Informed Search - II

A* Search Algorithm

Previous Lecture

- Heuristics
- Greedy Search

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

A* Search

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

A* (A Star)

- Although Greedy Search can considerably cut the search time (efficient), it is neither optimal nor complete.
- Uniform Cost Search minimizes the cost $g(n)$ from the initial state to n. UCS is optimal and complete but not efficient.

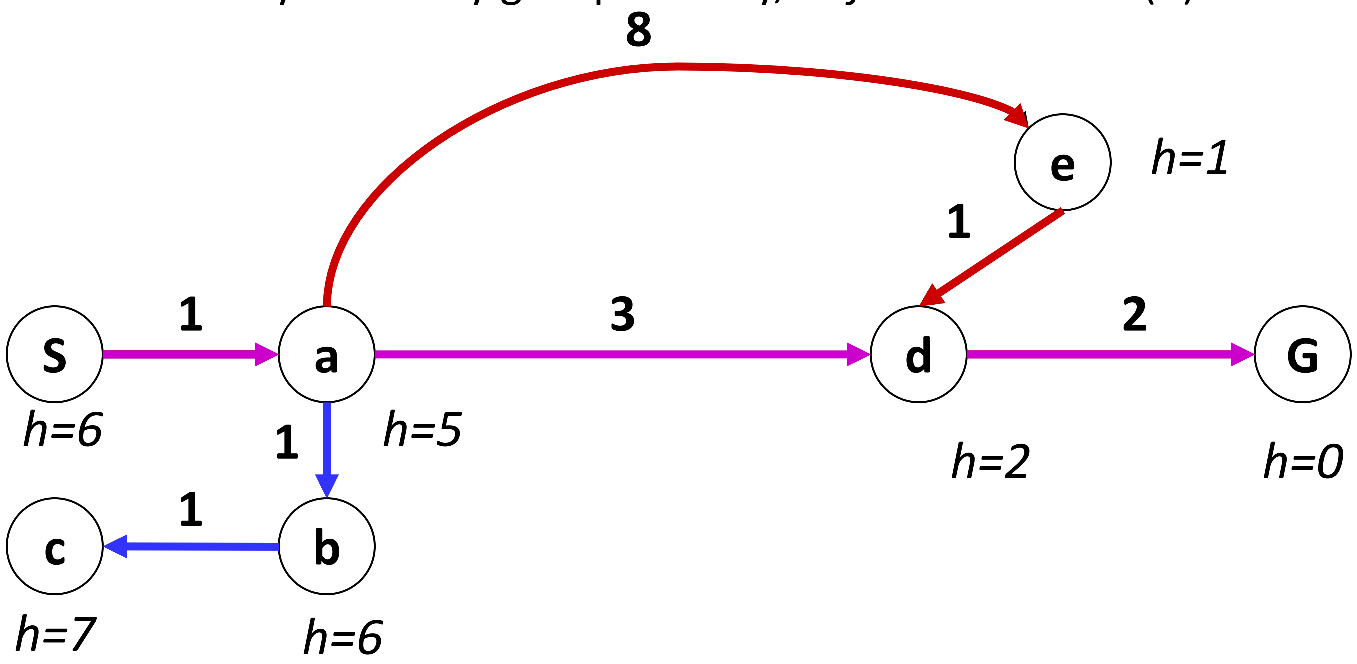
- New Strategy: Combine Greedy Search and UCS to get an efficient algorithm which is complete and optimal.

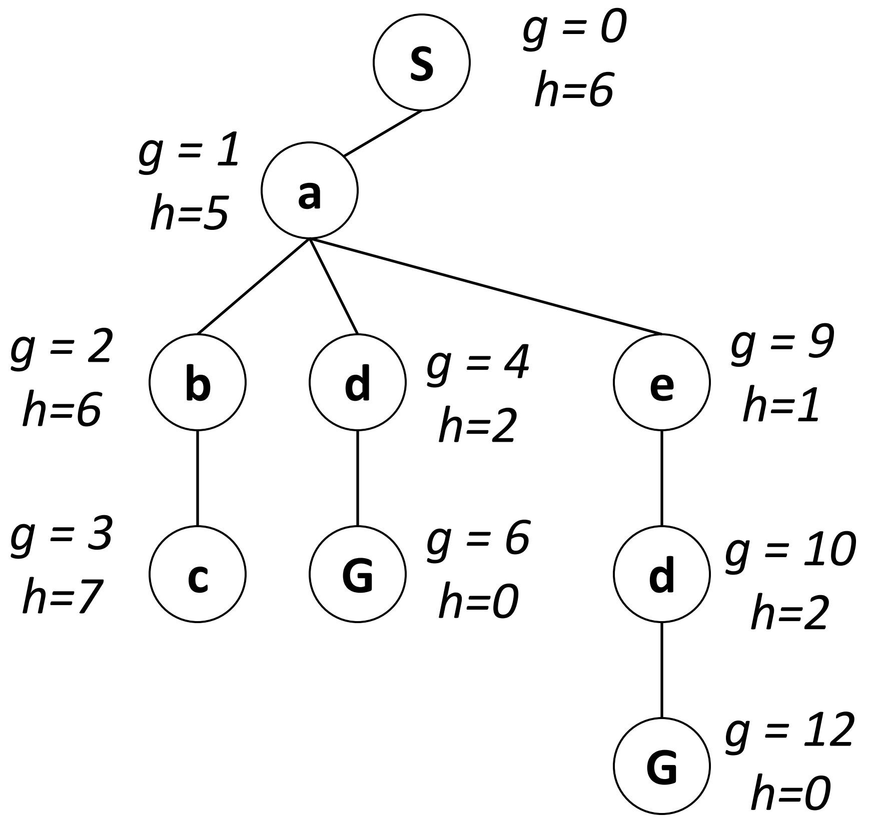
- A* uses a heuristic function which combines g(n) and h(n):

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

Combining UCS and Greedy

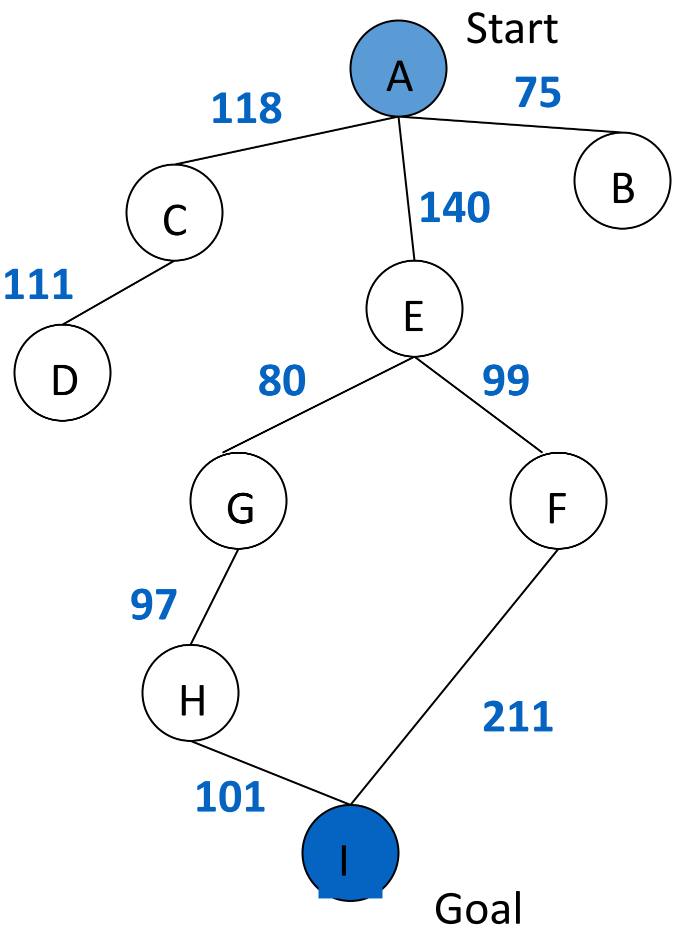
- Uniform-cost orders by path cost, or *backward cost* g(n)
- Greedy orders by goal proximity, or *forward cost* h(n)





- A* Search orders by the sum: f(n) = g(n) + h(n)

A* Search



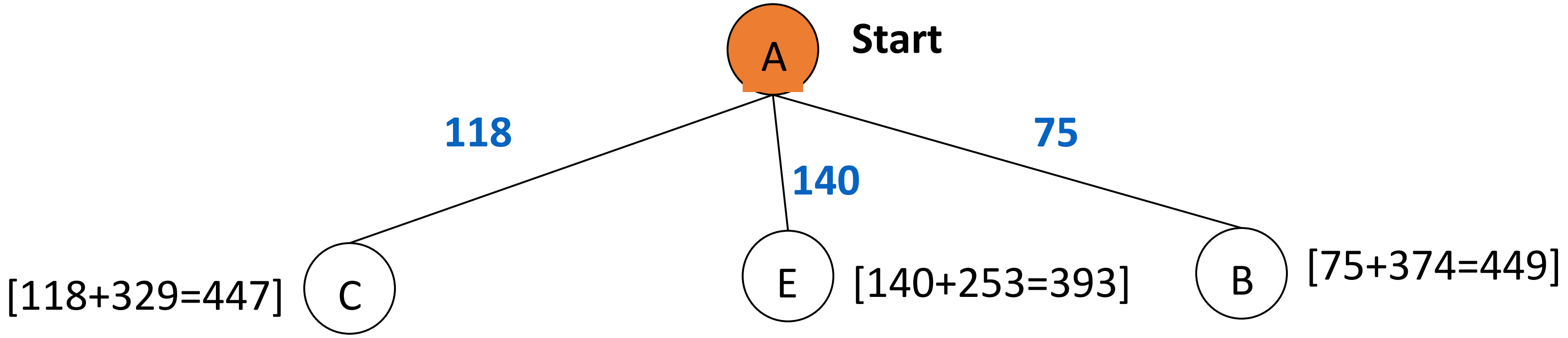
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

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

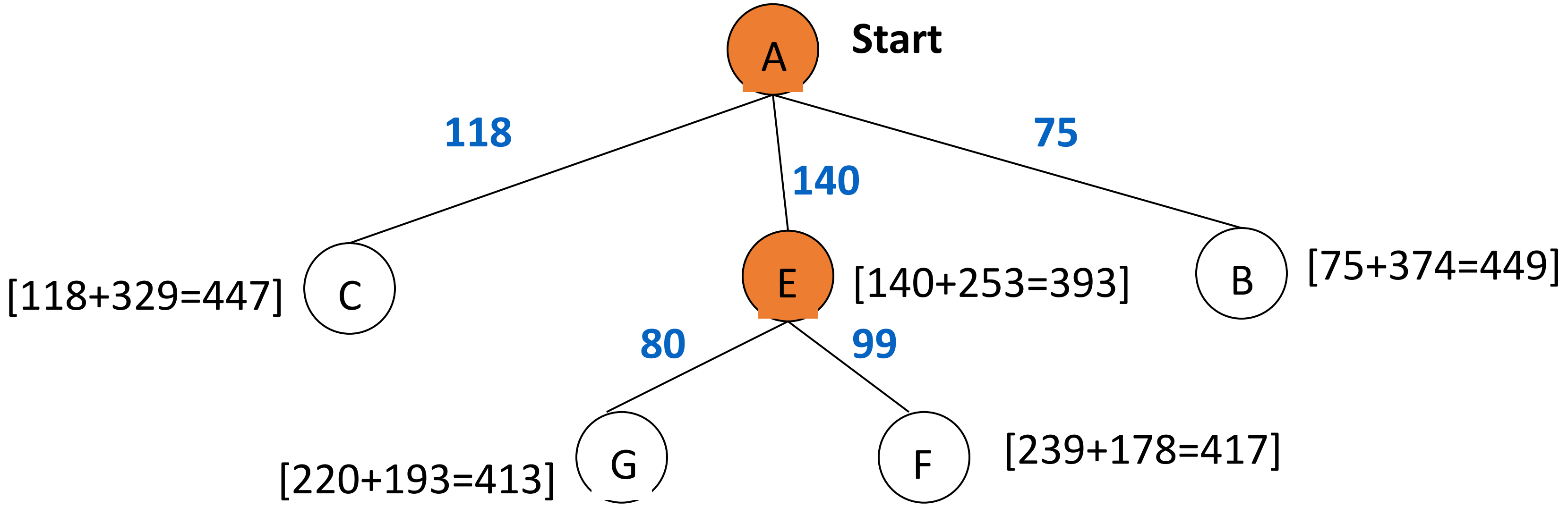
g(n): is the exact cost to reach node *n* from the initial state.

