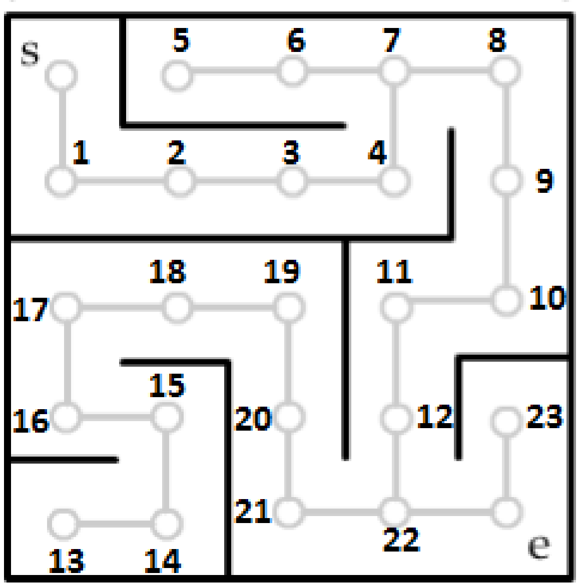
Activity

S and e are two players lost in a maze.

We know their starting locations.

Perform a bidirectional search so that they can meet at a single point.



Comparison Uninformed Search Strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes^{a}	Yes^{a,b}	No	No	Yes^{a}	Yes^{a,d}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes^{c}	Yes	No	No	Yes^{c}	Yes^{c,d}

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: ^{a} complete if b is finite; ^{b} complete if step costs $\geq \epsilon$ for positive ϵ; ^{c} optimal if step costs are all identical; ^{d} if both directions use breadth-first search.

Conclusions

- Interesting problems can be cast as search problems, but you've got to set up state space
- Problem formulation usually requires abstracting away real-world details to define a state space that can feasibly be explored

Outline

- Informed Search
- Uninformed Versus Informed Search
- Best First Search
- Greedy Search
- A* Search

Informed Search

- Informed searches use domain knowledge to guide the selection of the best path to continue searching
- Heuristics are used, which are informed guesses
- Heuristic means "serving to aid discovery"

Informed Search

- Define a heuristic function, ( )
  - uses domain-specific information in some way
  - is computable from the current state description
  - it estimates
    - the "goodness" of node ***n***
    - how close node ***n*** is to a goal
    - the cost of *minimal* cost path from node to a goal state

Informed Search

- $h() \geq 0$ for all nodes
- $h(n)$ close to 0 means we think *n* is close to a goal state
- $h(n)$ very big means we think *n* is far from a goal state

- All domain knowledge used in the search is encoded in the heuristic function,
- An example of a "*weak method*" for AI because of the limited way that domain-specific information is used to solve a problem

Uninformed versus Informed

Uninformed search	Informed search
• does not have any additional information on the quality of states. • So, it is impossible to determine which state is the better than others. As a result, search efficiency depends only on the structure of a state space	• heuristically informed search uses a certain kind of information about states in order to guide search along promising branches within a state space. • Using problem-specific knowledge as hints to guide the search.

Uninformed versus Informed (cont)

Uninformed search	Informed search
• look for solutions by <u>systematically</u> generating new states and checking each of them against the goal. 1. It is very <u>inefficient</u> in most cases. 2. Most successor states are "obviously" a bad choice. 3. Such strategies do not use problem-specific knowledge	1. They are almost always <u>more efficient</u> than uninformed strategies. 2. May reduce time and space complexities. 3. Evaluation function f(n) measures distance to the goal. 4. Order nodes in Frontier according to f(n) and decide which node to expand next.

Best-First Search

An **informed search strategy** uses problem-specificknowledge beyond the definition of the problem itself, so it can find solutions more efficiently than an uninformed strategy.
Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an **evaluation function, f (n)**.

•A key component of these algorithms is a heuristic function denoted *h(n)* :

h(n) = estimated cost of the cheapest path from node n to a goal node, *if n is a goal node, then h(n) = 0*.

Best-First Search

Special cases:

- Greedy Best first search
  - "Always chooses the successor node with the best f value" where $f(n) = h(n)$
  - We choose the one that is nearest to the final state among all possible choices

- A* search
  - Best first search using an "admissible" heuristic function f that takes into account the current cost g
  - Always returns the optimal solution path

Greedy Best First Search

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

Greedy Best-First Search

- Greedy best-first search tries to expand the node that is closest to the goal because it is likely to lead to a solution quickly.
- Thus, it evaluates nodes by using just the heuristic function; that is, f(n)=h(n).
- Evaluation function

  f(n) = *h(n)* (heuristic) = estimate of cost from *n* to *goal*

- Assume hold(n) = straight line distance from n to Bucharest, so Greedybest-first search expands the node that appears to be closest to the goal

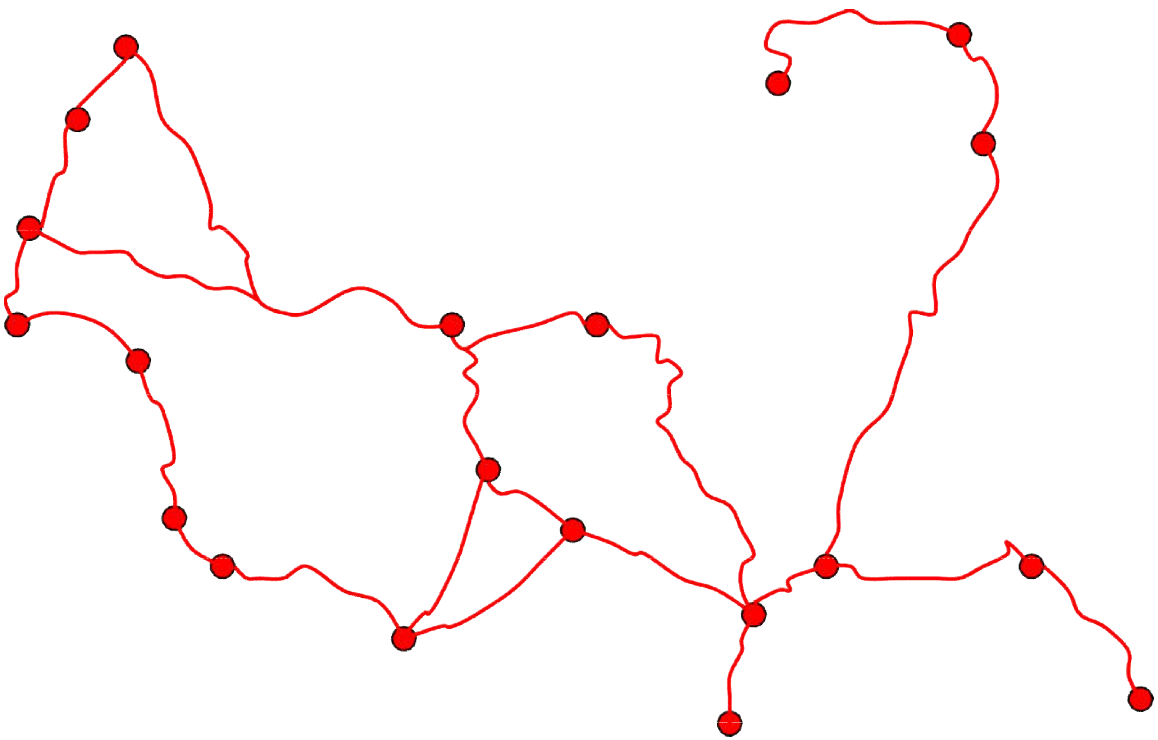
Romania

SATU MARE
BAIA MARE
SUCEAVA
BOTOŞANI
ZALĂU
IAŞI
ORADEA
BISTRIŢA
PIATRA - NEAMŢ
CLUJ NAPOCA
VASLUI
TÎRGU MUREŞ
BACĂU
MIERCUREA - CIUC
ARAD
ALBA IULIA
SFÎNTU GHEORGHE
SIBIU
TIMIŞOARA
DEVA
FOCŞANI
BRAŞOV
GALAŢI
REŞIŢA
BRĂILA
RÎMNICU VÎLCEA
BUZĂU
TULCEA
PLOIEŞTI
TÎRGOVIŞTE
PITEŞTI
DROBETA TURNU SEVERIN
SLOBOZIA
BUCUREŞTI
SLATINA
CRAIOVA
CONSTANŢA
CĂLĂRAŞI
ALEXANDRIA
GIURGIU
THE BLACK SEA

Romania


SATU MARE
BAIA MARE
SUCEAVA
BOTOSANI
ZALAU
ORADEA
BISTRITA
CLUJ NAPOCA
PIATRA - NEAMT
IASI
VASLUI
TIRGU MURES
BACAU
MIERCUREA - CIUC
ARAD
ALBA IULIA
SIBIU
SFINTU GHEORGHE
TIMISOARA
DEVA
BRASOV
FOCSANI
GALATI
RESITA
BRAILA
BUZAU
TULCEA
PLOIESTI
TIRGOVISTE
DROBETA TURNU SEVERIN
SLOBOZIA
BUCURESTI
SLATINA
CRAIOVA
CONSTANTA
CALARASI
ALEXANDRIA
GIURGIU
THE BLACK SEA

