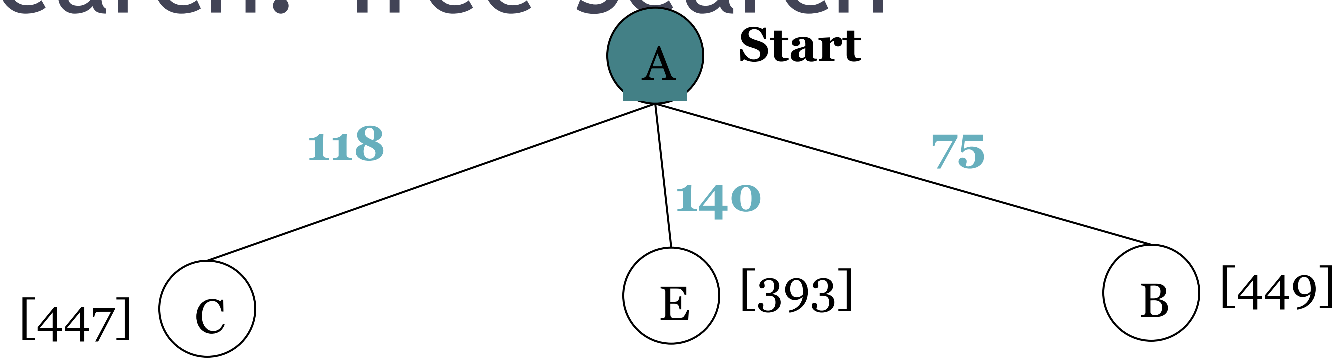
- Consistent Heuristic



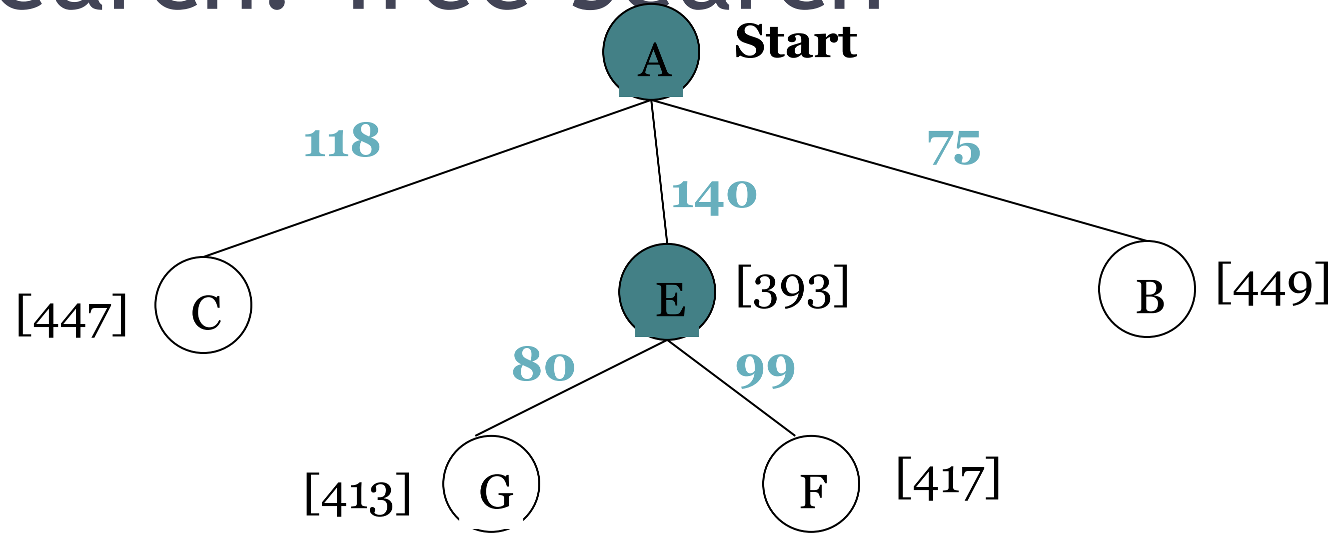
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	**138**
I	0