A* Search: Tree Search

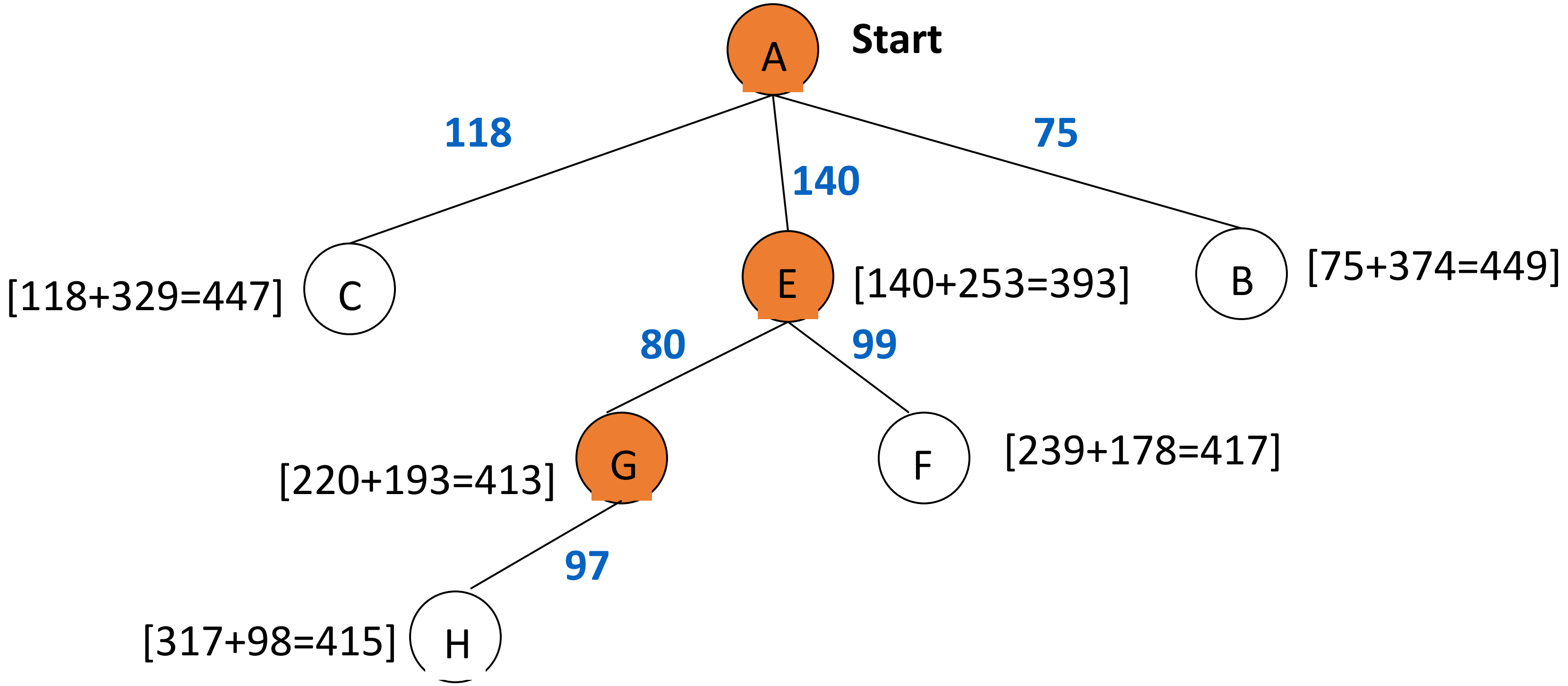
A* Search: Tree Search



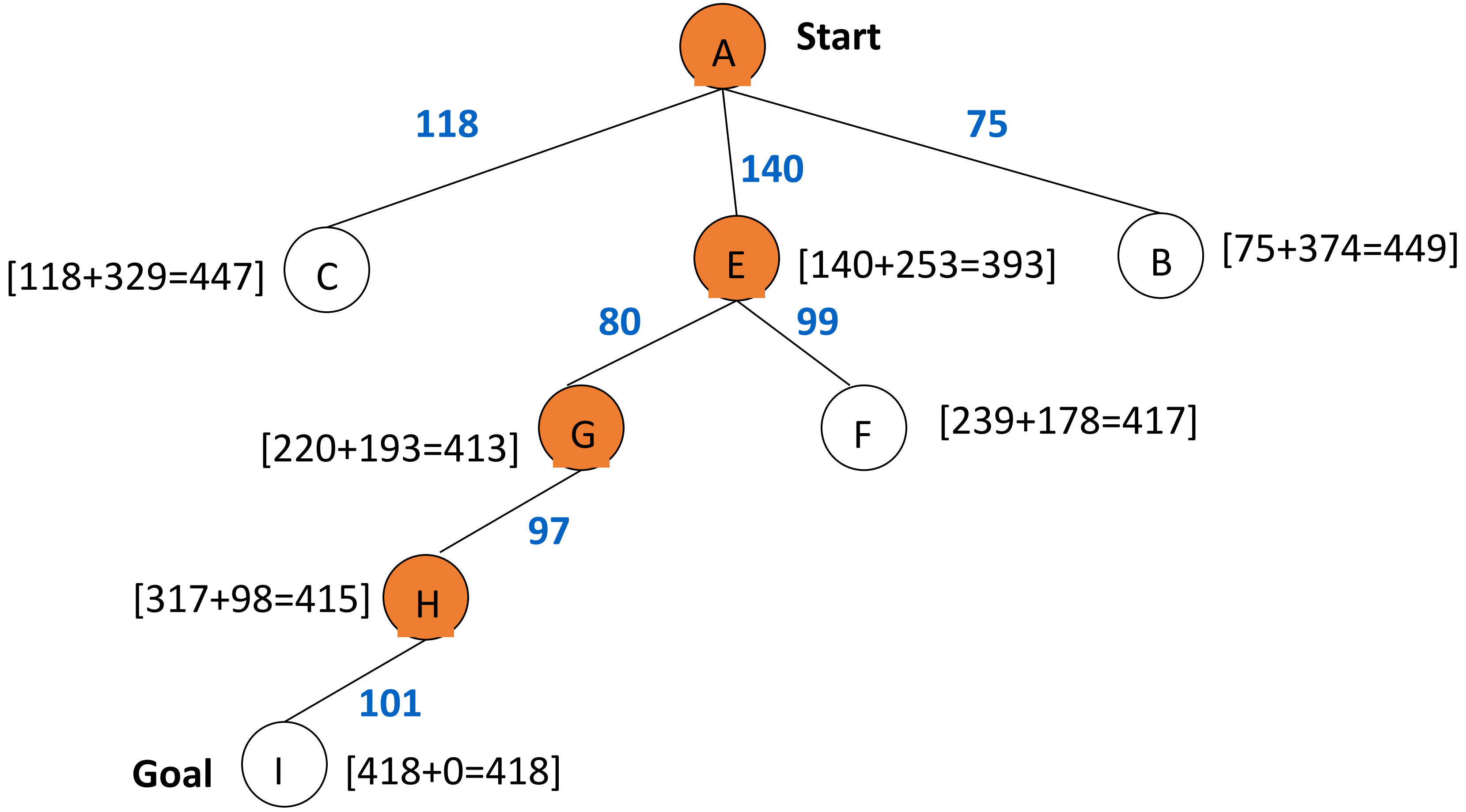
A* Search: Tree Search



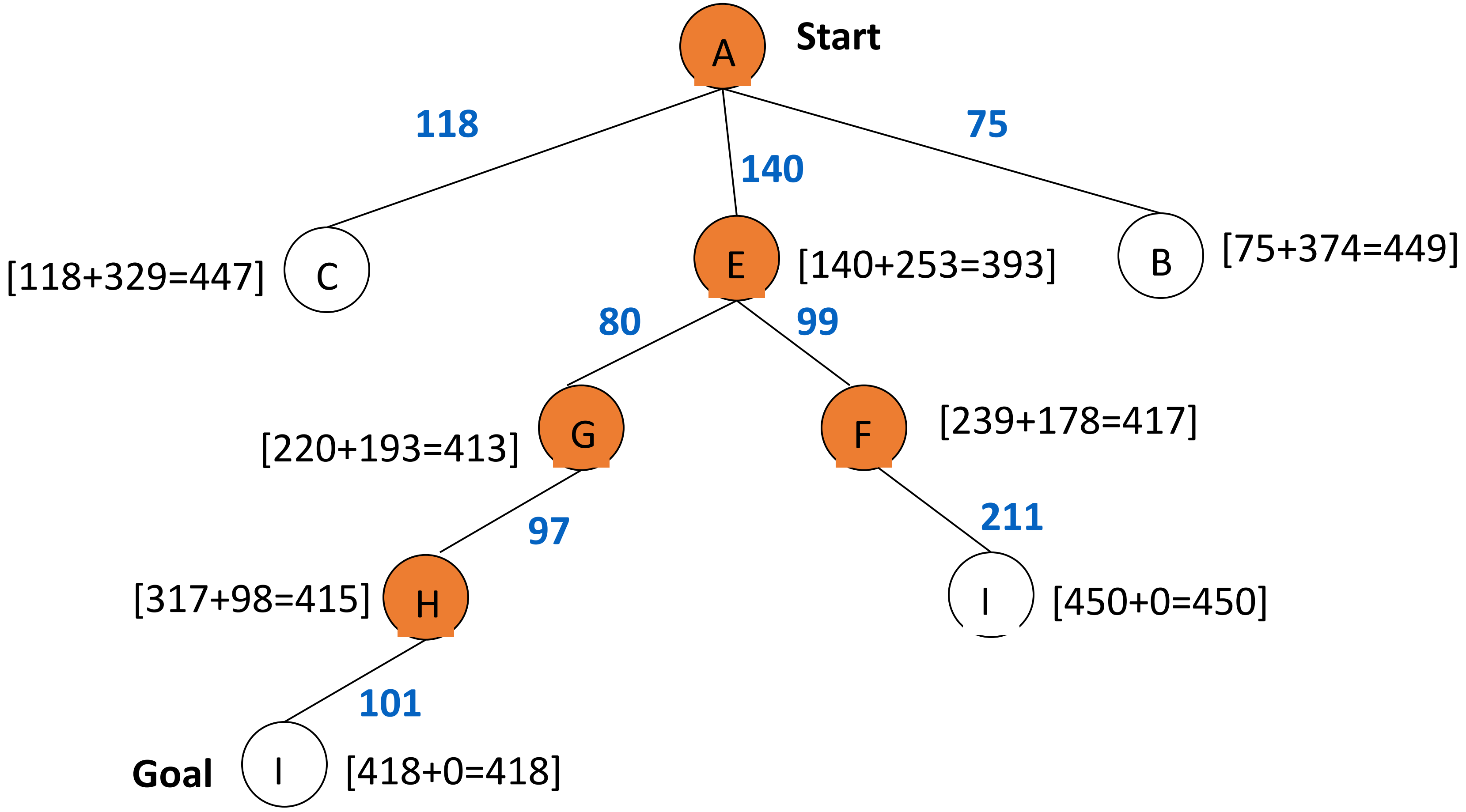
A* Search: Tree Search



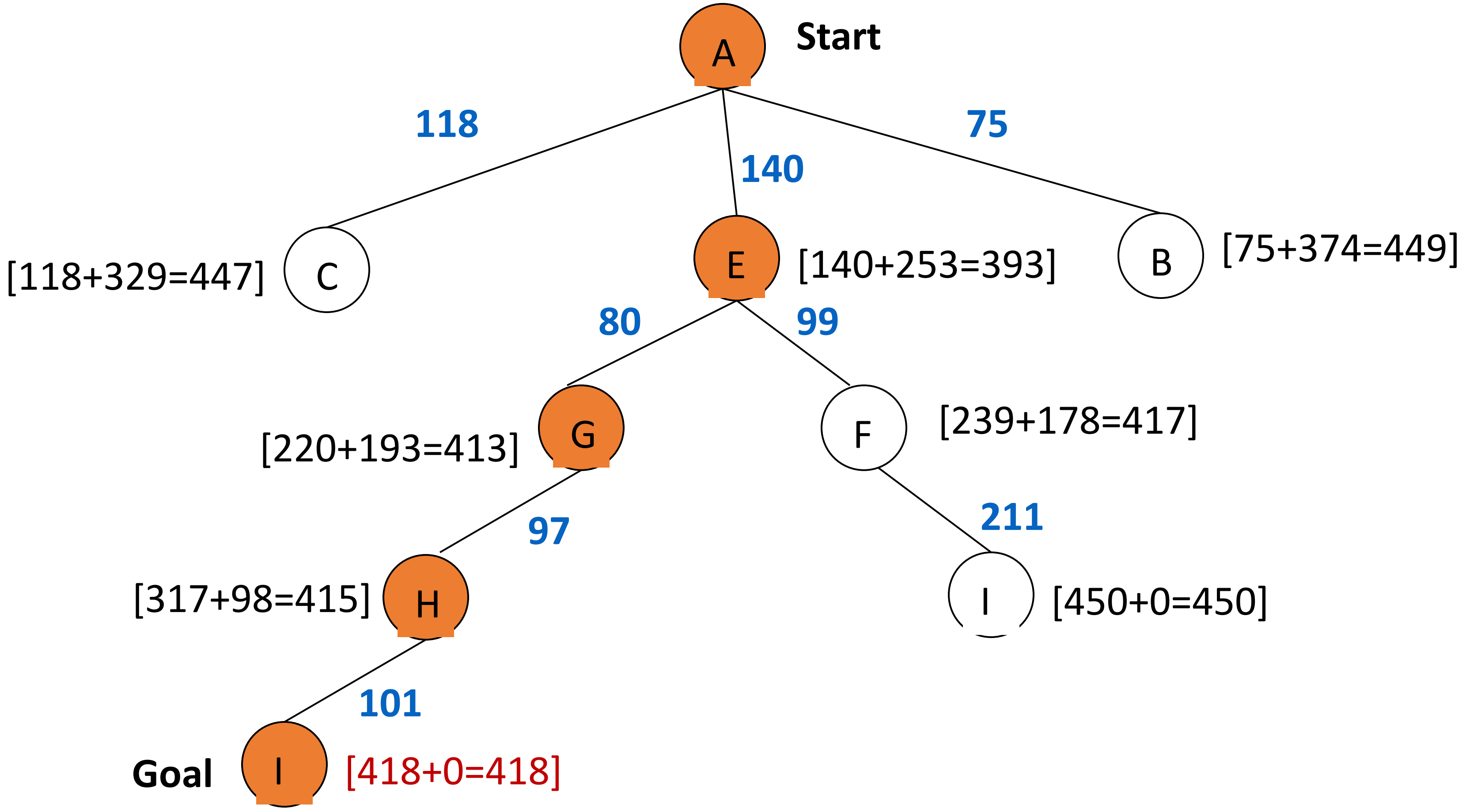
A* Search: Tree Search



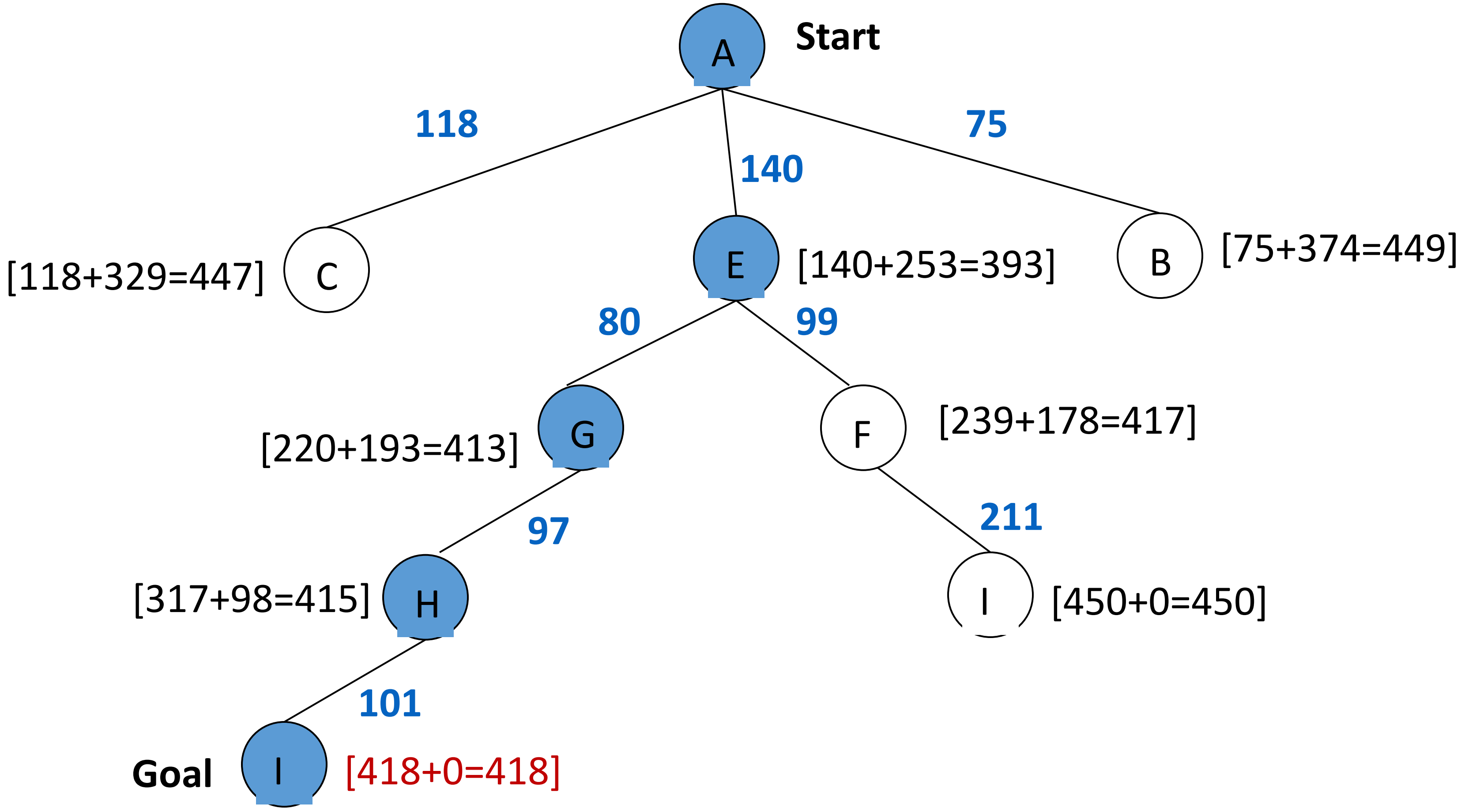
A* Search: Tree Search



A* Search: Tree Search

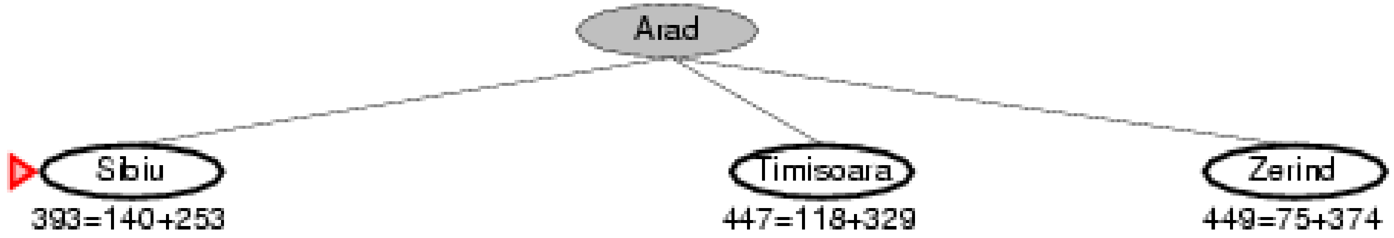


A* Search: Tree Search

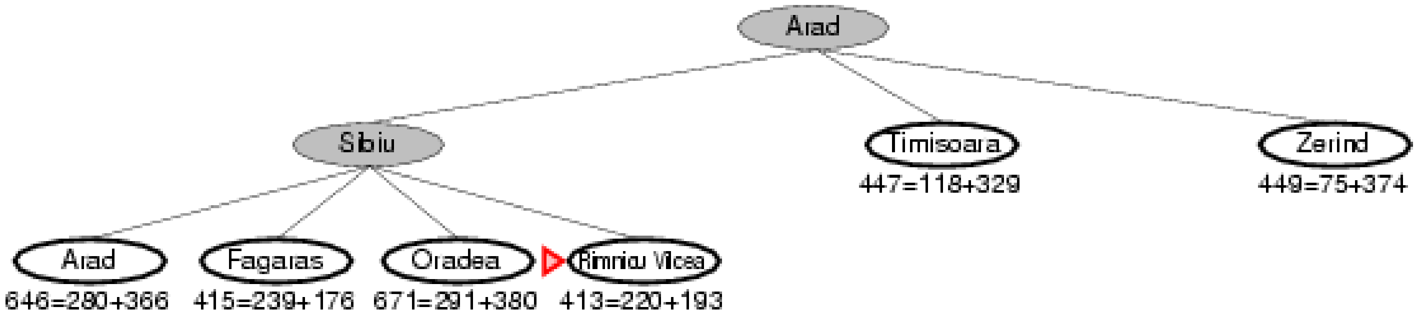


dist(A-E-G-H-I) =140+80+97+101=418

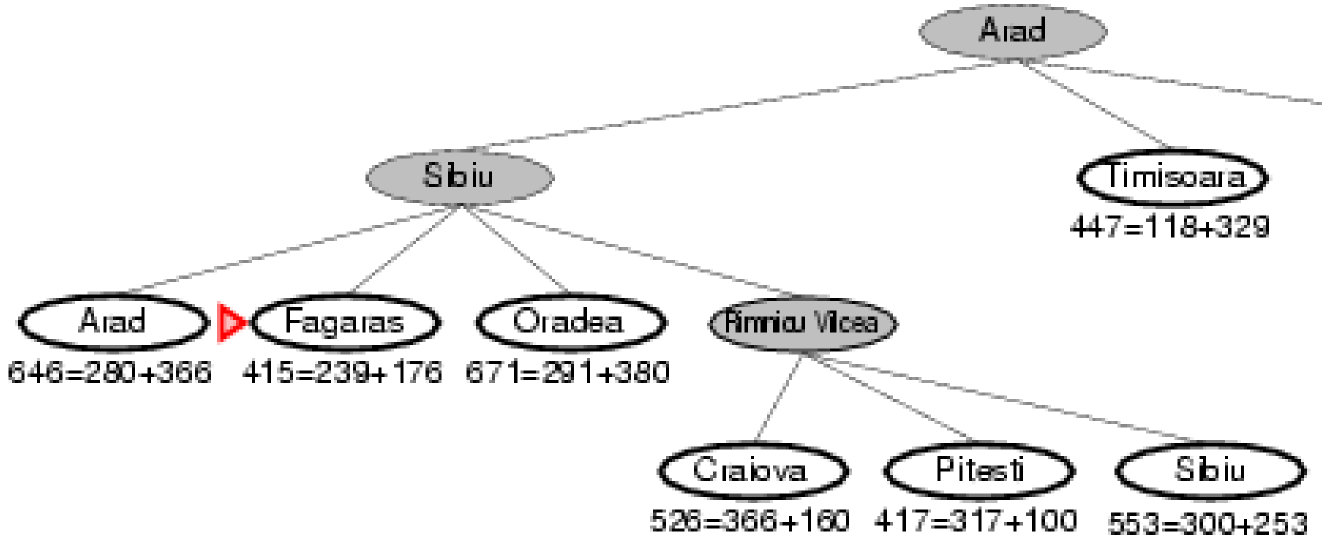
A* search example



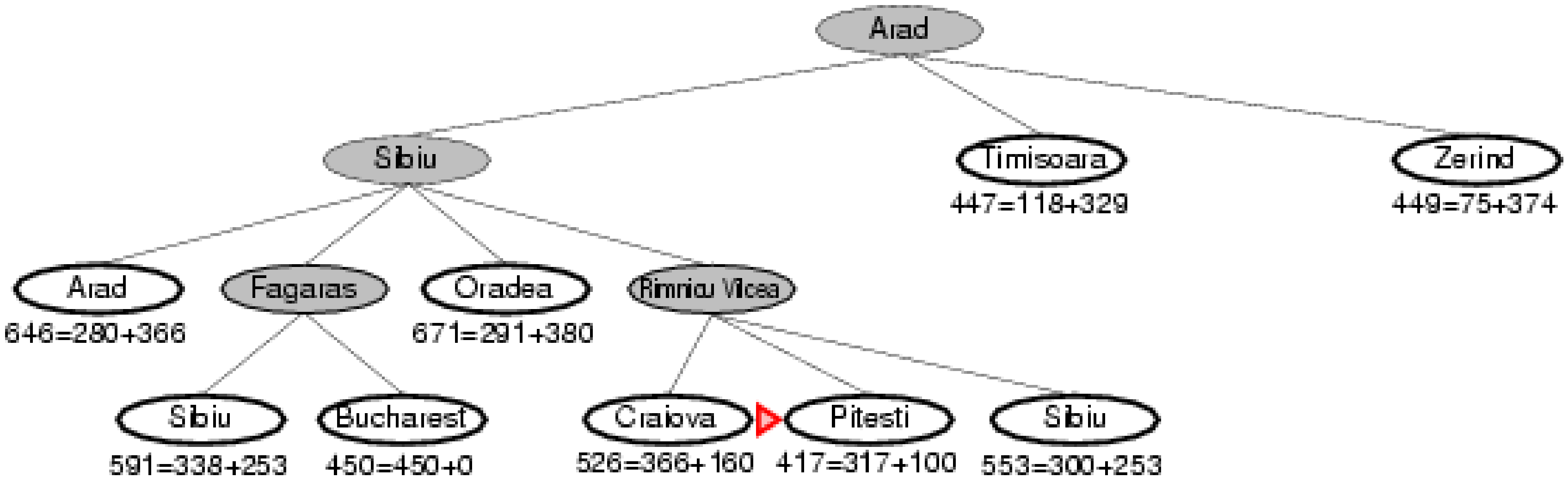
A* search example



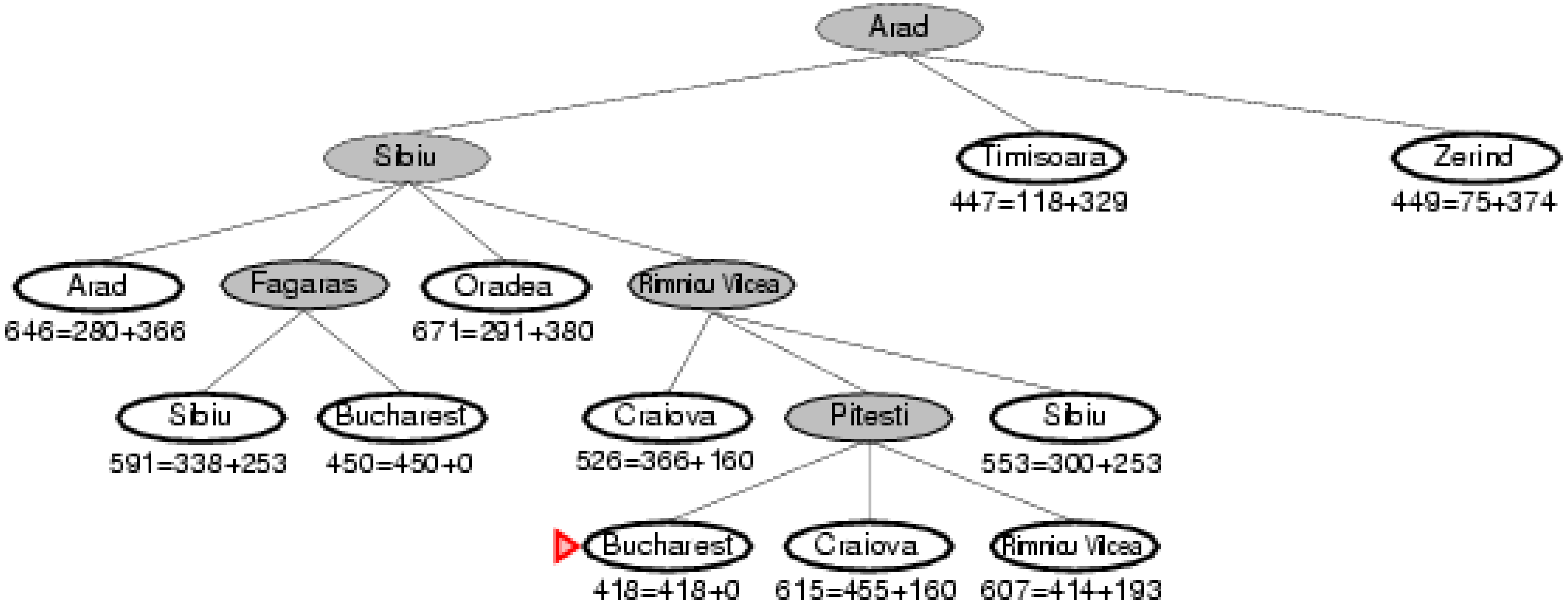
A* search example



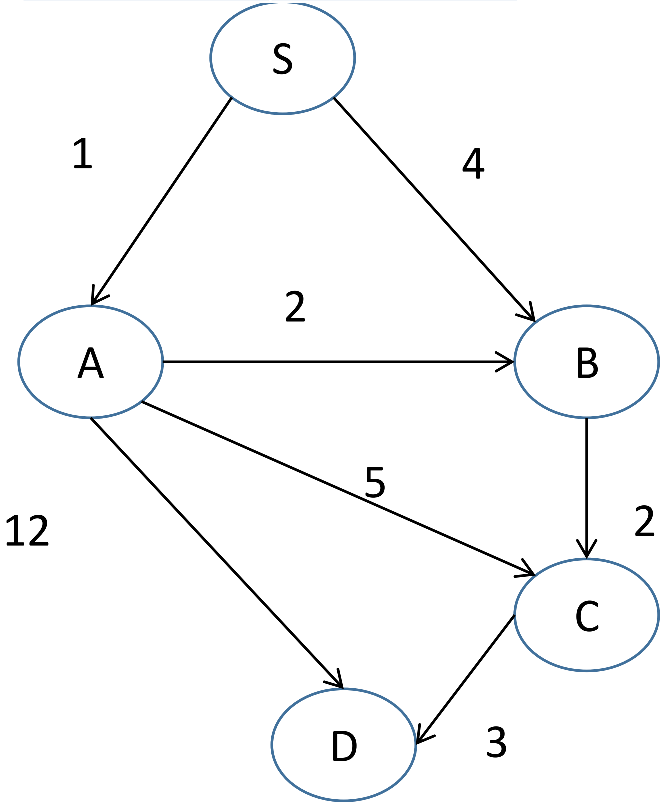
A^* search example



A* search example

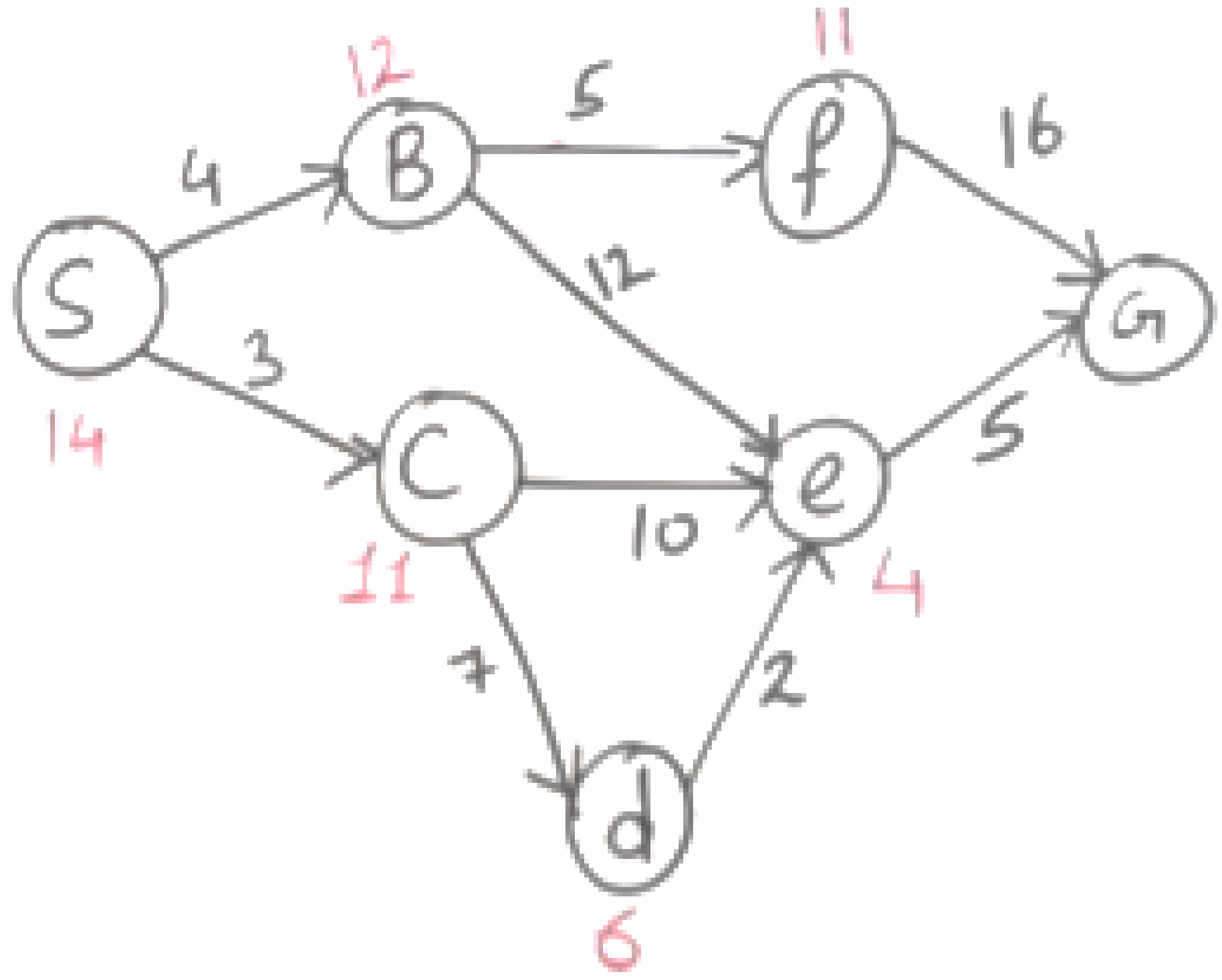


Heuristic Value	
S	7
A	6
B	2
C	1
D	0