Romania



Map ofRomania

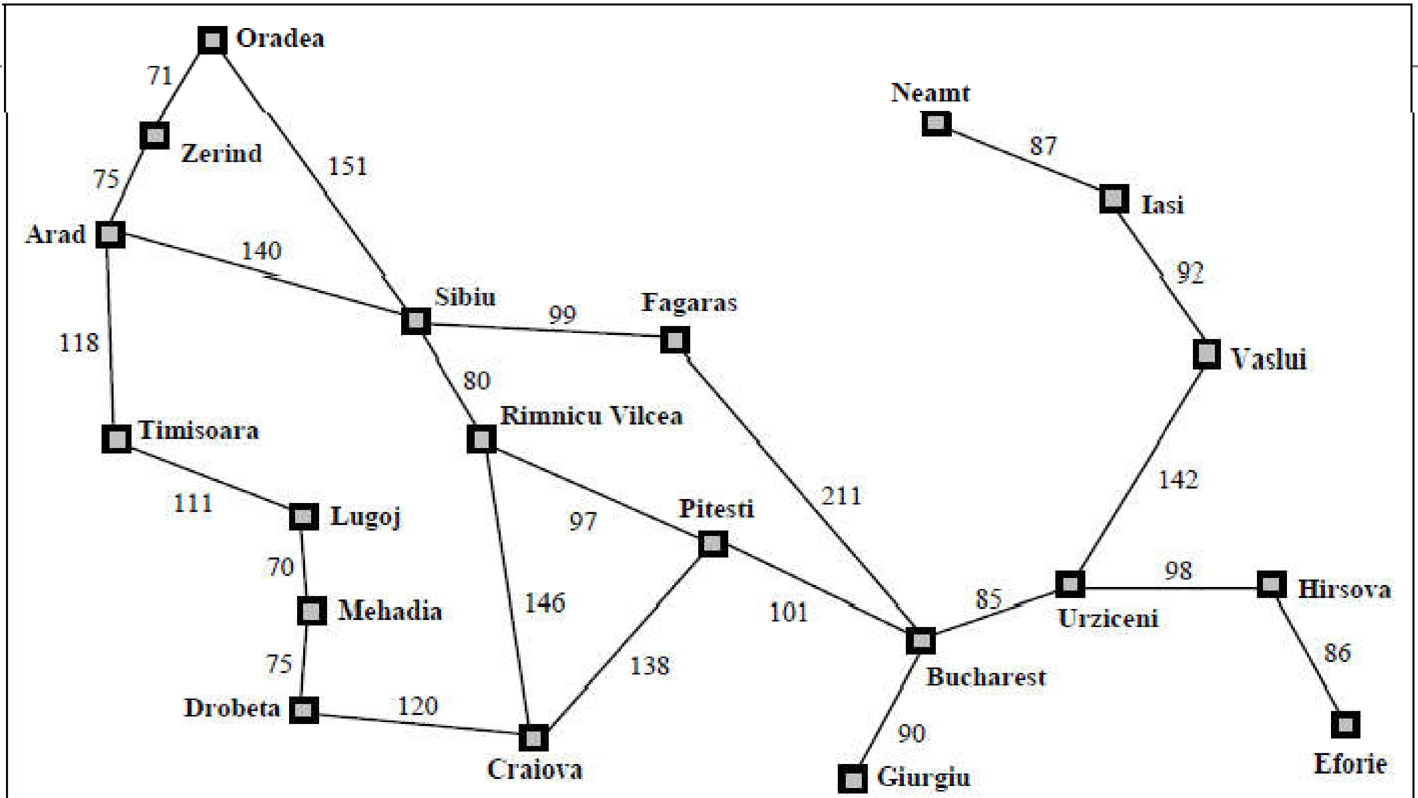
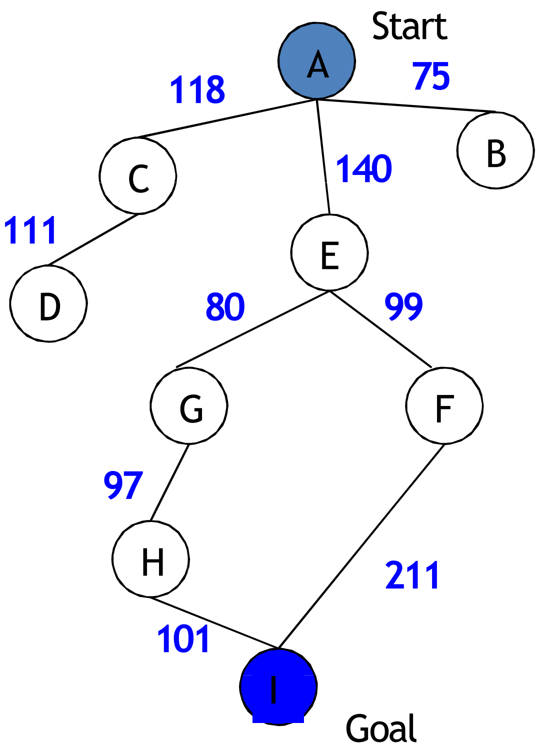


Figure 3.2 A simplified road map of part of Romania.