A* Search: Tree Search

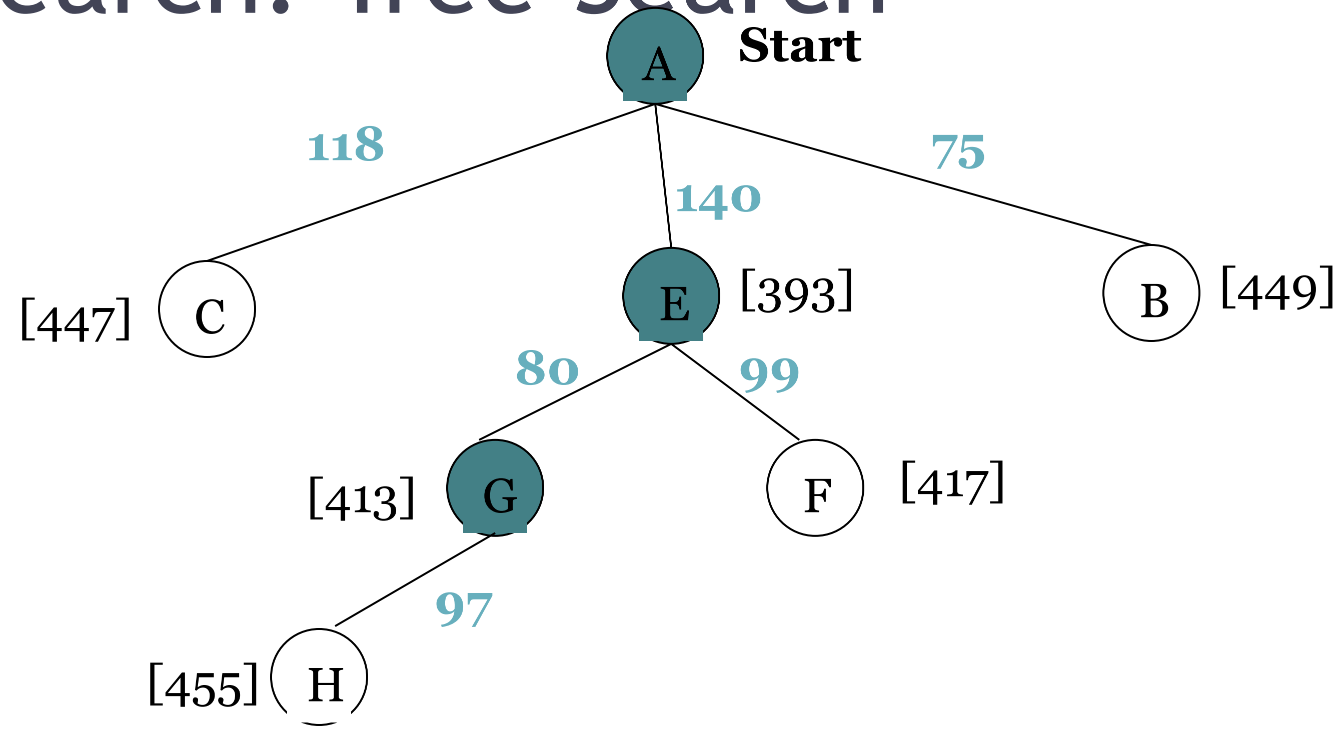
A* Search: Tree Search



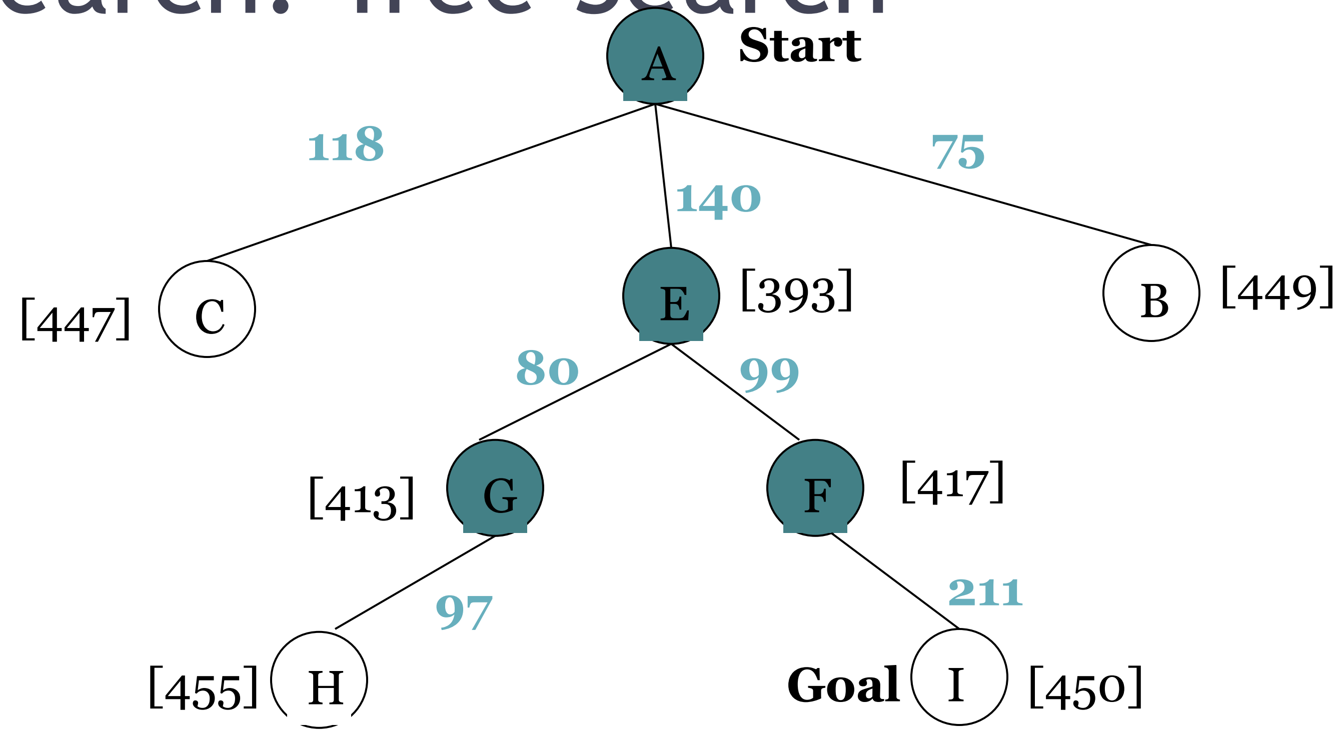