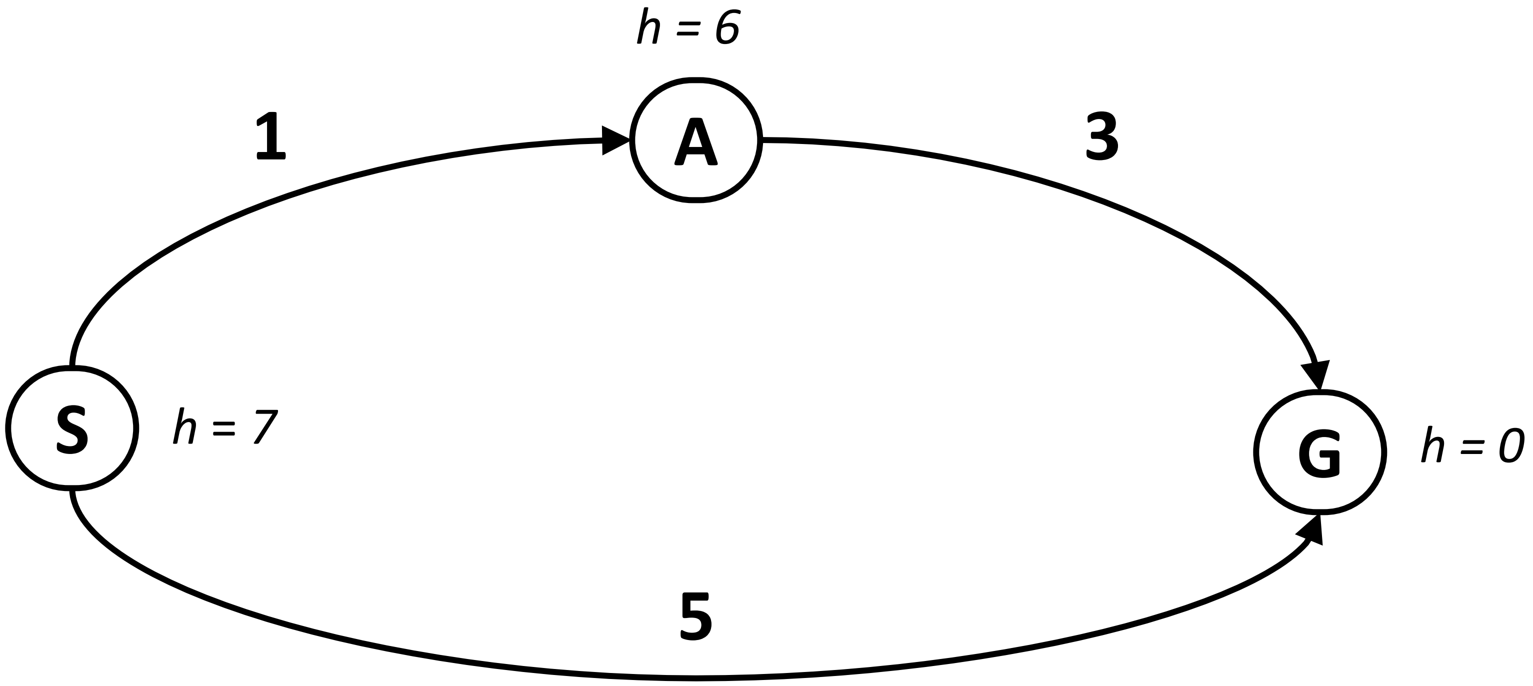
$f(n) = g(n)+h(n)$

	FRONTIER	EXPAND	EXPLORED
1	(S,0+7=7)	S	Empty
2	(S-A,1+6=7)(S-B,4+2=6)	B	S
3	(S-A,7)(S-B-C,6+1=7)	A	S,B
4	~~(S-A-B,3+2=5)~~(S-A-C,6+1=7)(S-A-D,13+0=13)(S-B-C,7)	C	S,B,A
5	(S-A-C-D,9+0=9)	D	S,B,A,C
6	OR (S-B-C-D,9+0=9)		
7			



S
B
C
d
e
f
G
4
3
5
12
10
7
2
16
5
14
12
11
6
4
11

Is A* Optimal?



- What went wrong?
  - Actual bad goal cost < estimated good goal cost
  - We need estimates to be less than actual costs!

Admissible Heuristic Functions

- A heuristic $h(n)$ is admissible if $h(n)$ never overestimates the cost to reach the goal.
  - Admissible heuristics are "optimistic" approximation

- Mathematically *h* is admissible if

For all n, $h(n) \leq C(n)$

where

- *n* is the node, *h* is heuristic
- *h(n)* is the cost indicated by *h* to reach goal from *n*
- *C(n)* is actual cost to reach a goal from *n*