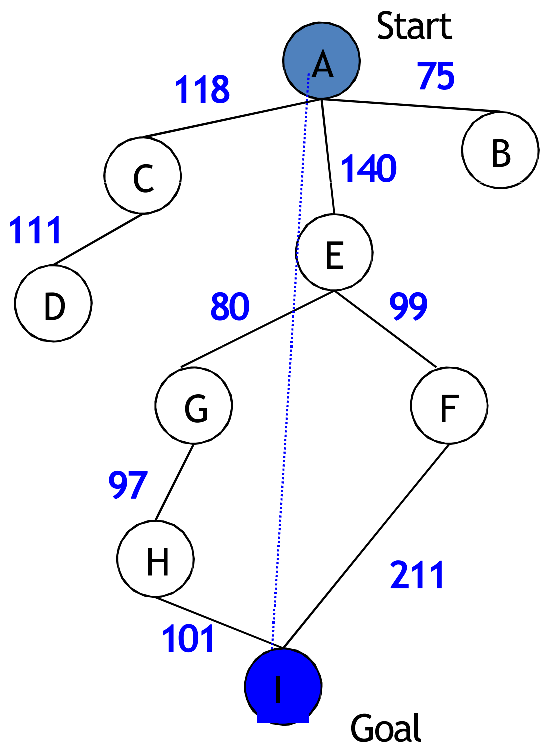
Greedy Search



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**

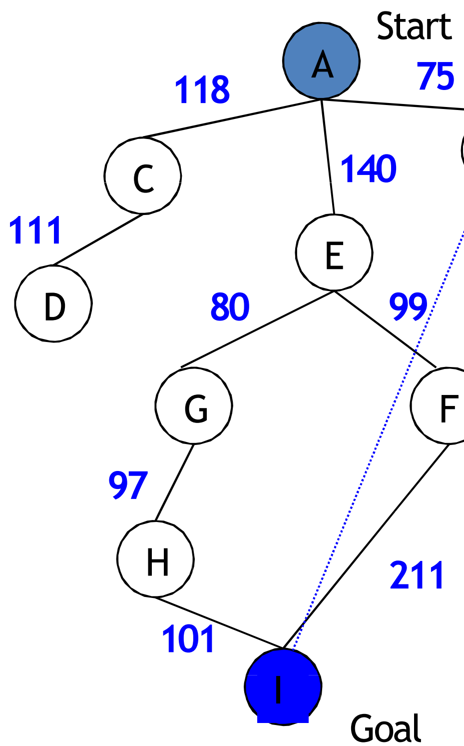
Greedy Search



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**

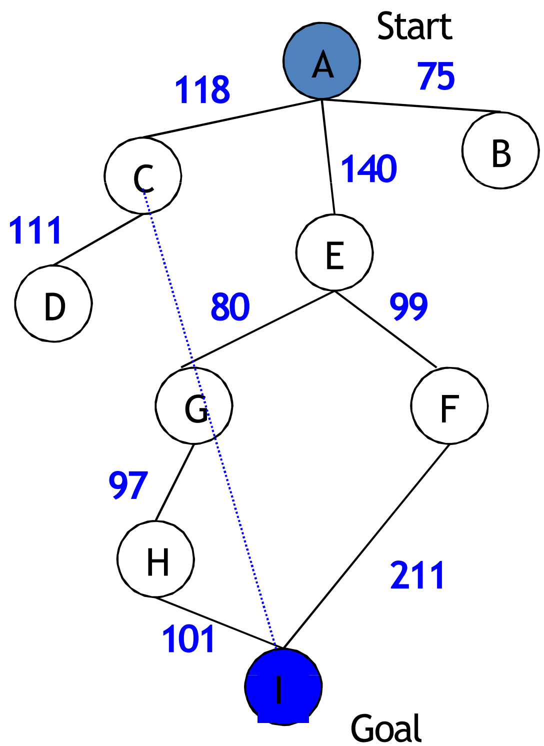
Greedy Search



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**

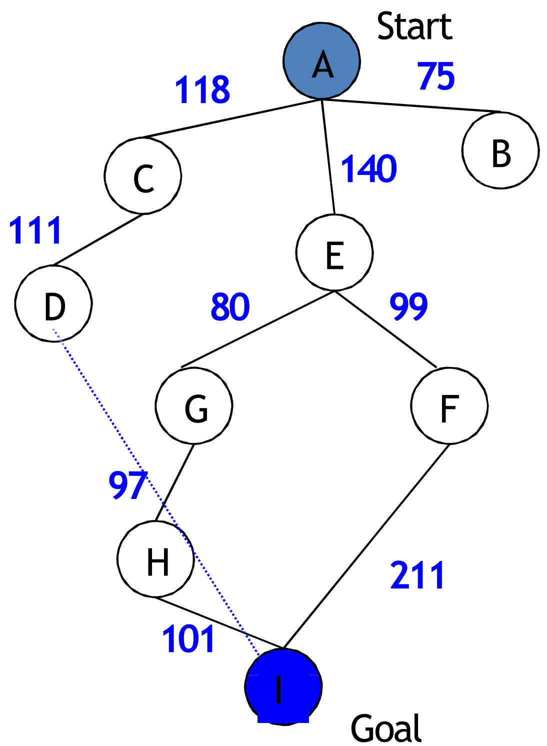
Greedy Search



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**

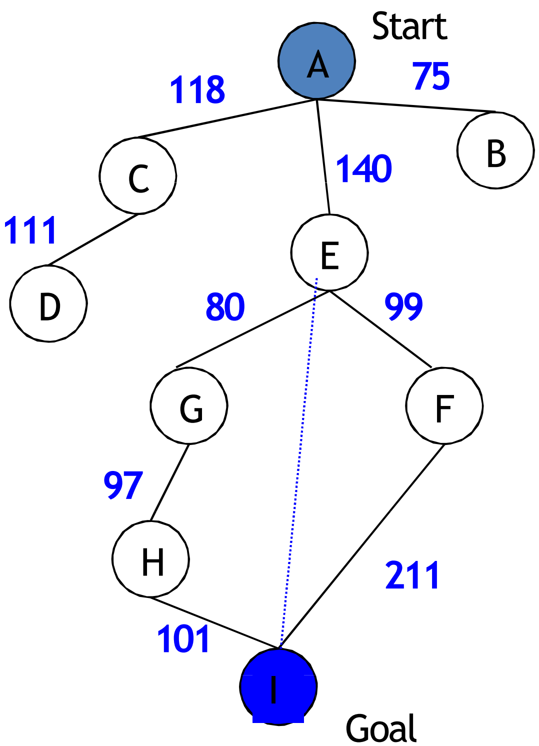
Greedy Search



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**

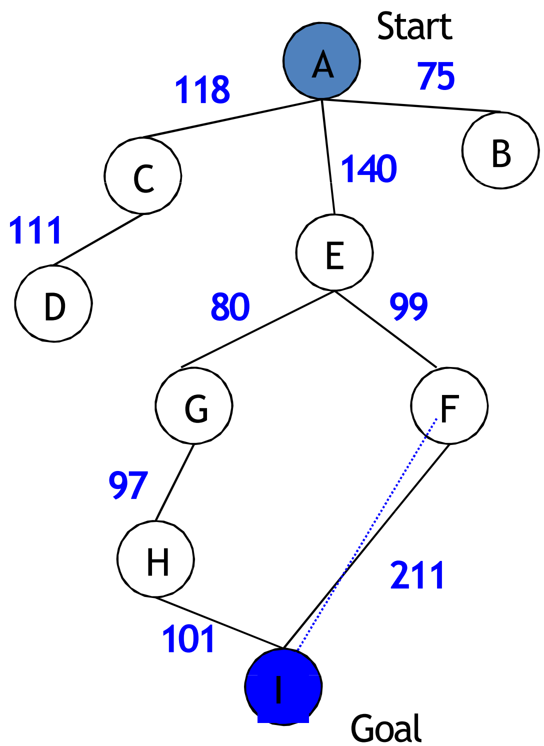
Greedy Search



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**

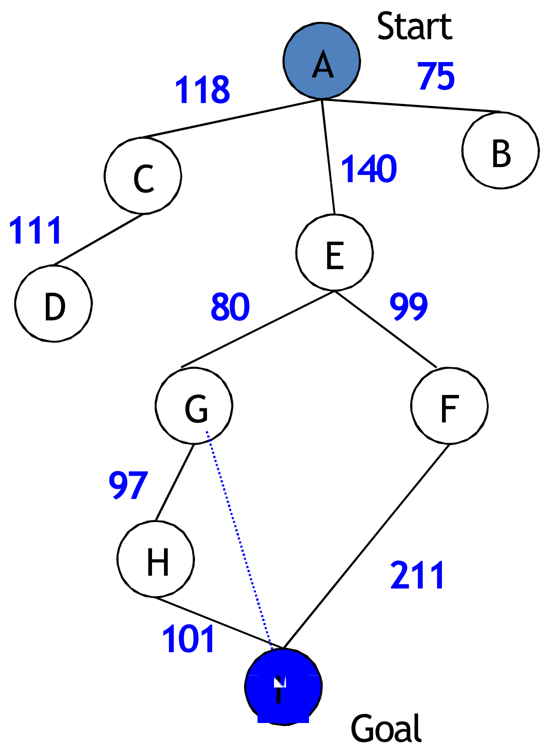
Greedy Search



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**

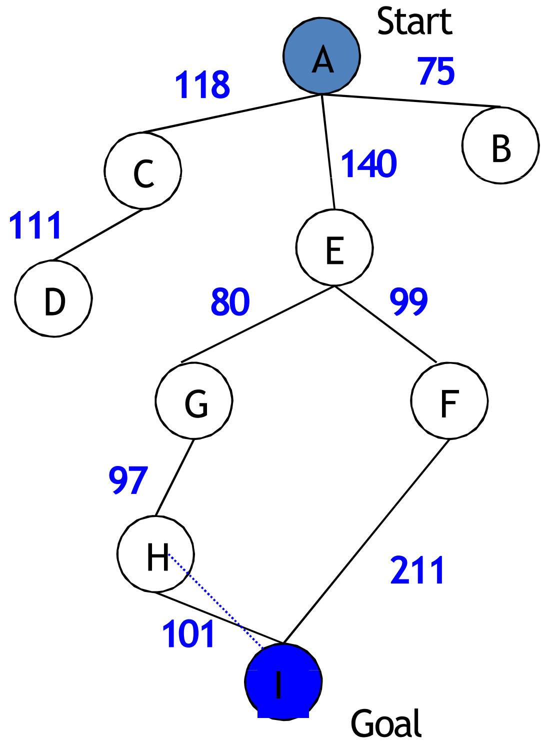
Greedy Search



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**

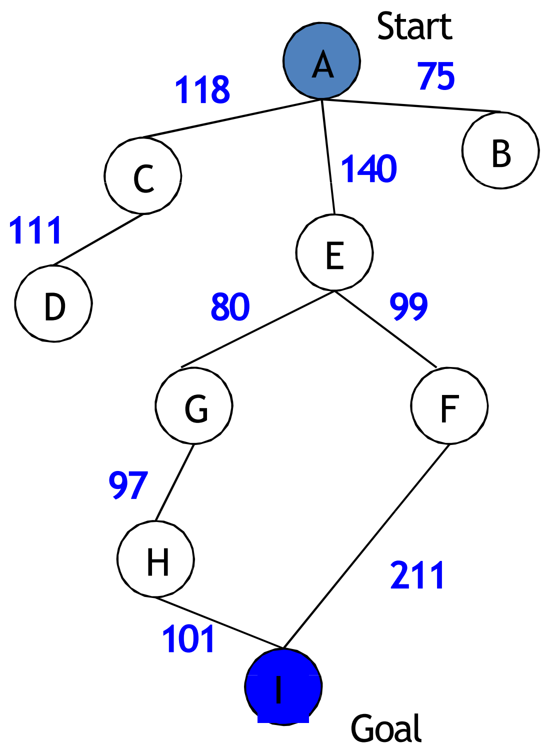
Greedy Search



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



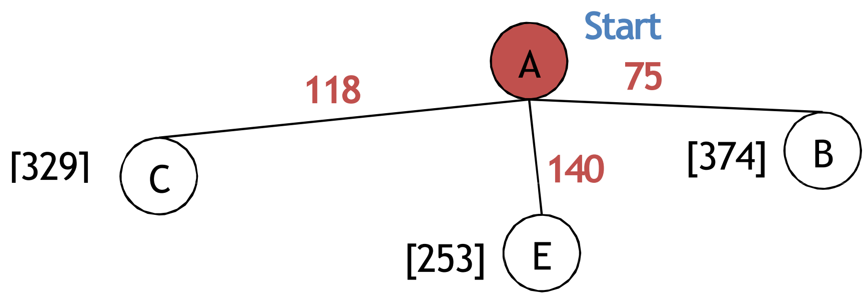
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**

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

Greedy Search – Tree Search

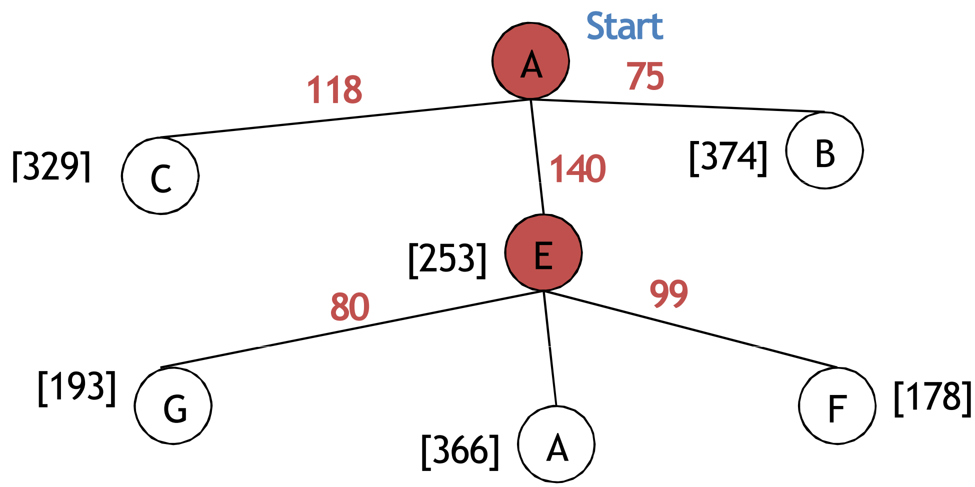
State	Heuristic : h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**

Greedy Search – Tree Search



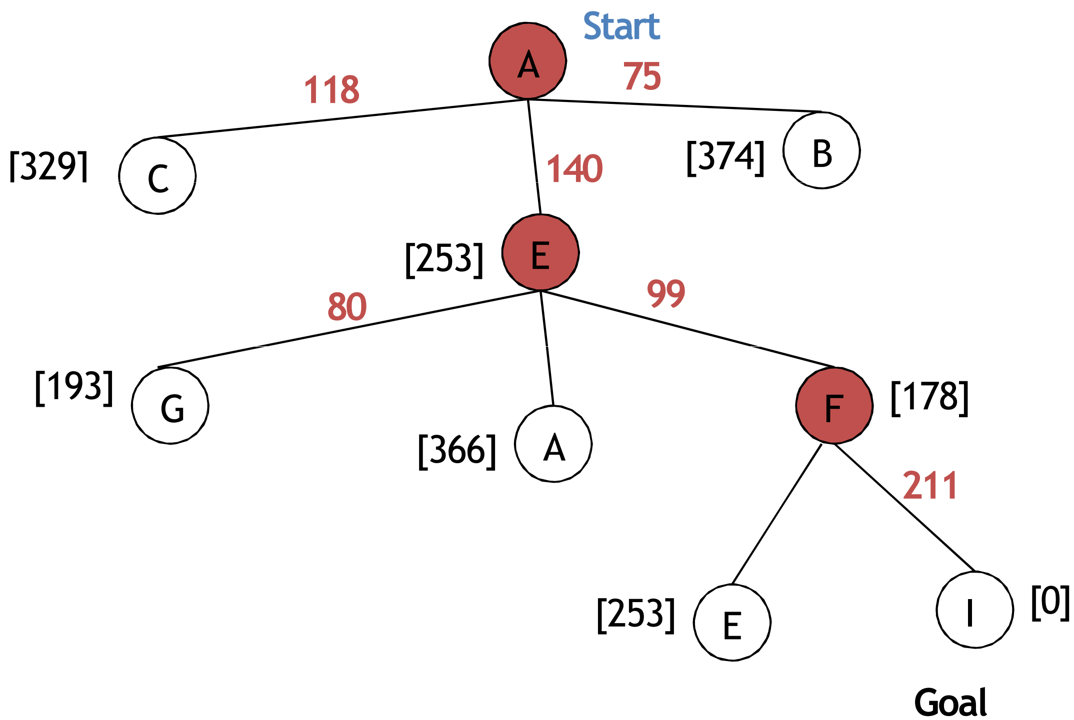
State	Heuristic : h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**

Greedy Search – Tree Search



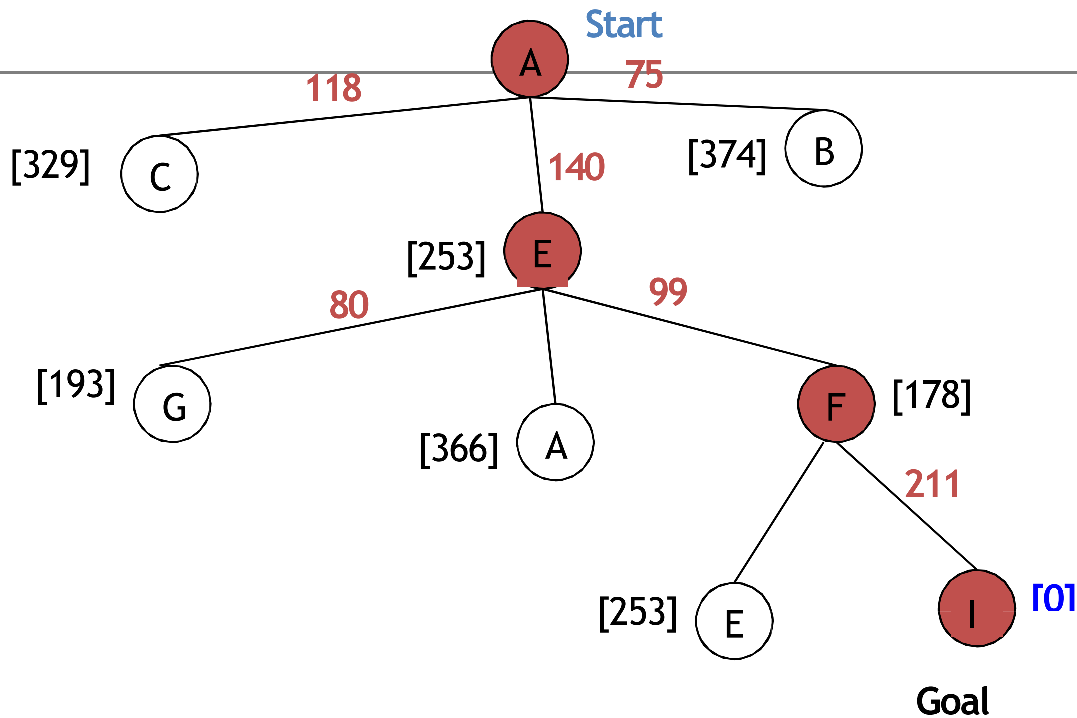
State	Heuristic : h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	**0**