A* Search: Tree Search



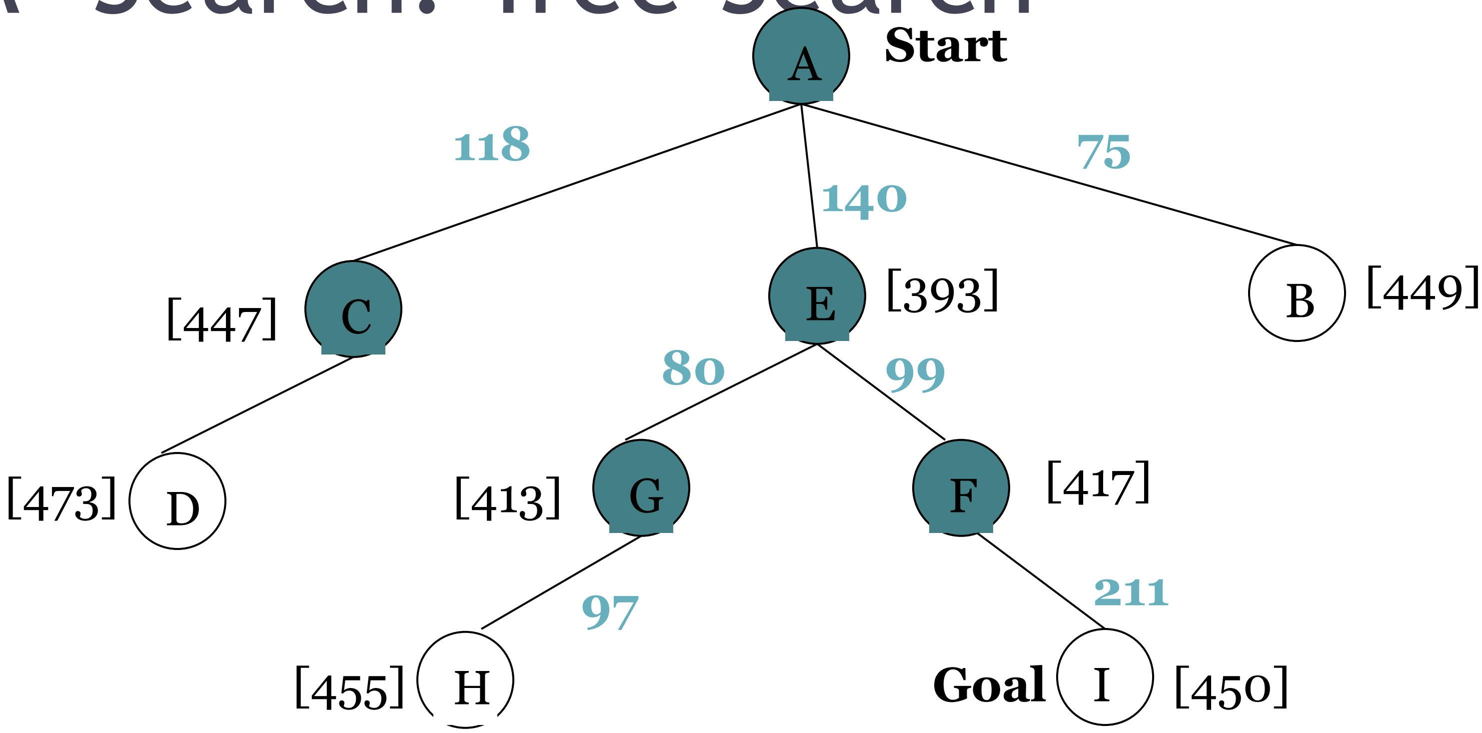
A* Search: Tree Search