Admissible heuristics Examples

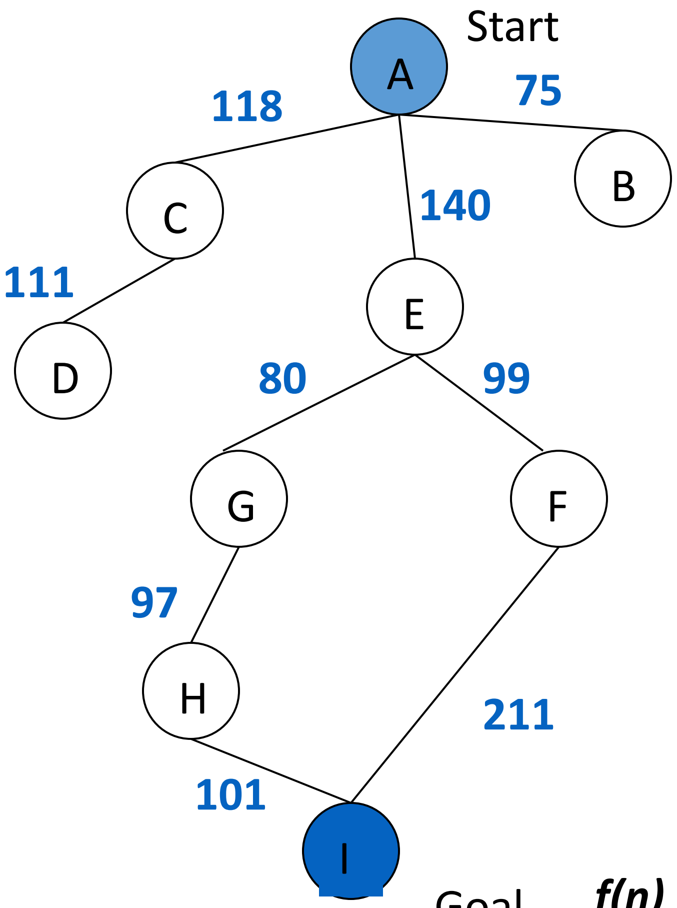
- $h_{SLD}(n)$ Straight line distance heuristic
  - Because never overestimate the actual road distance.
- Number of misplaced tiles in n-puzzle problem

Admissible Heuristic Functions

- Evaluation function $f(n) = g(n) + h(n)$ is admissible if $h(n)$ is admissible.
- However, $g(n)$ is the exact cost to reach node *n* from the initial state.
- Therefore, $f(n)$ never over-estimate the true cost to reach the goal state through node *n*.

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

A* Search: if h not admissible

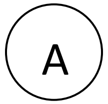


State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	**138**
I	0

f(n) = g(n) + h **(*n*) – (H-I) Overestimated**

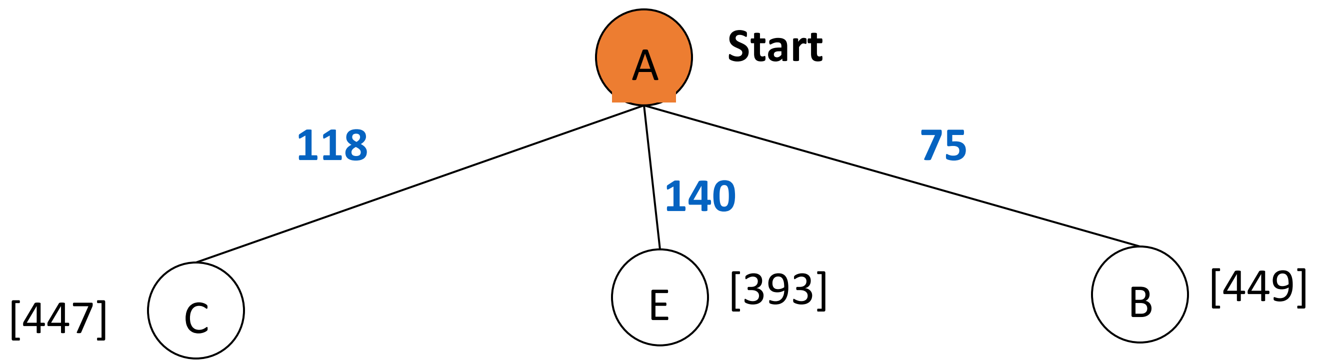
g(n): is the exact cost to reach node *n* from the initial state.