Greedy Search – Tree Search



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

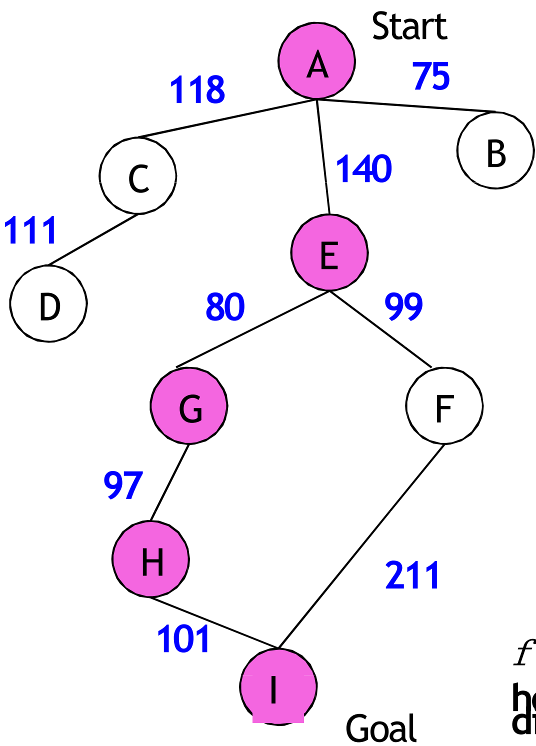
Greedy Search – Tree Search



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

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

Greedy Search: Optimal?

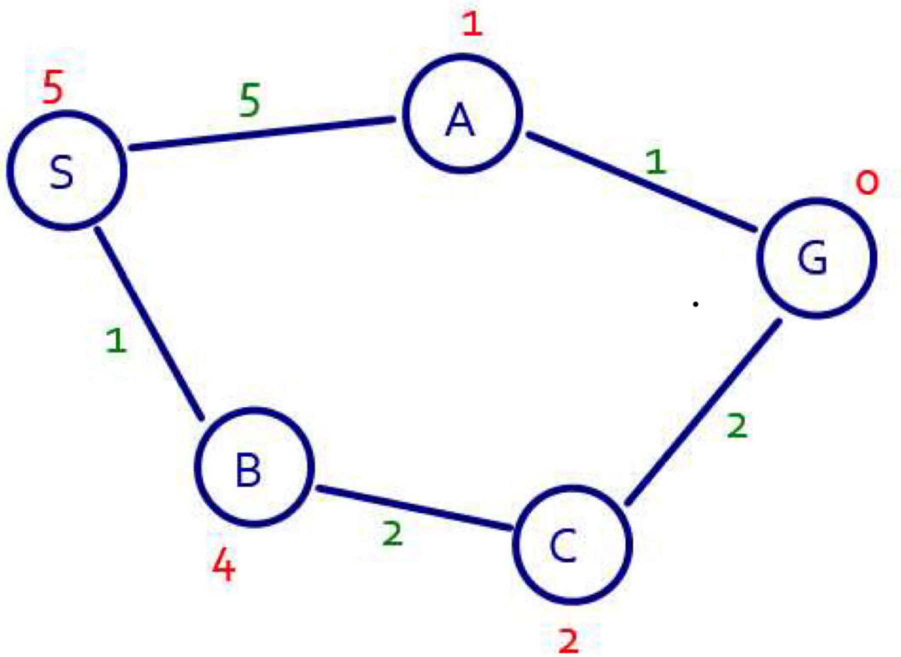


State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	**253**
F	178
G	**193**
H	**98**
I	0

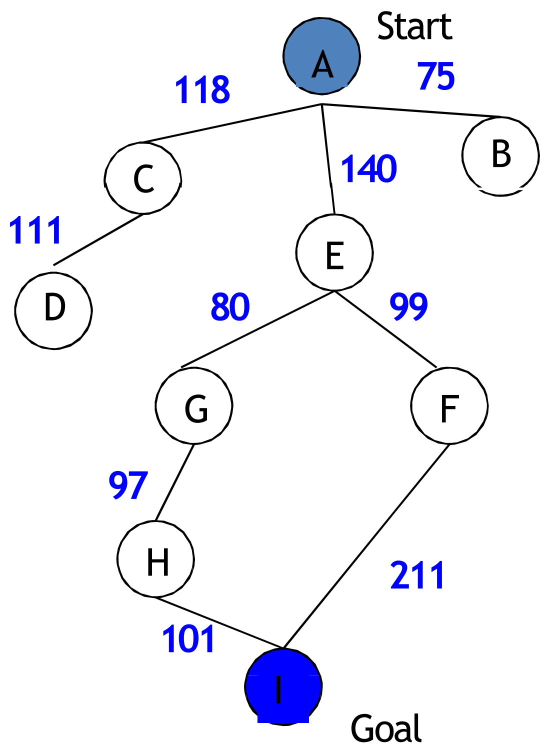
$f(n) = h(n)$ = straight-line distance heuristic

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

Greedy Search: Optimal?



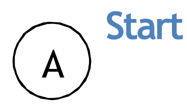
Greedy Search: Complete ?



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

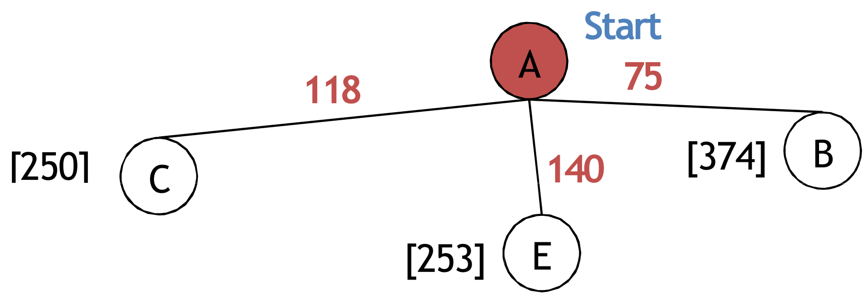
$f(n) = h(n)$ = **straight-line distance heuristic**

Greedy Search: Tree Search



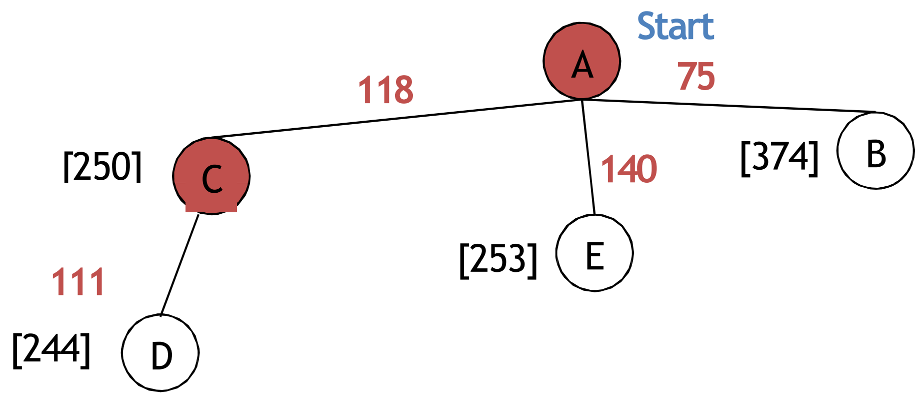
State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

Greedy Search: Tree Search



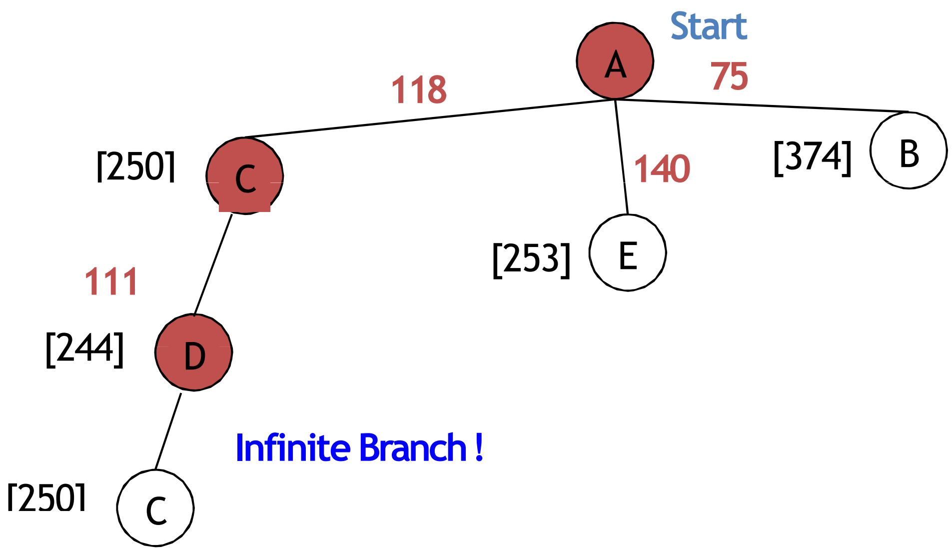
State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