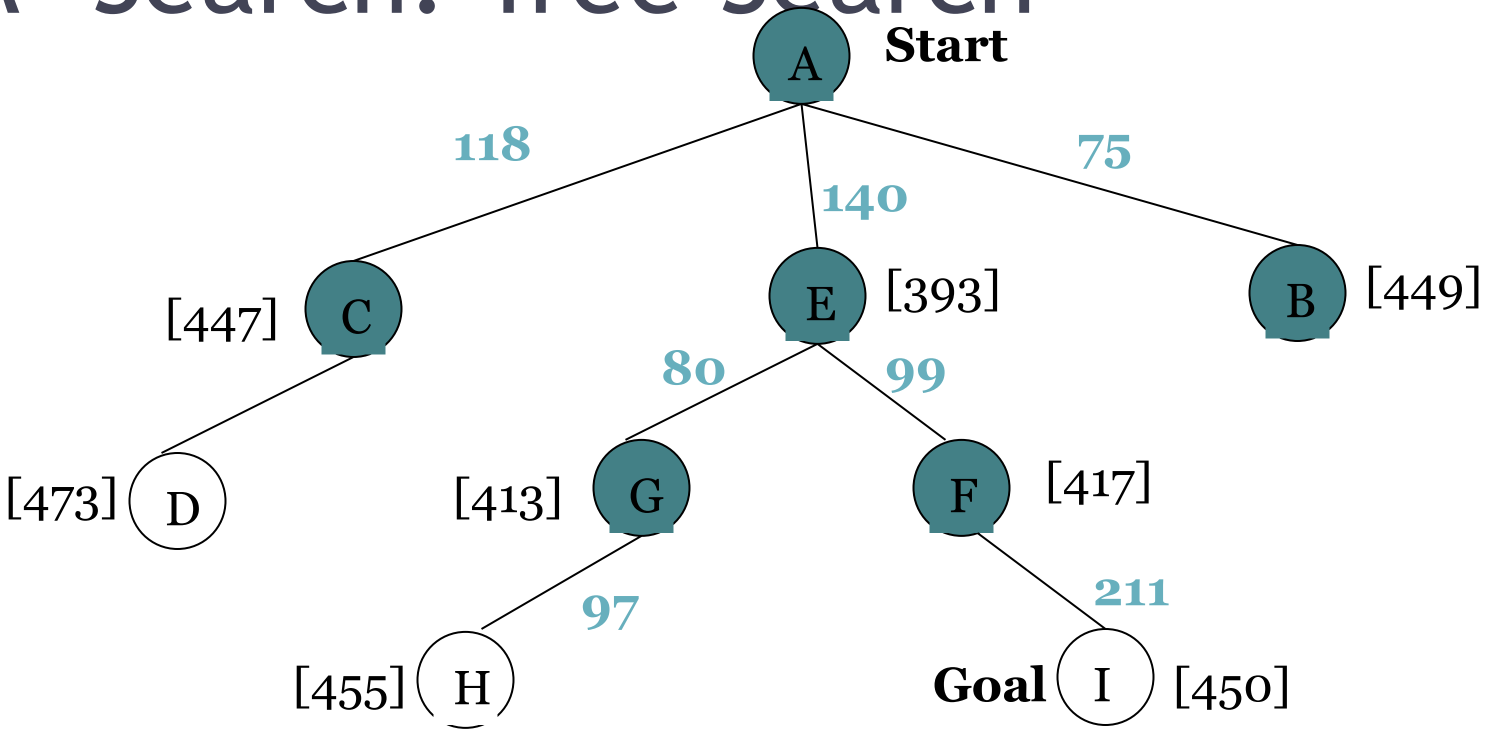
A* Search: Tree Search



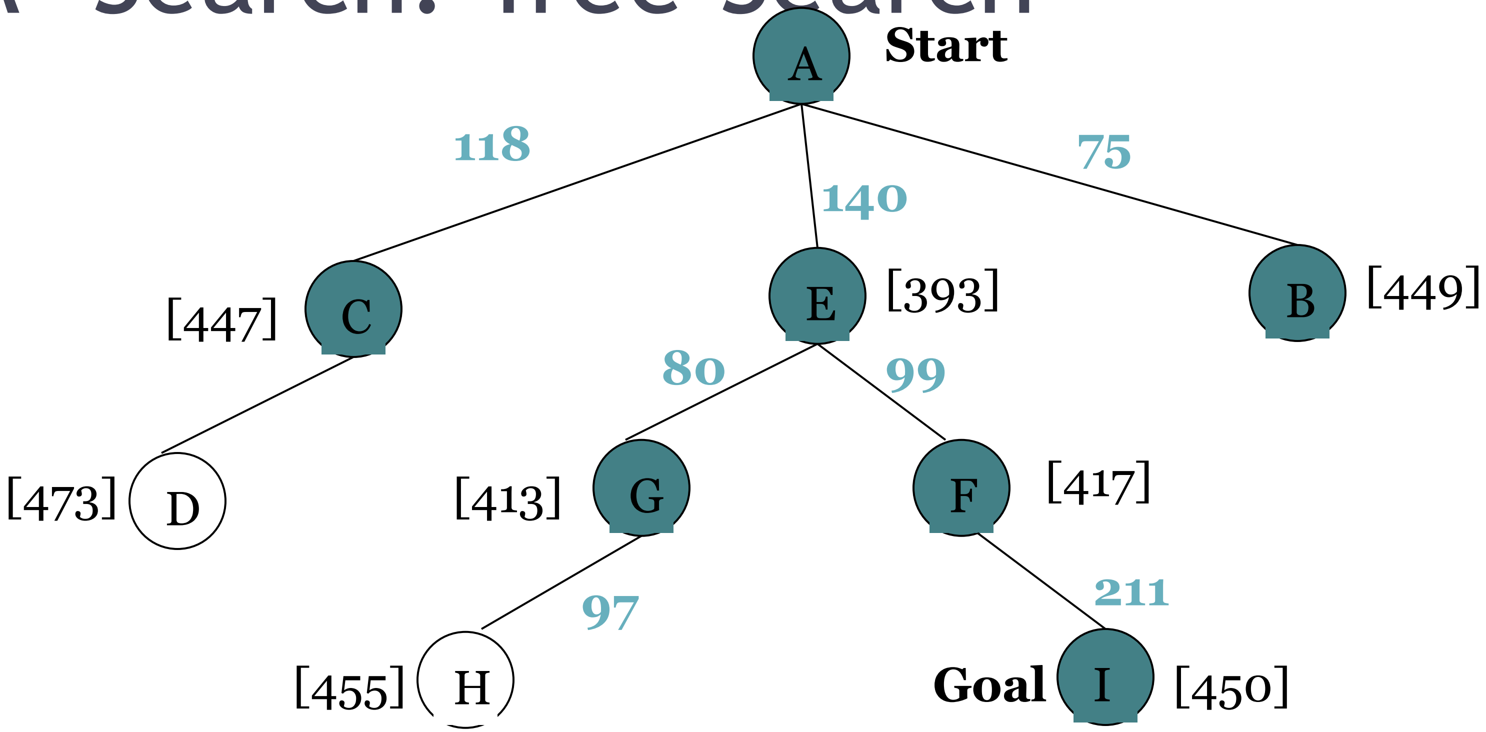