A* Search: Tree Search

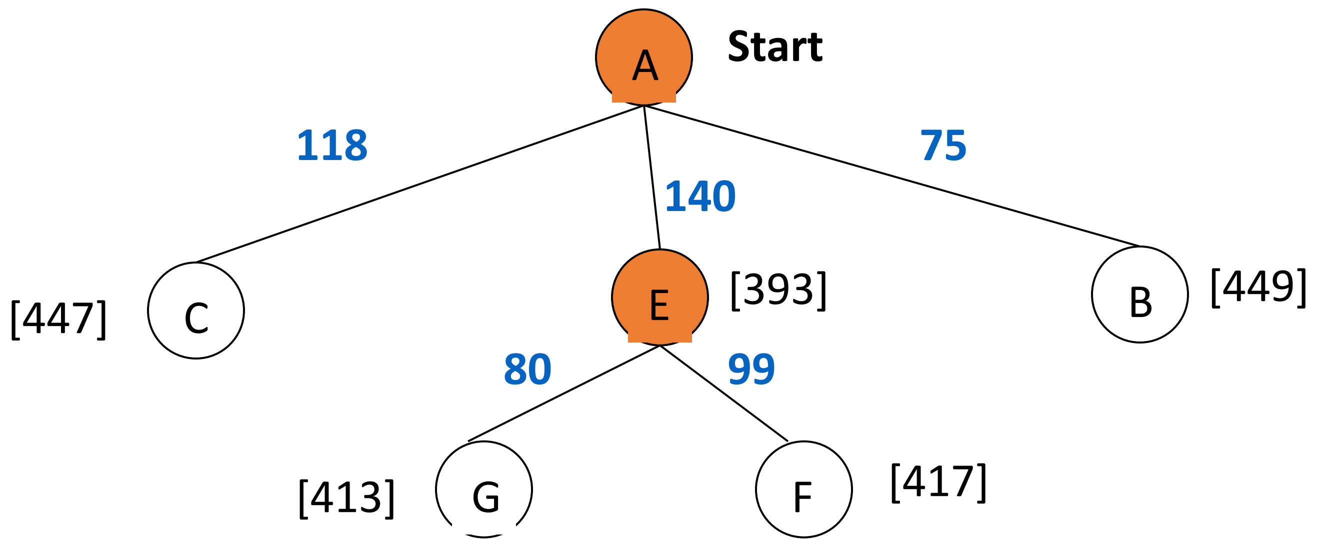


Start

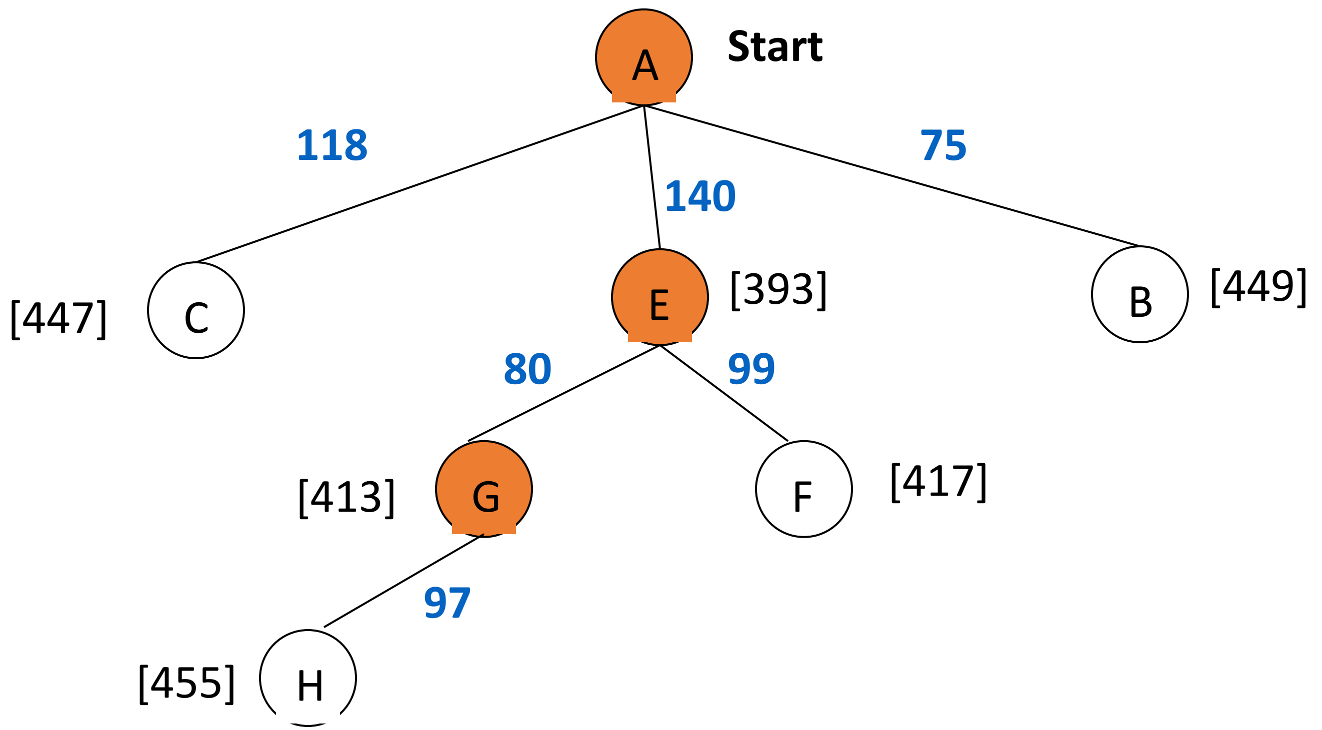
A* Search: Tree Search



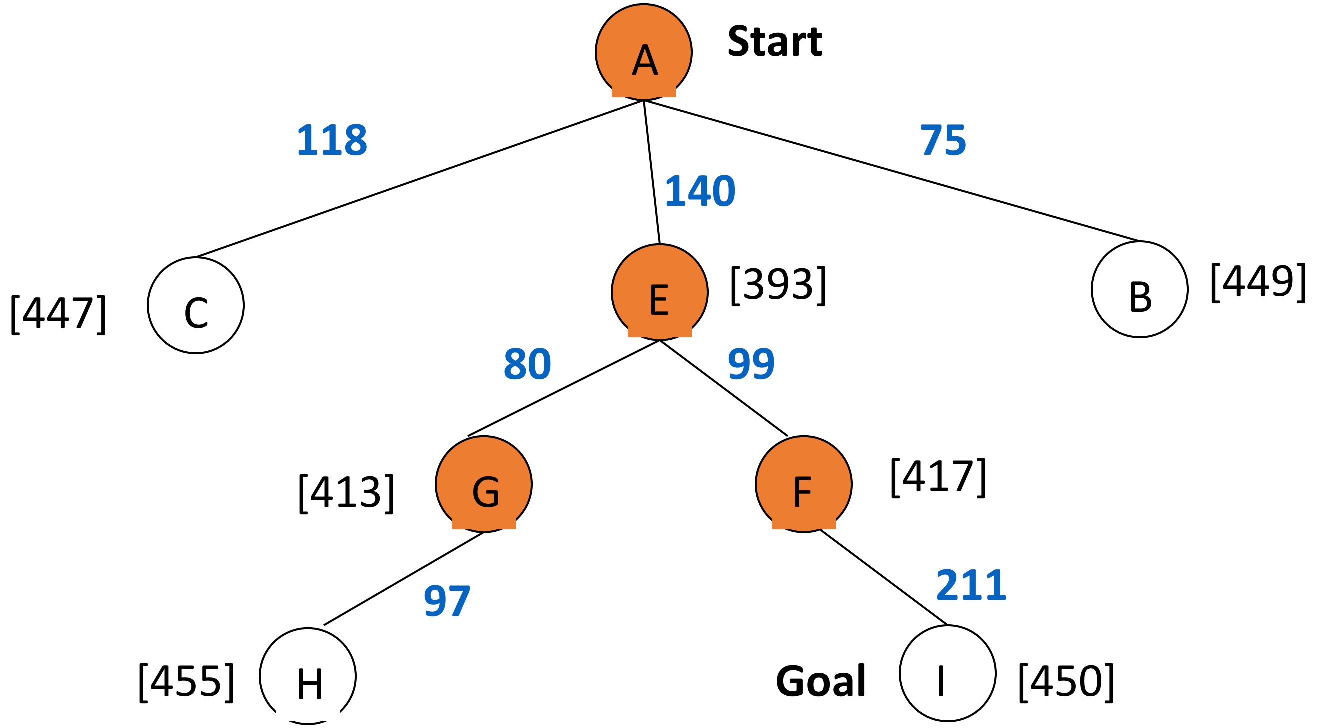
A* Search: Tree Search



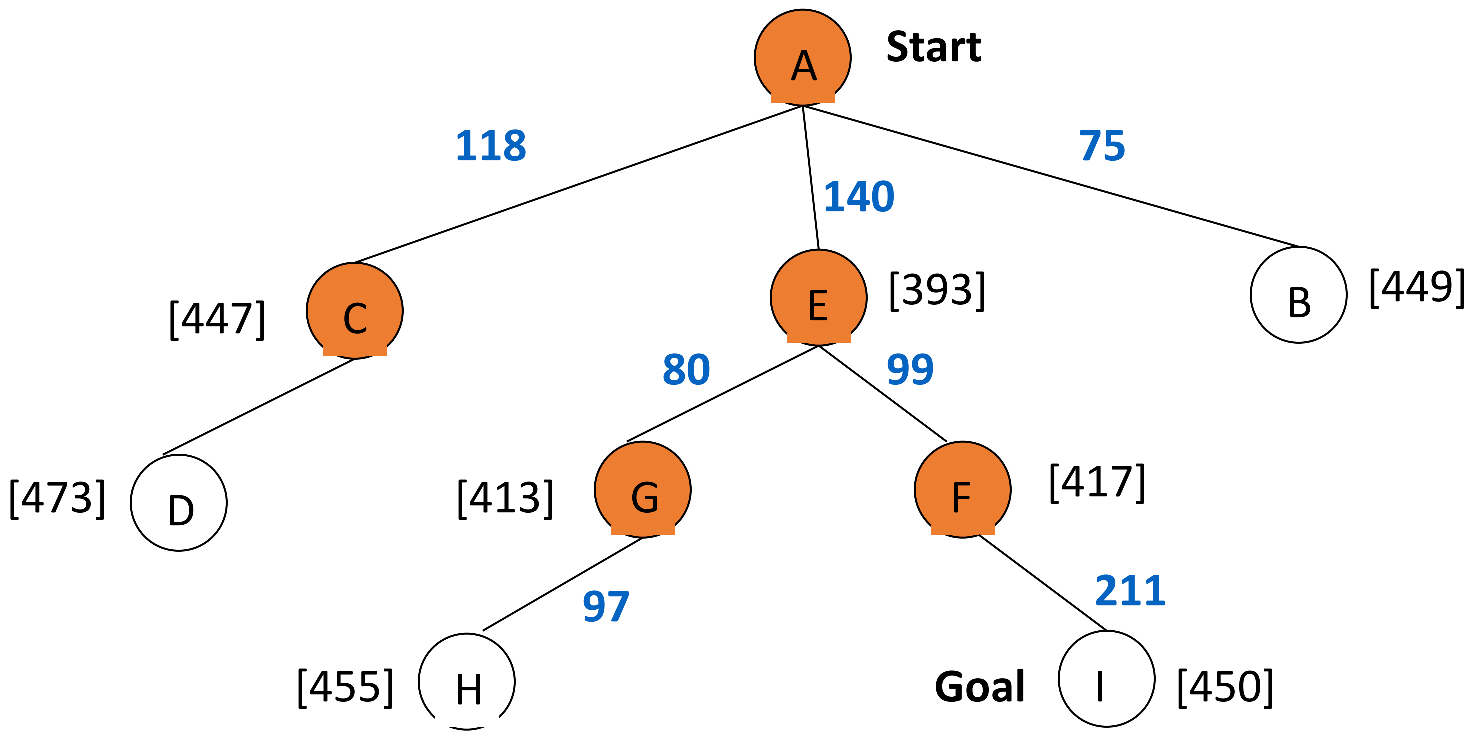
A* Search: Tree Search



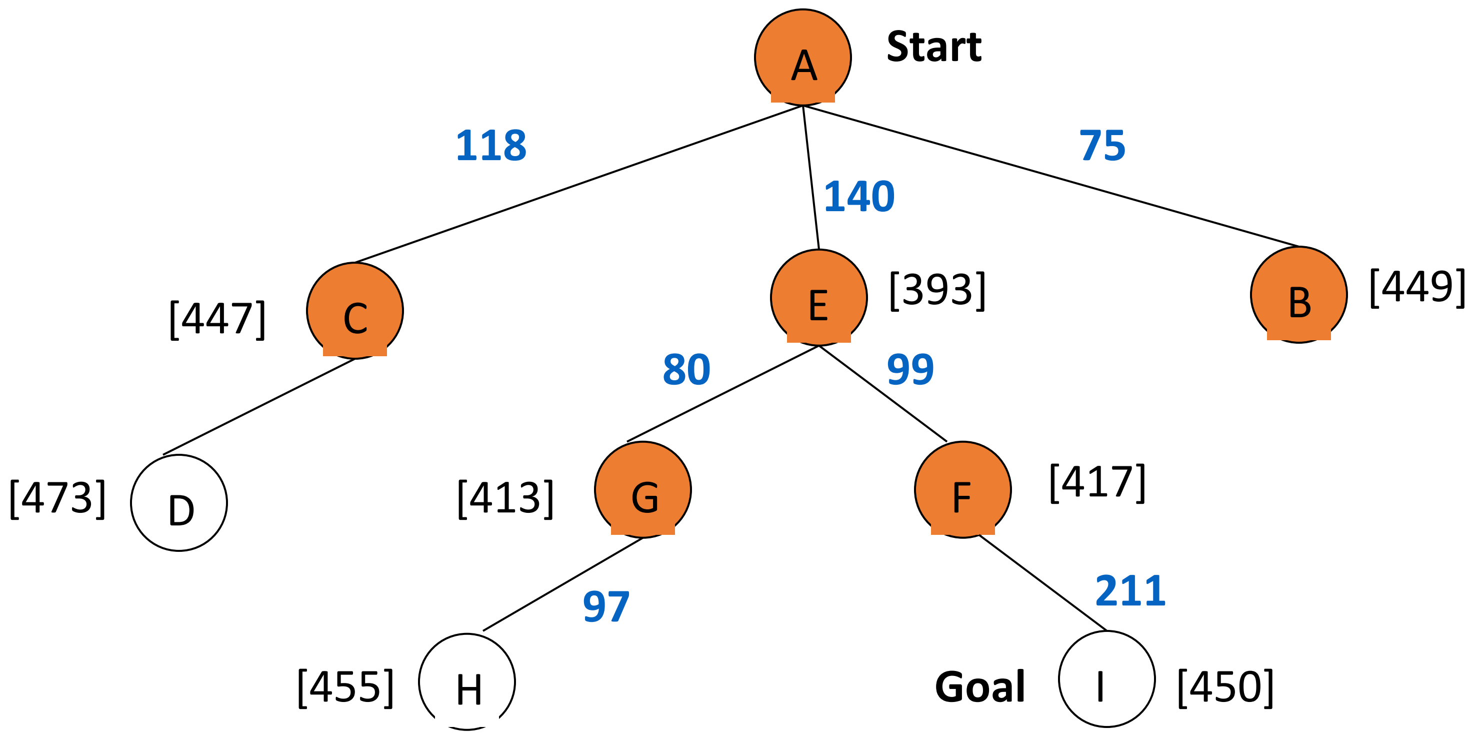
A* Search: Tree Search



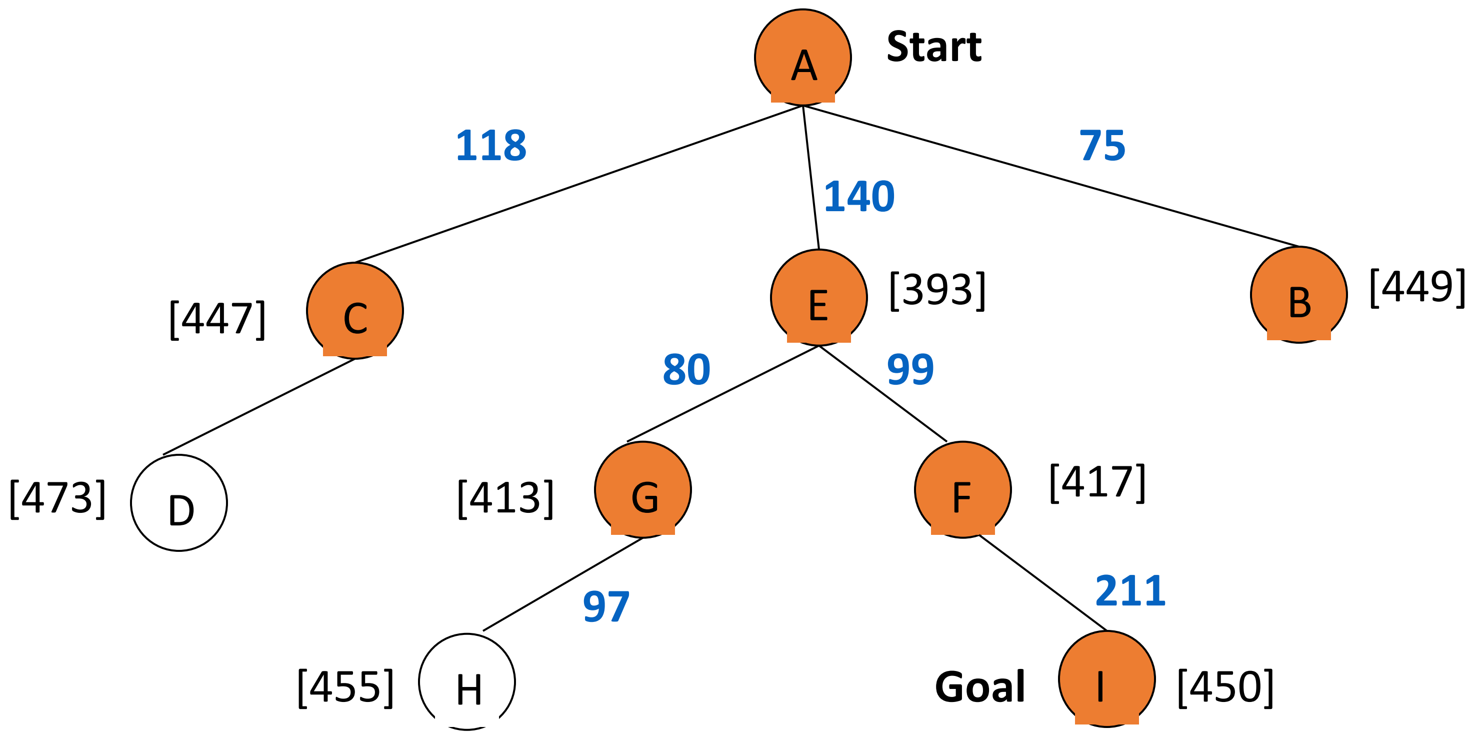
A* Search: Tree Search



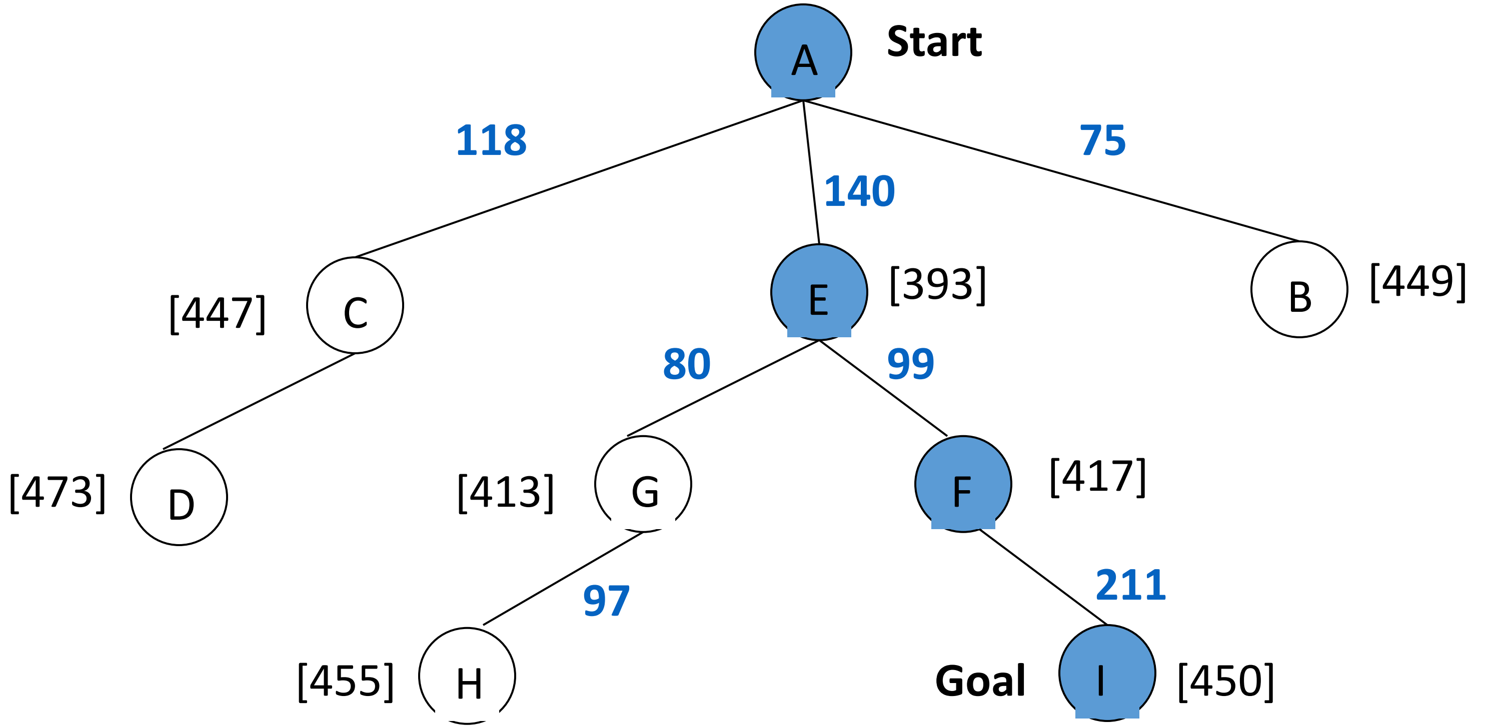
A* Search: Tree Search



A* Search: Tree Search



A* Search: Tree Search



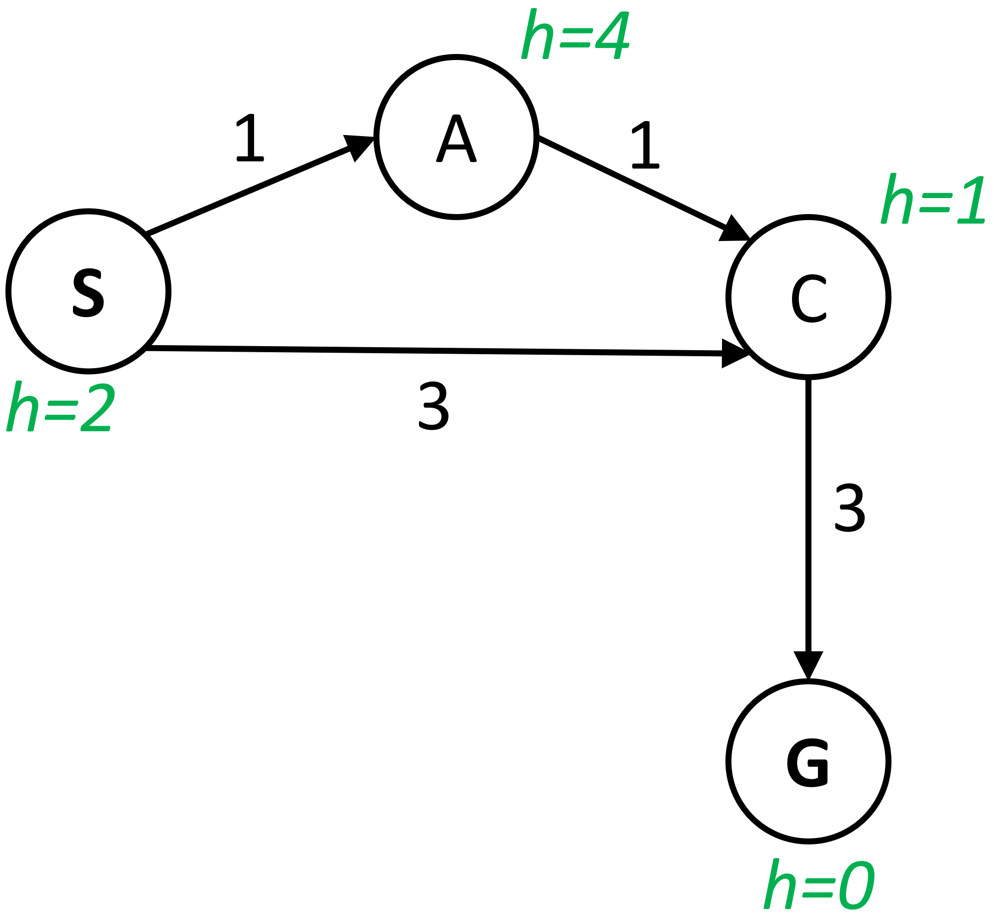
A* not optimal !!!

When heuristic h is not admissible

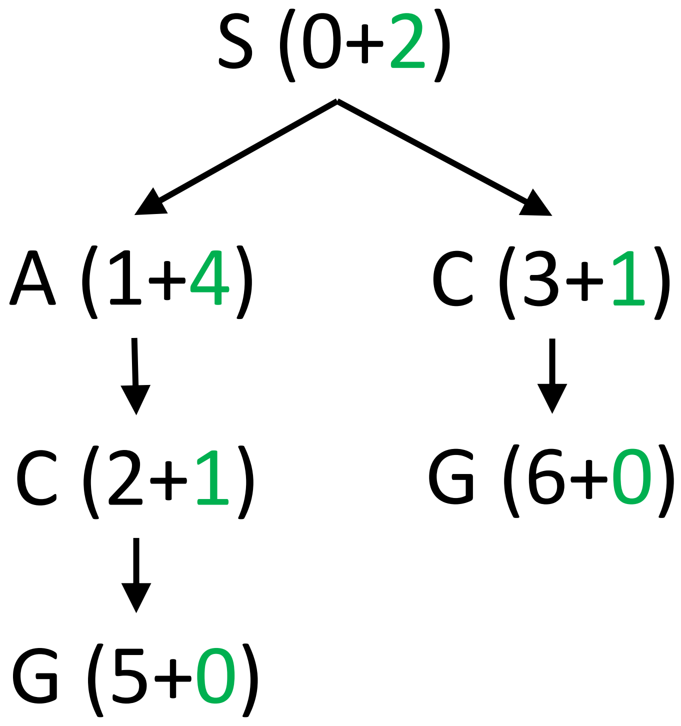
dist(A-E-F-I) =140+99+211=450

A* Tree Search

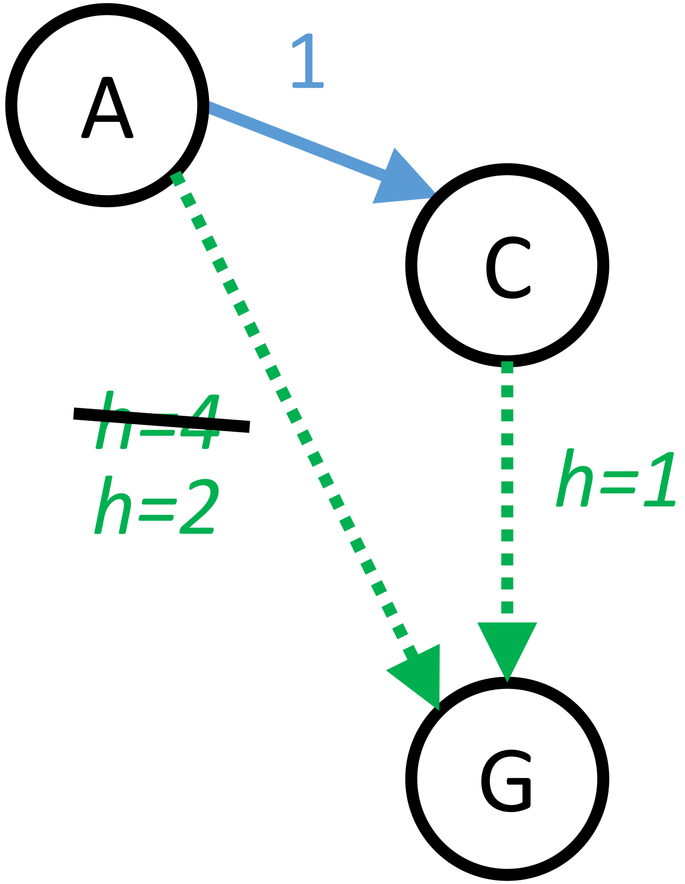
State space graph



Search tree



Consistency of Heuristics



- Main idea: Estimated heuristic costs ≤ actual costs
  - Admissibility:
    heuristic cost ≤ actual cost to goal
    h(A) ≤ actual cost from A to G