Greedy Search: Tree Search



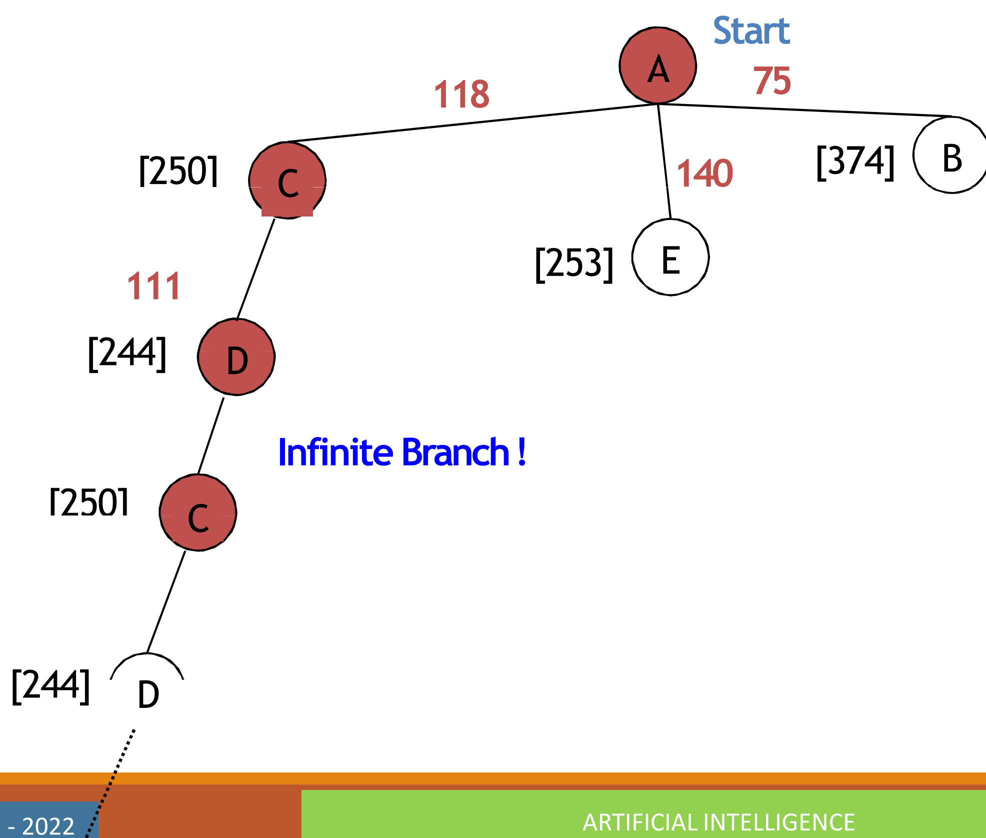
State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

Greedy Search: Tree Search



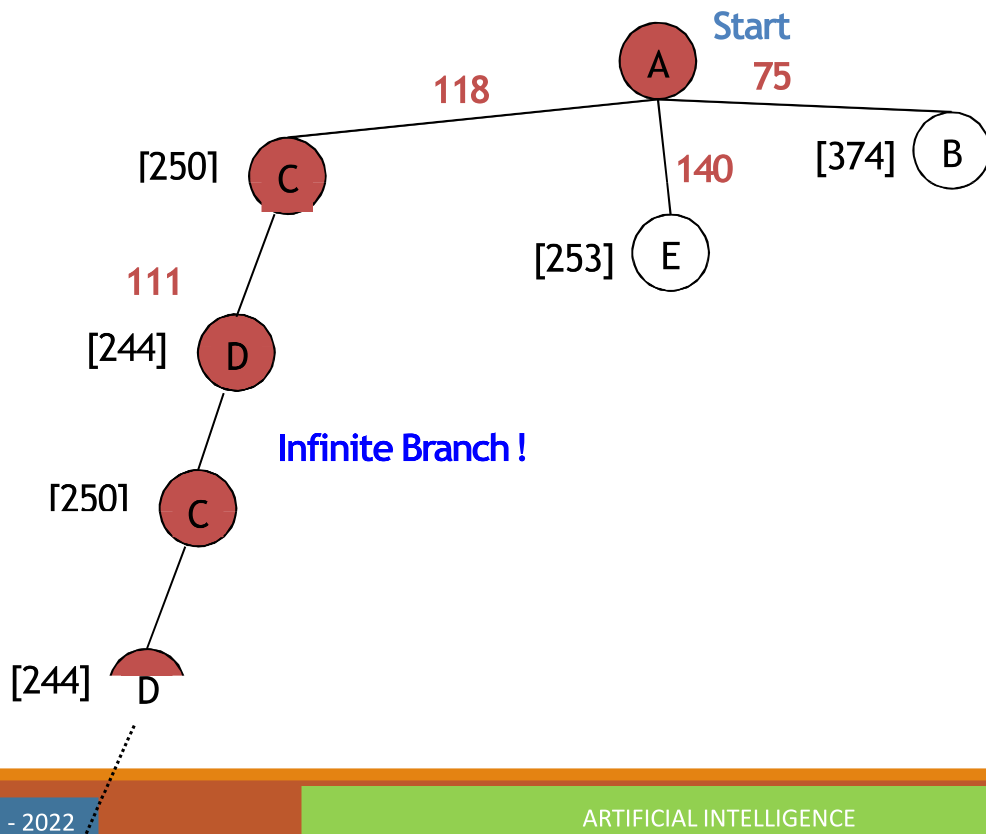
State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

Greedy Search: Tree Search



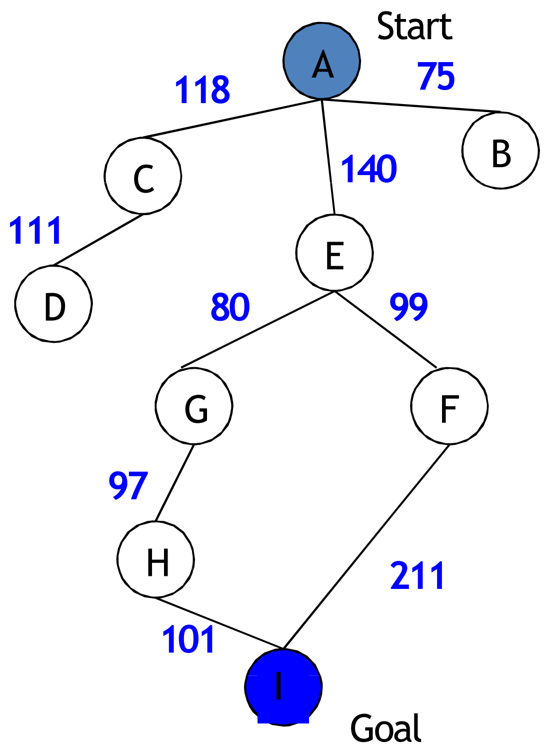
State	Heuristic: h(n)
A	366
B	374
** C	**250**
D	244
E	253
F	178
G	193
H	98
I	0

Greedy Search: Tree Search



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

Greedy Search: Time and SpaceComplexity ?



- Greedy search is not optimal.
- Greedy search is incomplete without systematic checking of repeated states.
- 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

A* Search

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

A* (A Star)

- Greedy Search minimizes a heuristic h(n) which is an estimated cost from a node n to the goal state. Greedy Search is efficient but it is not 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* (A Star)

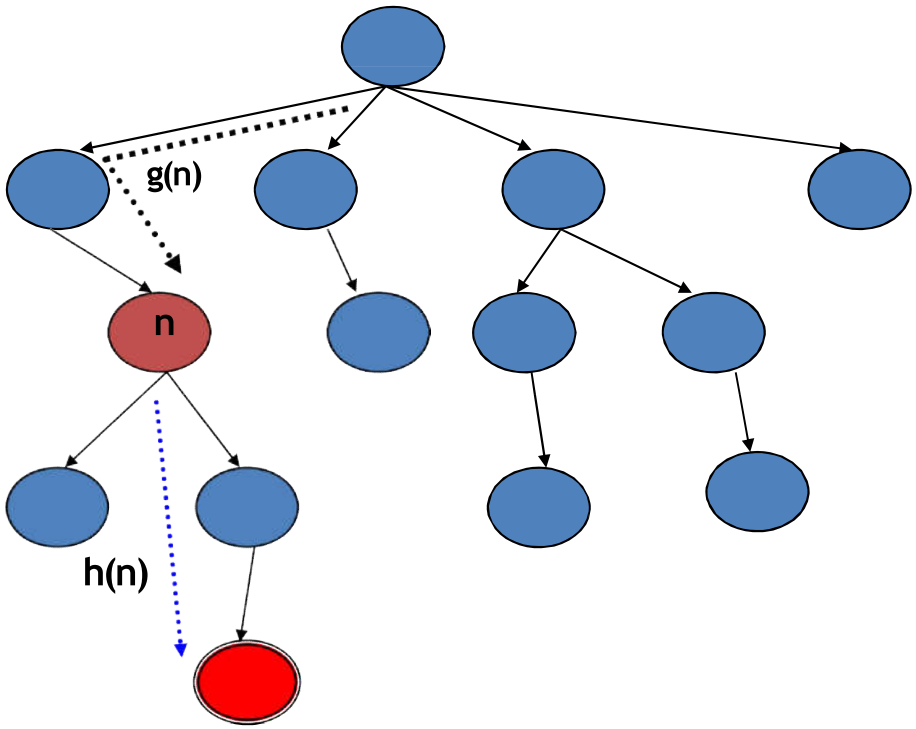
- A* uses a heuristic function which combines g(n) and h(n): $f(n) = g(n) + h(n)$

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

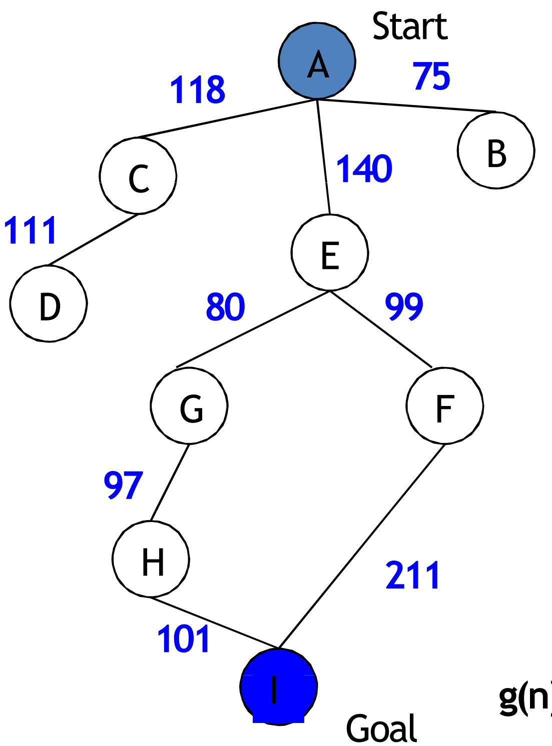
- **h(n)** is an estimation of the remaining cost to reach the goal.