A* Search: Tree Search



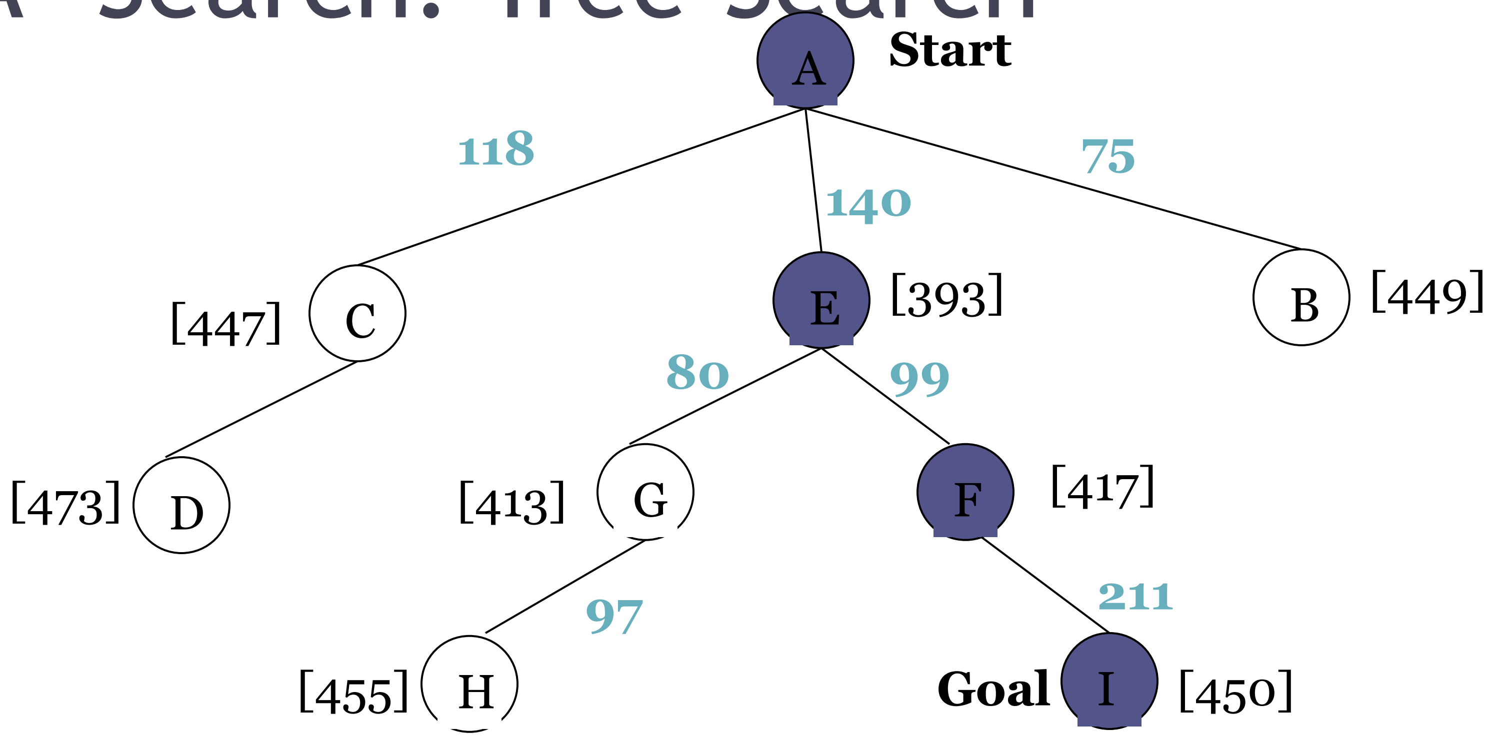
A* Search: Tree Search



A* Search: Tree Search



A* Search: Tree Search



A* not optimal !!!

When heuristic h is not admissible

dist(A-E-F-I) =140+99+211=45