  - Consistency:
    “heuristic step cost” ≤ actual cost for each step
    h(A) – h(C) ≤ cost(A to C)
    triangle inequality
    h(A) ≤ cost(A to C) + h(C)
- Consequences of consistency:
  - The f value along a path never decreases
  - A* graph search is optimal

But an admissible heuristic is not always consistent

A* Algorithm

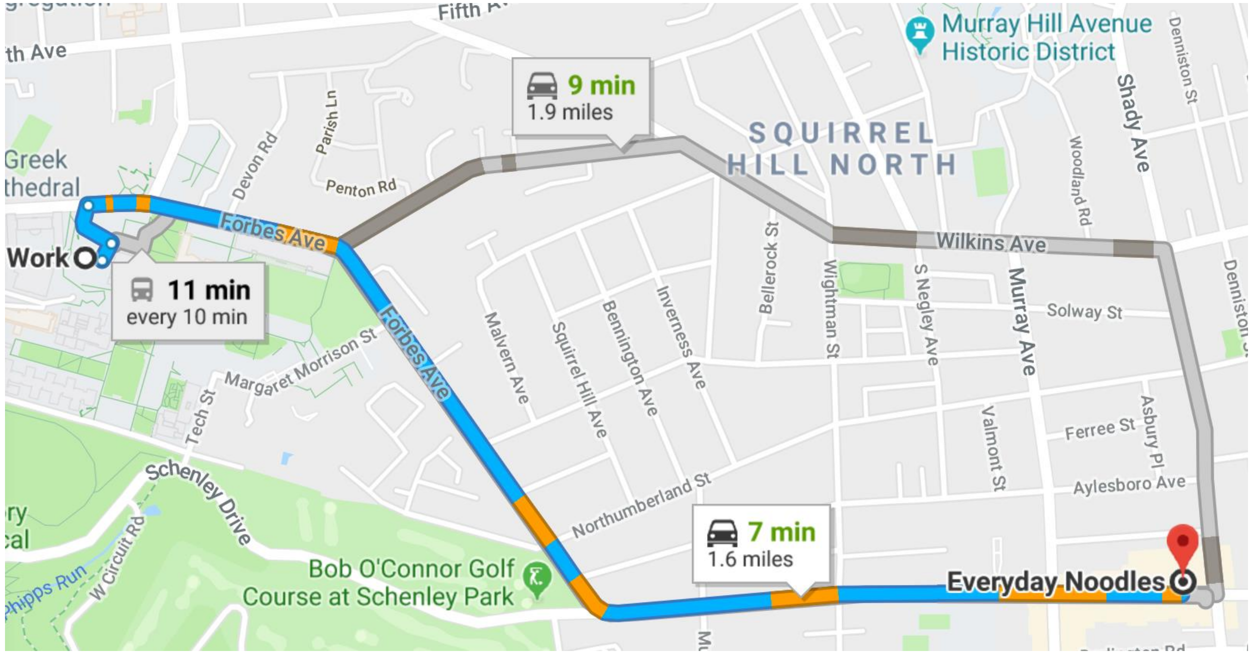
1. Search queue Q is empty.
2. Place the start state s in Q with f value h(s).
3. If Q is empty, return failure.
4. Take node n from Q with lowest f value.
   (Keep Q sorted by f values and pick the first element).
5. If n is a goal node, stop and return solution.
6. Generate successors of node n.
7. For each successor n’ of n do:
   a) Compute f(n’) = g(n) + h(n’).
   b) If n’ is new (never generated before), add n’ to Q.
   c) If node n’ is already in Q with a higher f value, replace it with current f(n’) and place it in sorted order in Q.
   End for
8. Go back to step 3.