A* (A Star)

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



A* Search



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

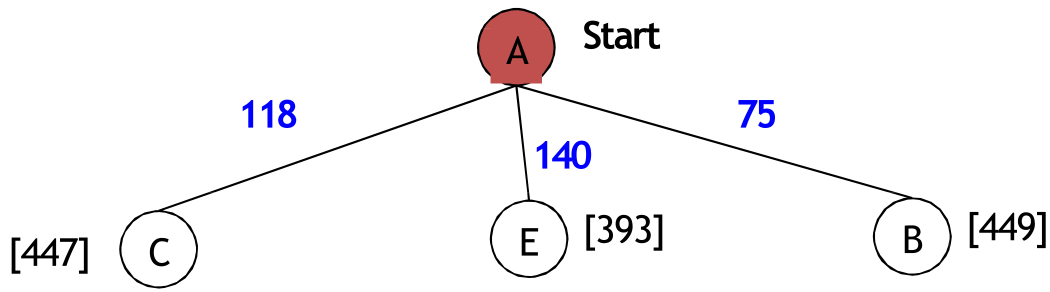
$$f(n) = g(n) + h(n)$$

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

A* Search: TreeSearch

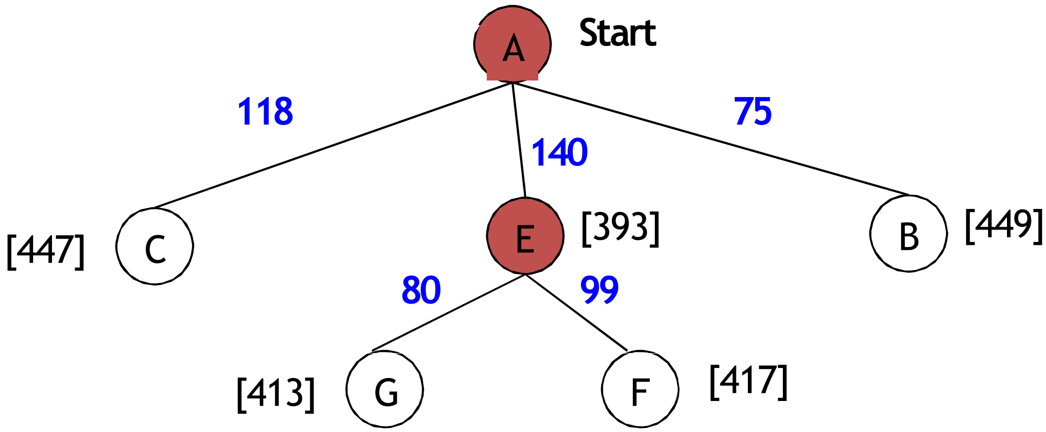
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* Search: TreeSearch



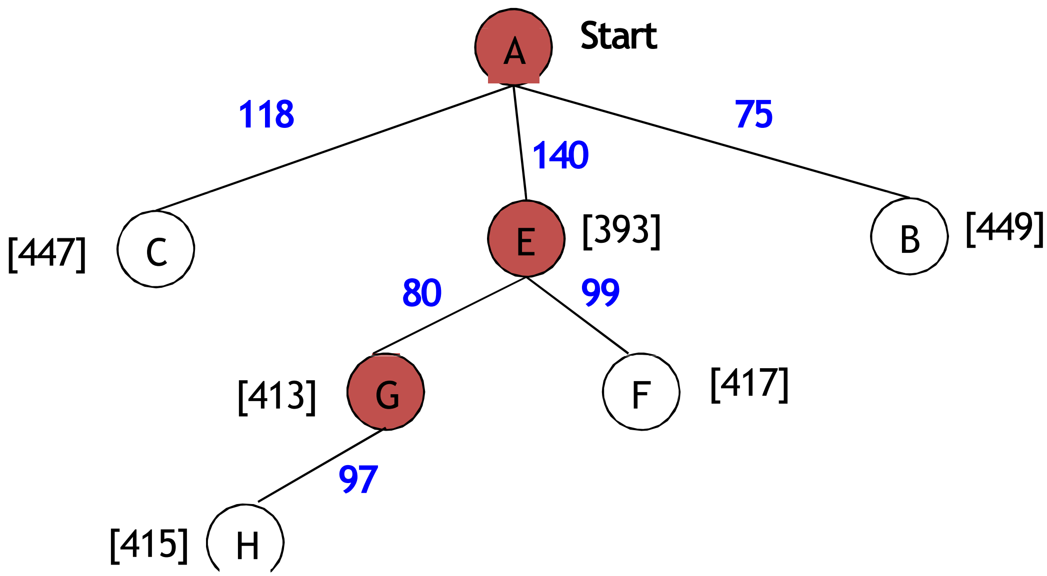
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* Search: TreeSearch



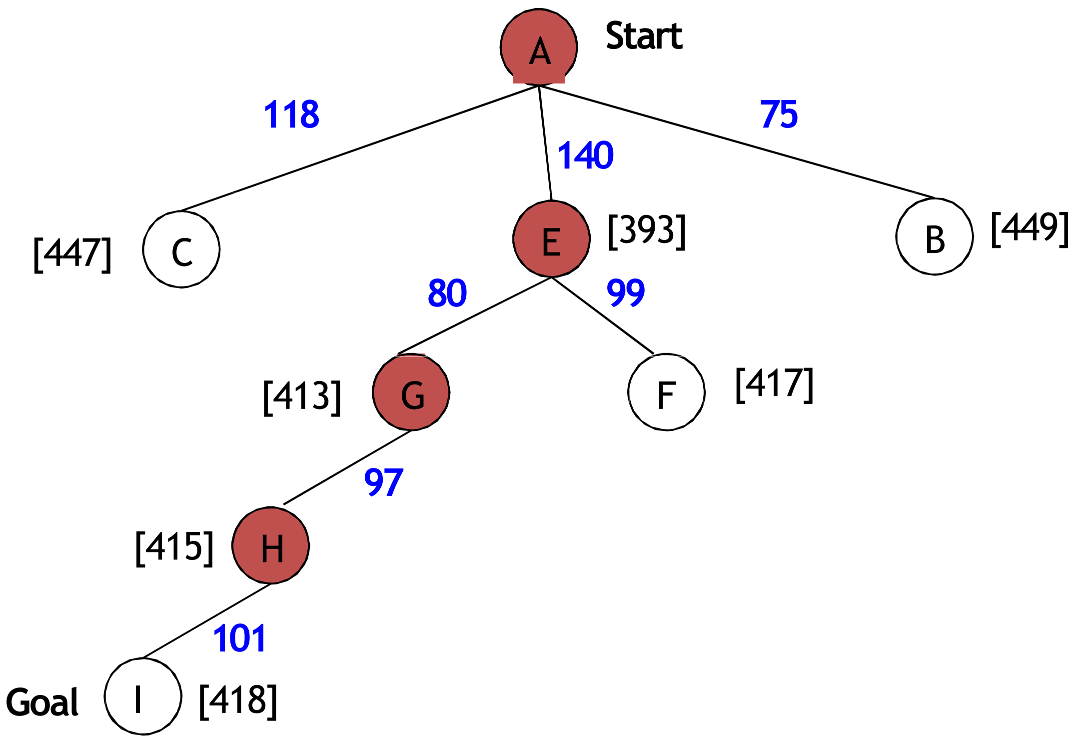
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* Search: TreeSearch



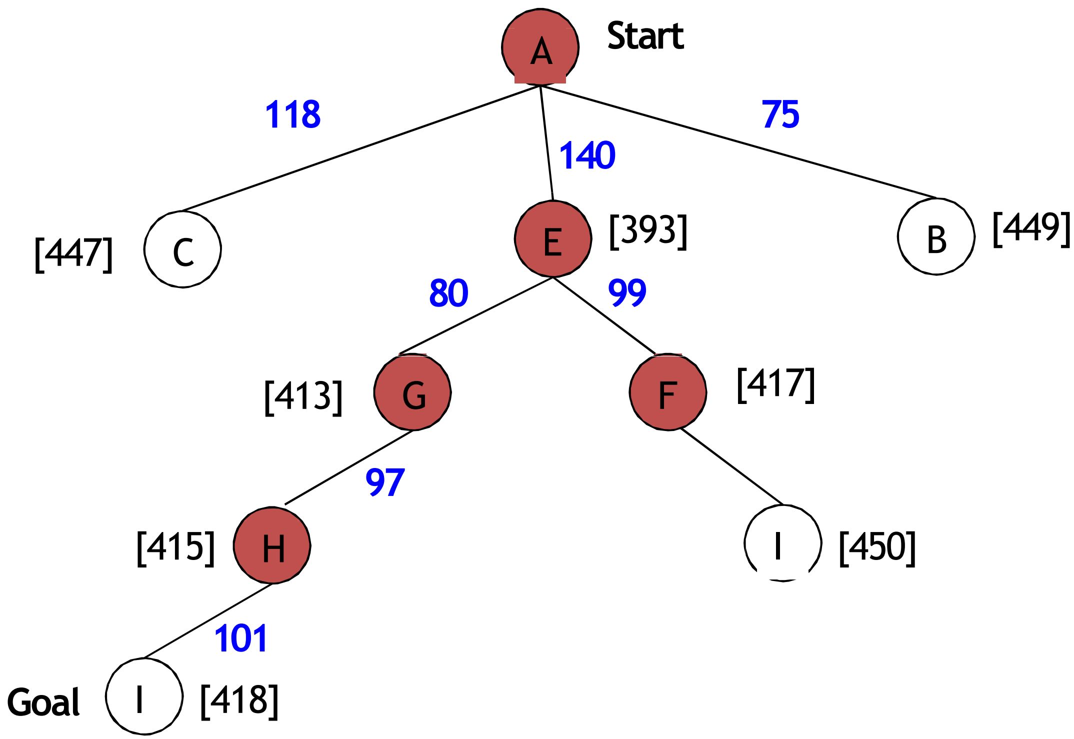
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* Search: TreeSearch



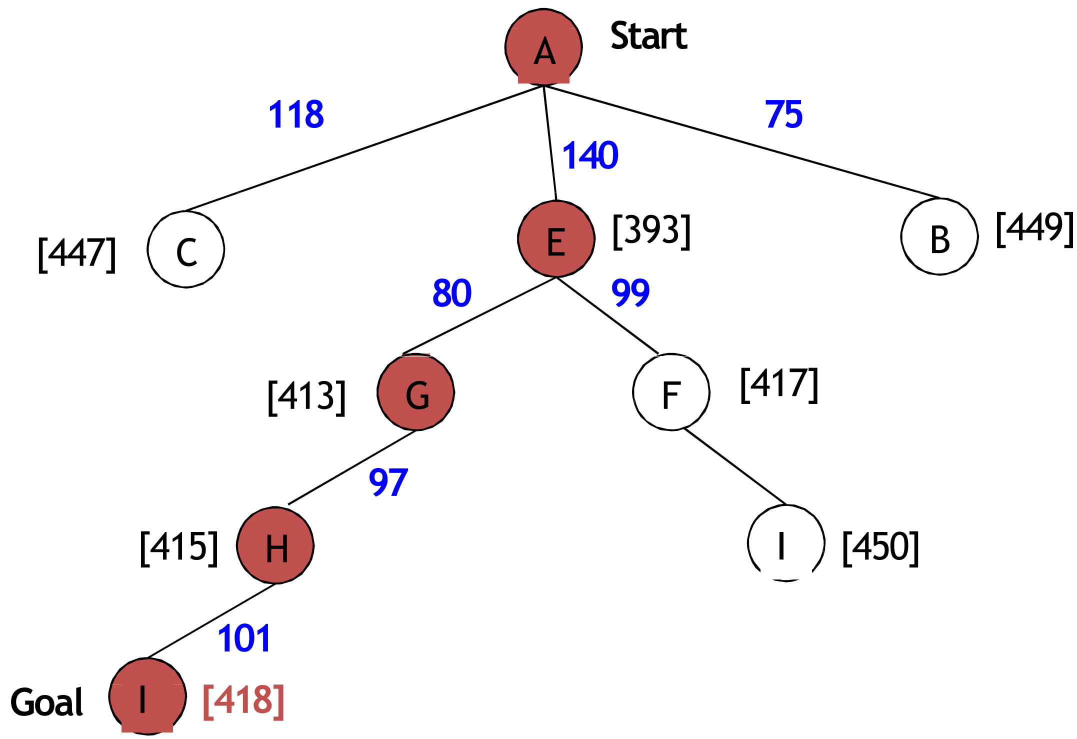
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A * Search: TreeSearch



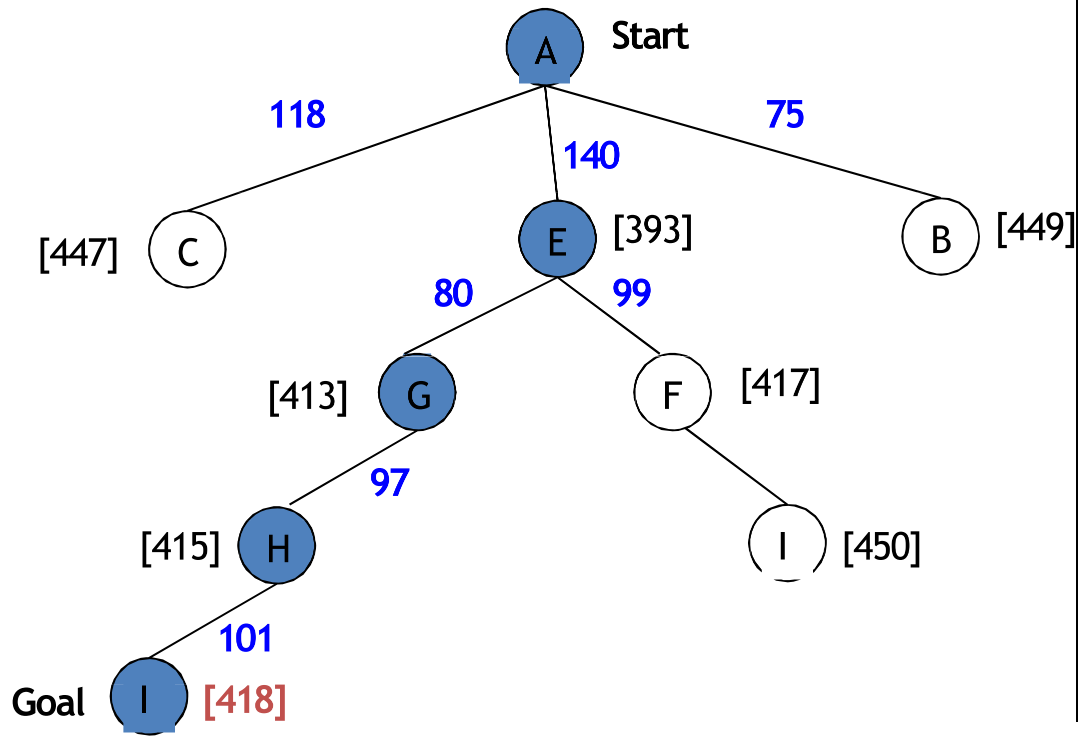
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* Search: TreeSearch



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

A* Search: TreeSearch

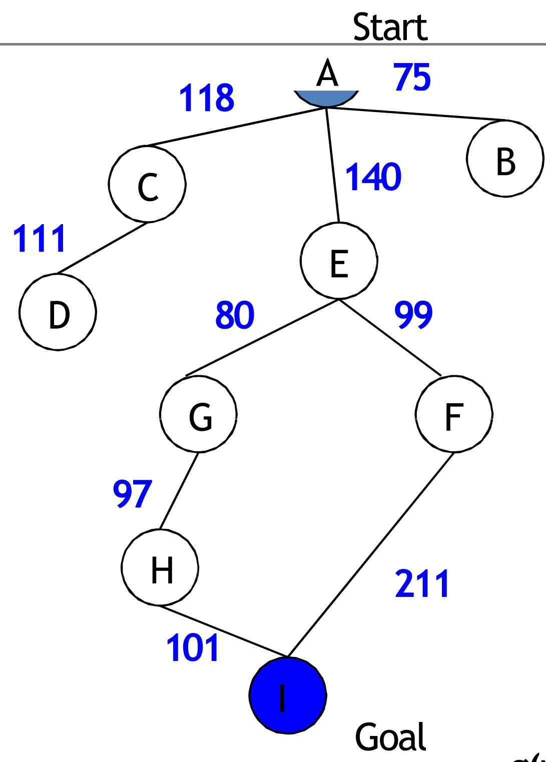


State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

A* with f() not Admissible

- h() overestimates the cost to reach 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 the true cheapest cost from n to the goal.
- A* is admissible if it uses an admissible heuristic, and h(goal) = 0.
- If the heuristic function, h always underestimates the true cost (h(n) is smaller than h*(n)), then A* is guaranteed to find an optimal solution.

A* Search: *h* not admissible !



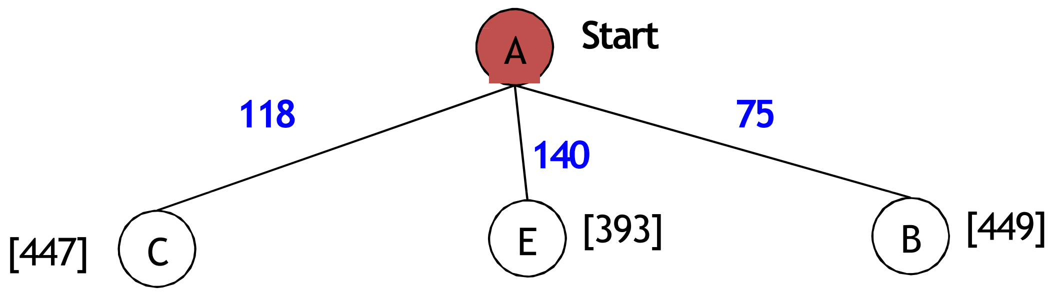
State	Heuristic: h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	**138**
I	0

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

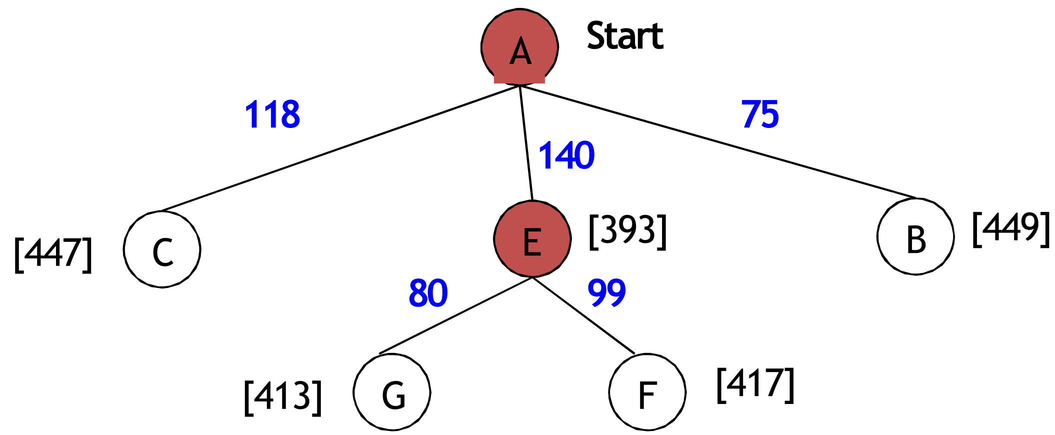
g(n): is the exact cost to reach node *n* from the initial state.