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
    initialize the explored set to be empty
    initialize the frontier as a priority queue using g(n) as the priority
    add initial state of problem to frontier with priority g(S) = 0
    loop do
        if the frontier is empty then
            return failure
        choose a node and remove it from the frontier
        if the node contains a goal state then
            return the corresponding solution
        add the node state to the explored set
        for each resulting child from node
            if the child state is not already in the frontier or explored set then
                add child to the frontier
            else if the child is already in the frontier with higher g(n) then
                replace that frontier node with child
```

```
function A-STAR-SEARCH(problem) returns a solution, or failure
    initialize the explored set to be empty
    initialize the frontier as a priority queue using f(n) = g(n) + h(n) as the priority
    add initial state of problem to frontier with priority f(S) = 0 + h(S)
    loop do
        if the frontier is empty then
            return failure
        choose a node and remove it from the frontier
        if the node contains a goal state then
            return the corresponding solution
        add the node state to the explored set
        for each resulting child from node
            if the child state is not already in the frontier or explored set then
                add child to the frontier
            else if the child is already in the frontier with higher f(n) then
                replace that frontier node with child
```

A* Applications

- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Video games
- Machine translation
- Speech recognition
- …

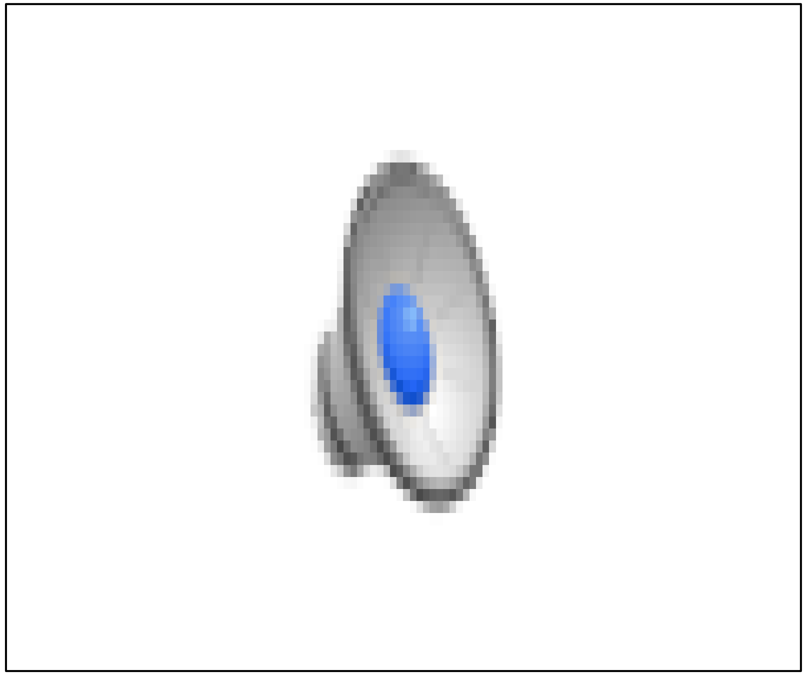
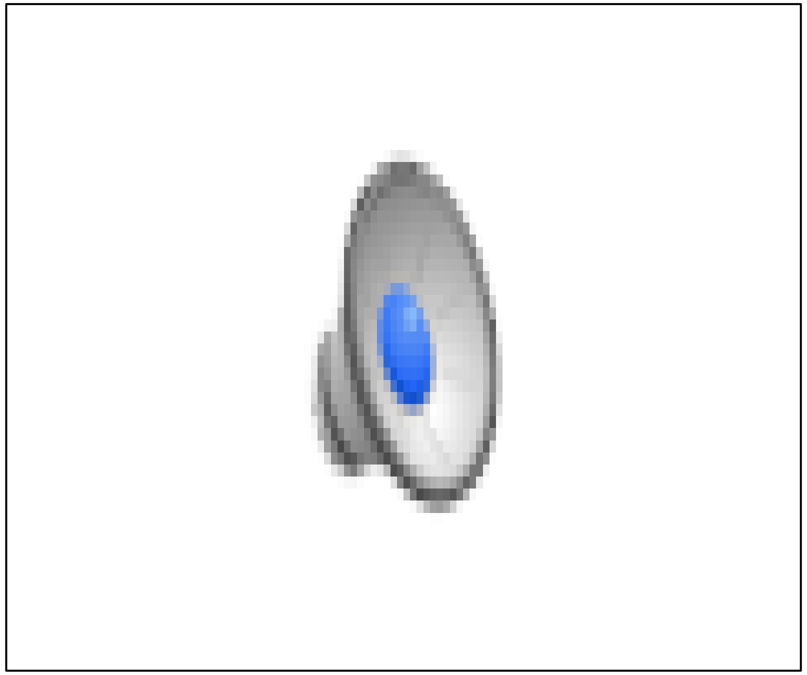
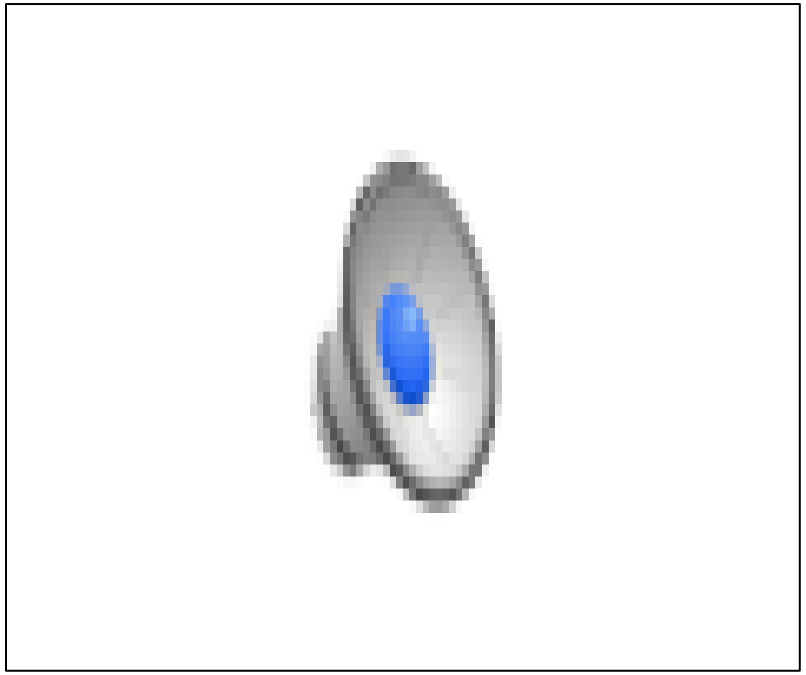


Properties of A*

Properties of A*

- Complete? **Yes**
  - (unless there are infinitely many nodes with $f \leq f(G)$)
- Time? **Exponential**
  - It depends on the quality of heuristic but is still exponential.
  - with respect to the length of the solution
  - better than other algorithms, but still problematic
- Space? Keeps all nodes in memory
  - **$O(b^d)$** space complexity,
  - A* keeps all generated nodes in memory
  - But an iterative deepening version is possible (IDA*).
- Optimal? **Yes**
  - A* is optimal if heuristic *h* is admissible (optimal).

Video of Demo Contours

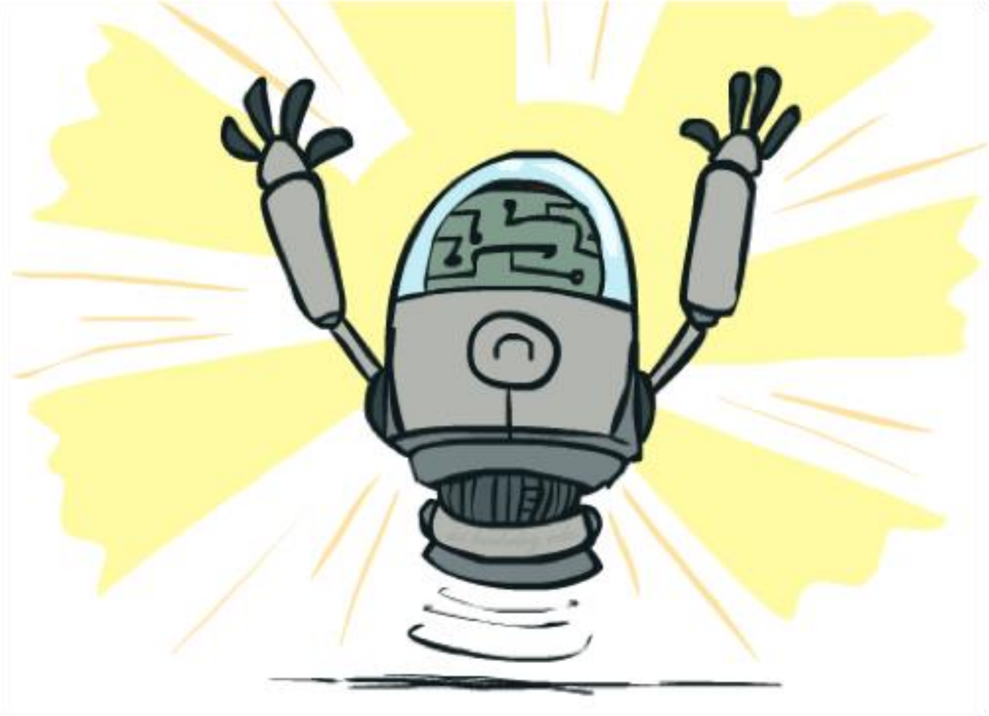




Optimality

- Tree search:
  - A* is optimal if heuristic is admissible
  - UCS is a special case (h = 0)

- Graph search:
  - A* optimal if heuristic is consistent
  - UCS optimal (h = 0 is consistent)

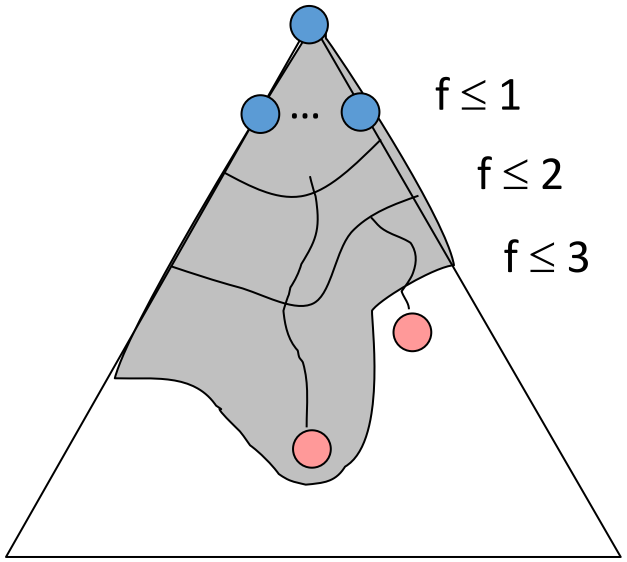
- Consistency implies admissibility



- In general, most natural admissible heuristics tend to be consistent, especially if from relaxed problems

Optimality of A* Graph Search

- Sketch: consider what A* does with a consistent heuristic:
  - Fact 1: In tree search, A* expands nodes in increasing total f value (f-contours)
  - Fact 2: For every state s, nodes that reach s optimally are expanded before nodes that reach s suboptimally



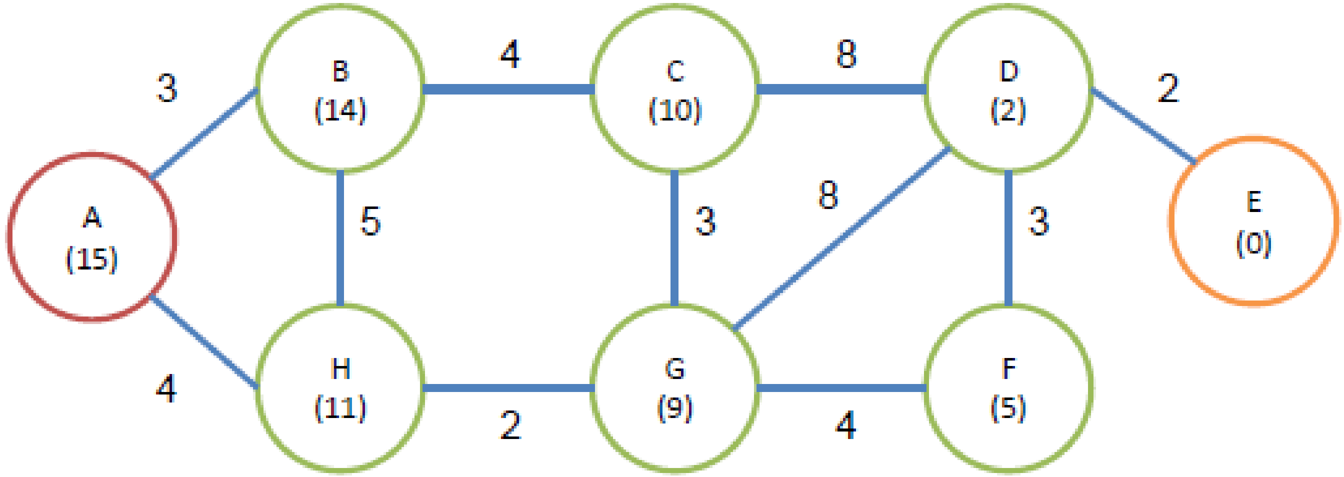
Relationship of Search Algorithms

- Notation
  - g(n) = known cost so far to reach n
  - h(n) = estimated (optimal) cost from n to goal
  - f(n) = g(n)+h(n) = estimated (optimal) total cost through n

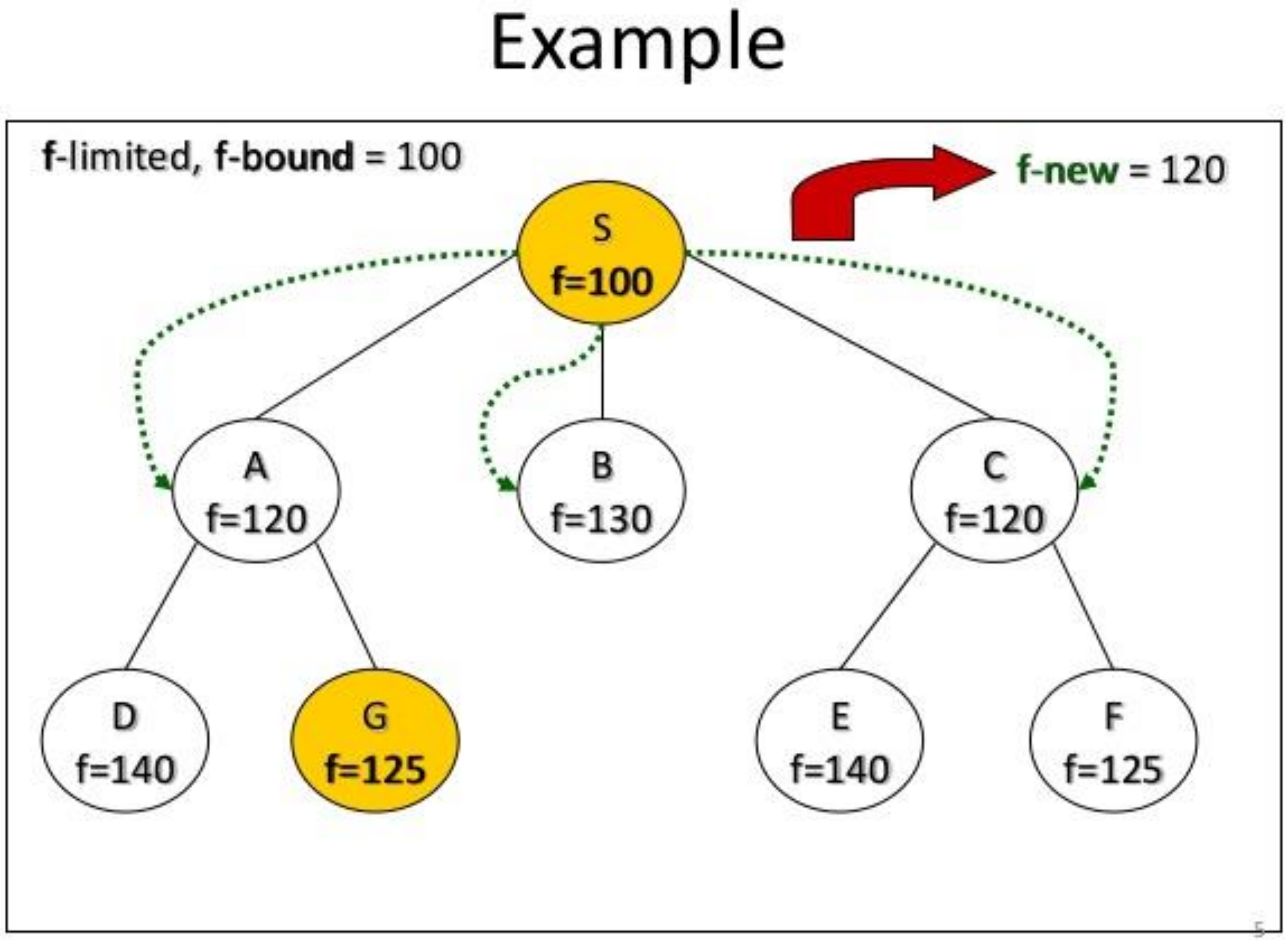
- Uniform cost search: sort frontier by g(n)
- Greedy best-first search: sort frontier by h(n)
- A* search: sort frontier by f(n)
  - Optimal for admissible / consistent heuristics
  - Generally the preferred heuristic search framework
  - Memory-efficient versions of A* are available: RBFS, SMA*

Question

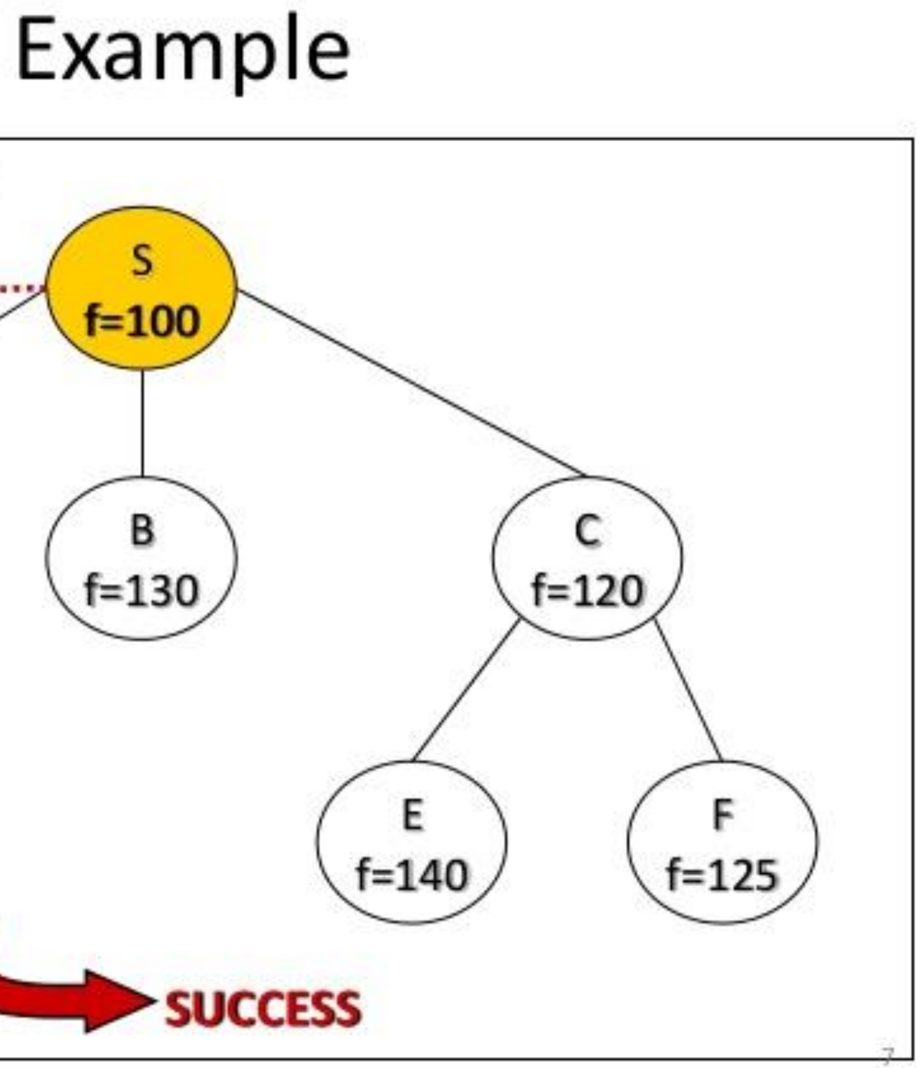
- Perform Uniform Cost Search, the Best-First Search, and the A* algorithm. Here we suppose that **<u>A is the initial node, and E is the target</u>**. The cost of each edge and the heuristic value of the each node are also given in the figure.

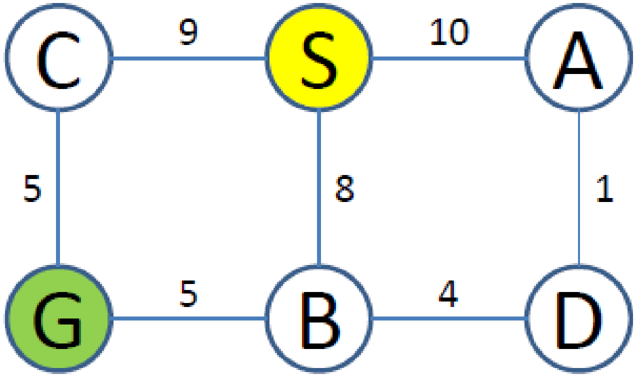


IDA*

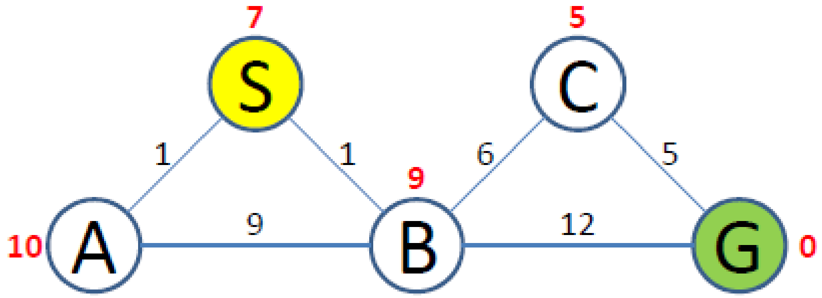

Example
f-limited, f-bound = 100
f-new = 120
S
f=100
A
f=120
B
f=130
C
f=120
D
f=140
G
f=125
E
f=140
F
f=125
5

IDA*

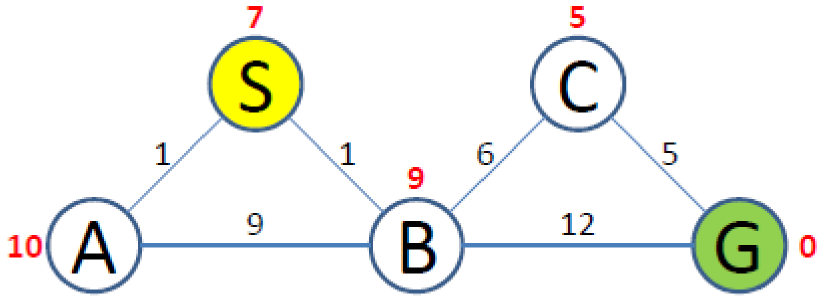

Example
f-limited, f-bound = 125
S
f=100
A
f=120
B
f=130
C
f=120
D
f=140
G
f=125
E
f=140
F
f=125
SUCCESS
7

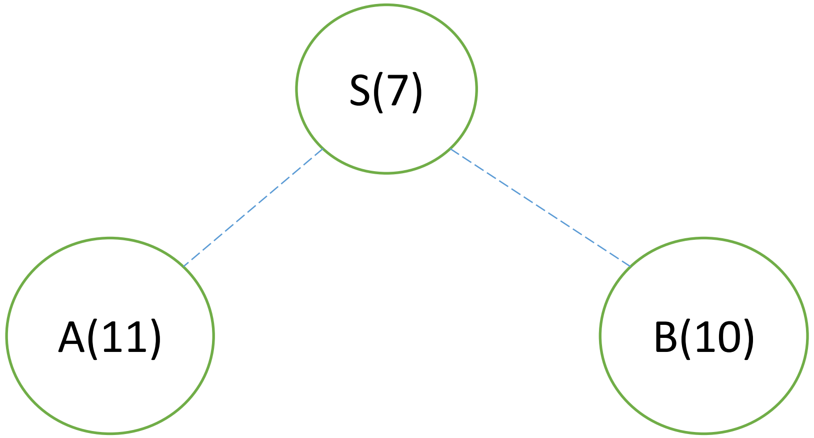


	S	A	B	C	D	G
heuristic	0	0	4	3	0	0

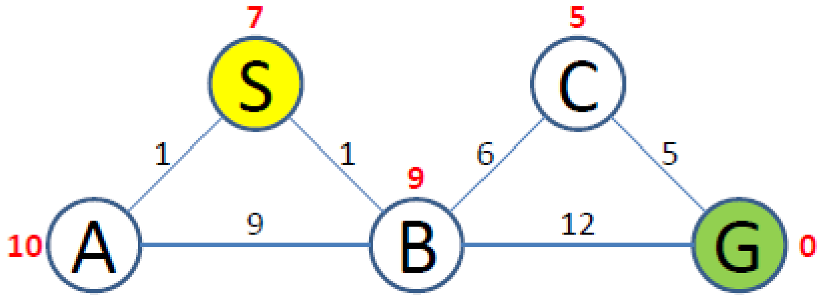


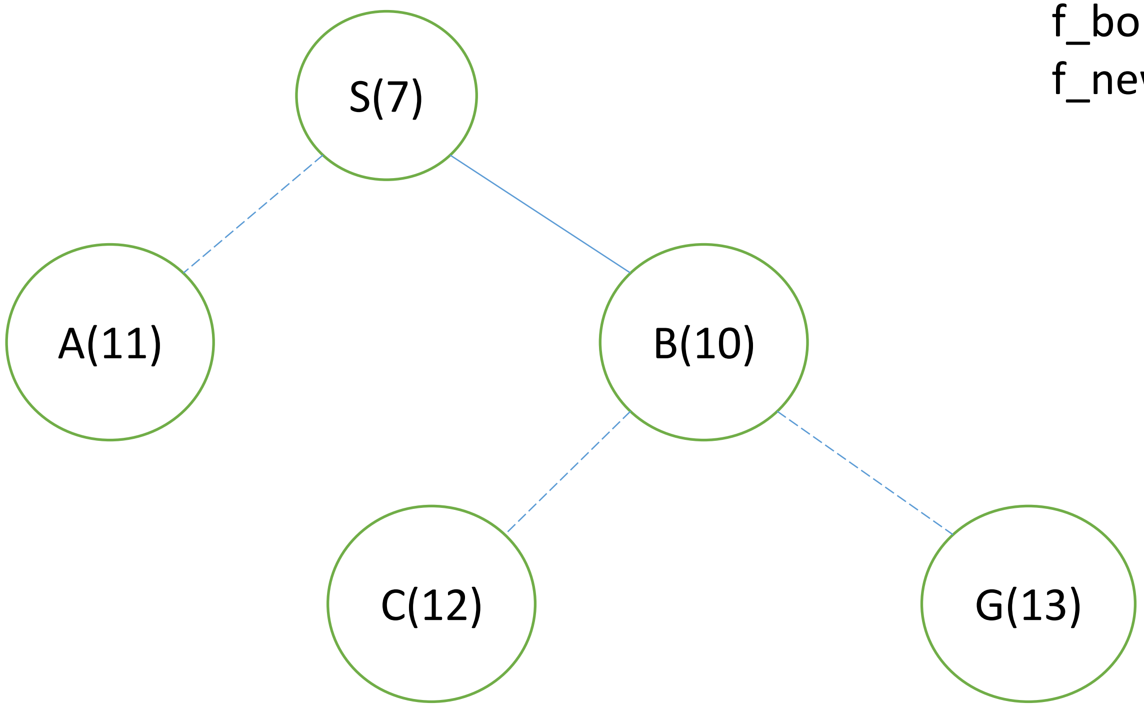
f_bound = 0
f_new = 7





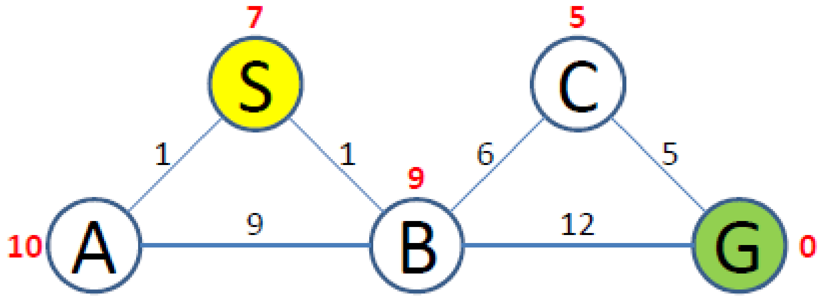
f_bound = 7
f_new = 10





f_bound = 10

f_new = 11



f_bound = 11
f_new =