A* Search: TreeSearch

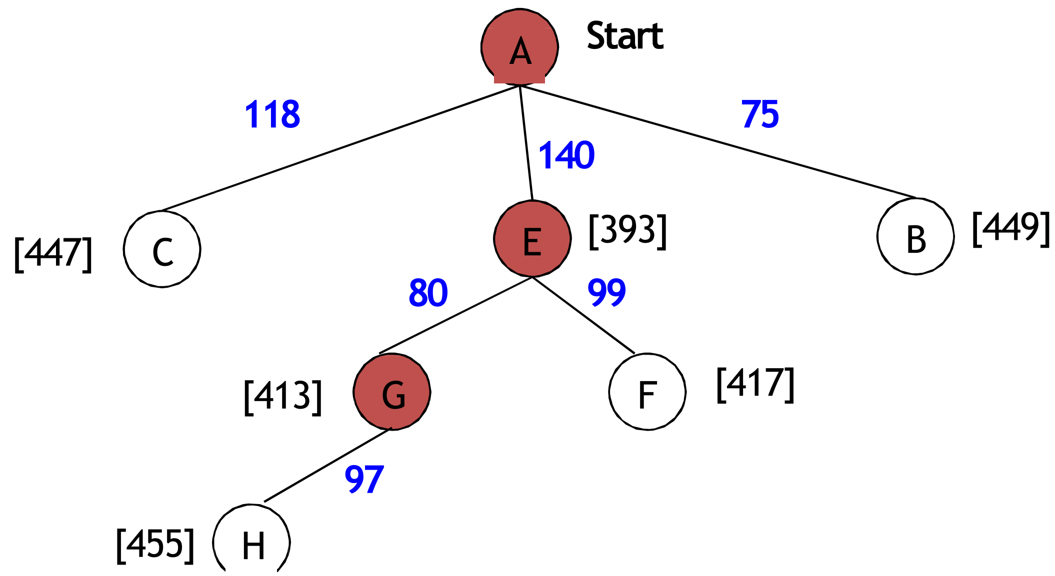
A* Search: TreeSearch



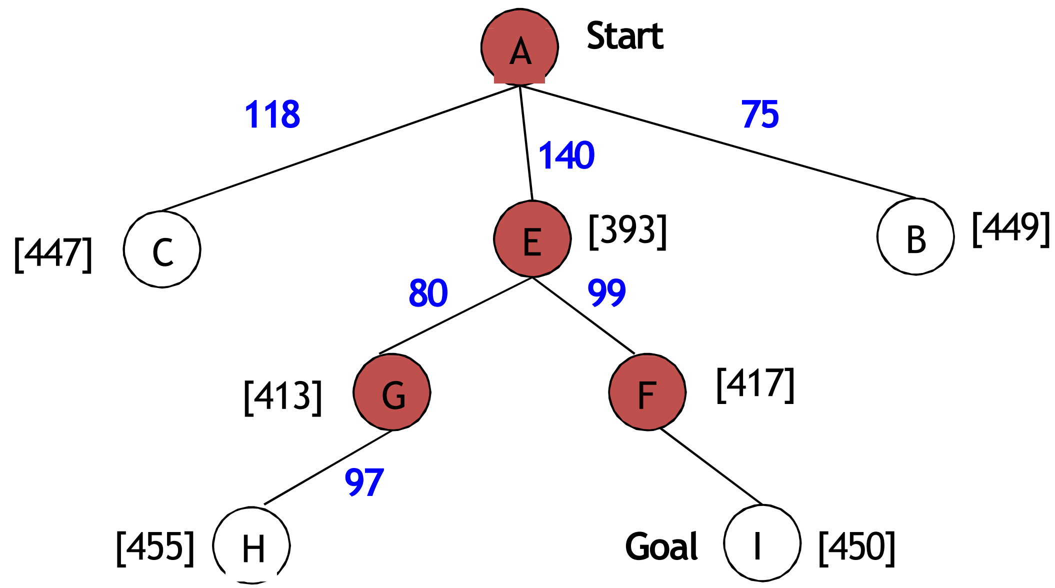
A* Search: TreeSearch



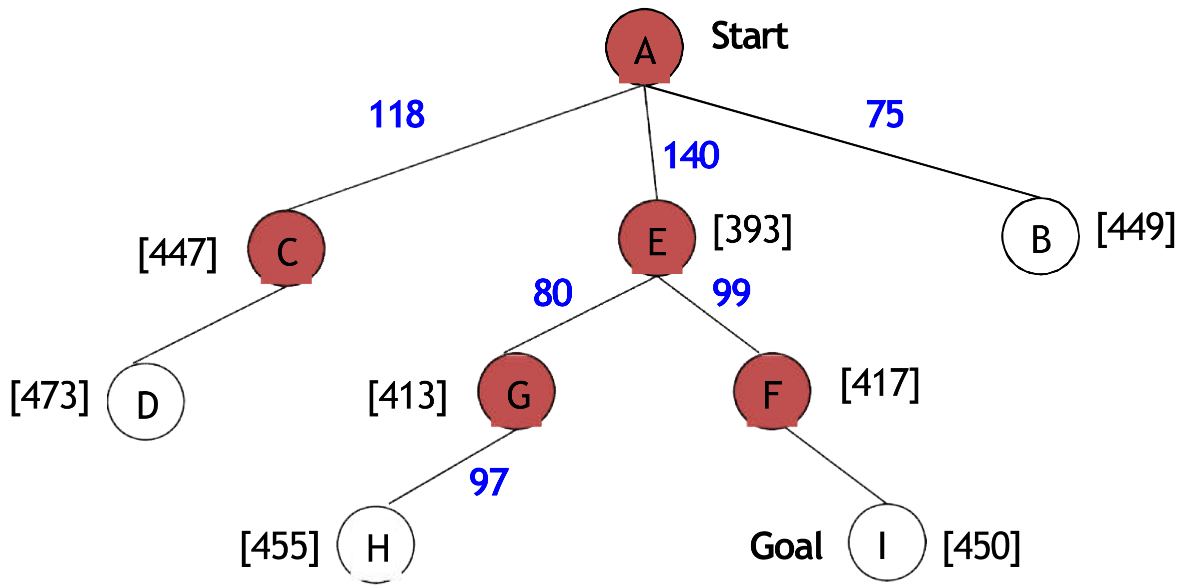
A* Search: TreeSearch



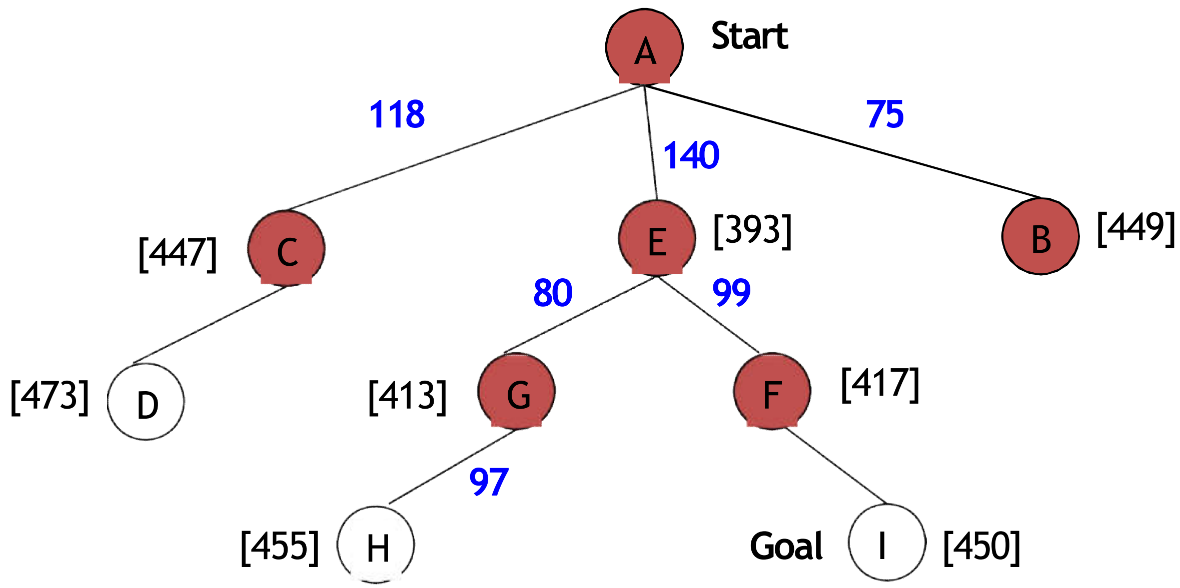
A* Search: TreeSearch



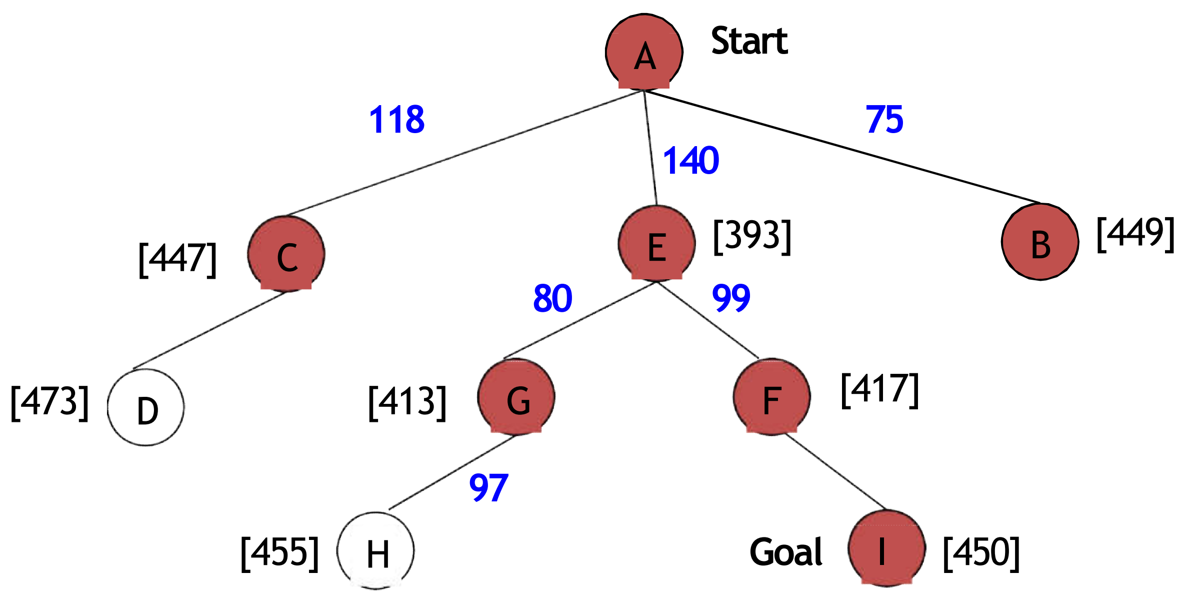
A* Search: TreeSearch



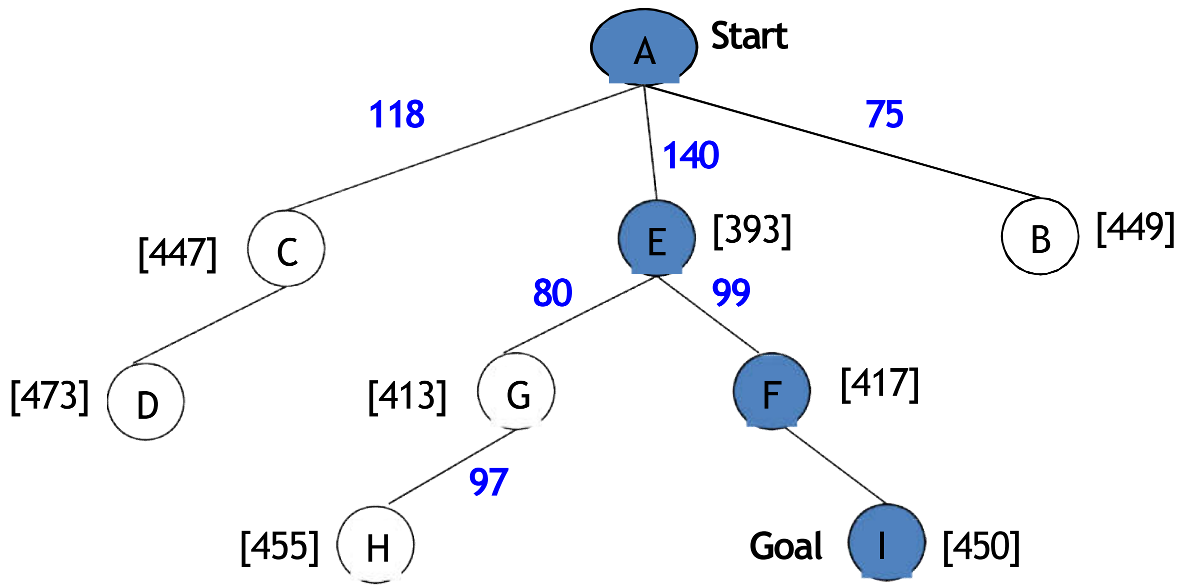
A* Search: TreeSearch



A* Search: TreeSearch



A* Search: TreeSearch



A* not optimal !!!

A* Search

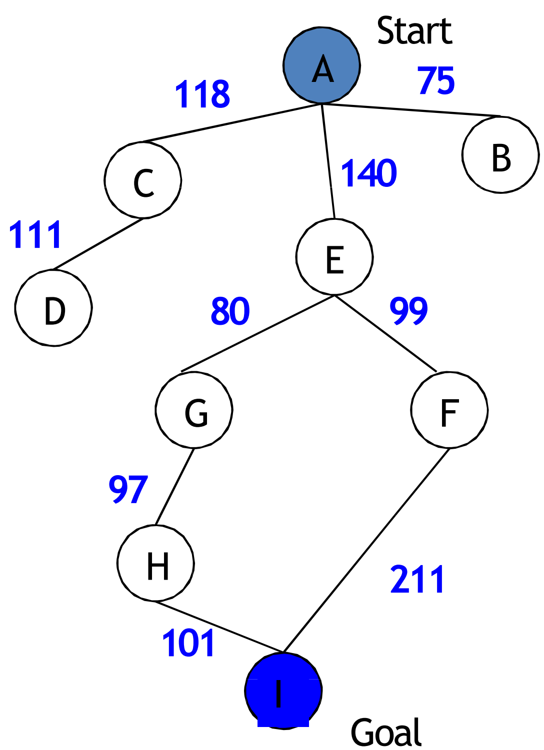
A* with systematic checking for repeated states …

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 the 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) + cost(n,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.

A* Search:Analysis

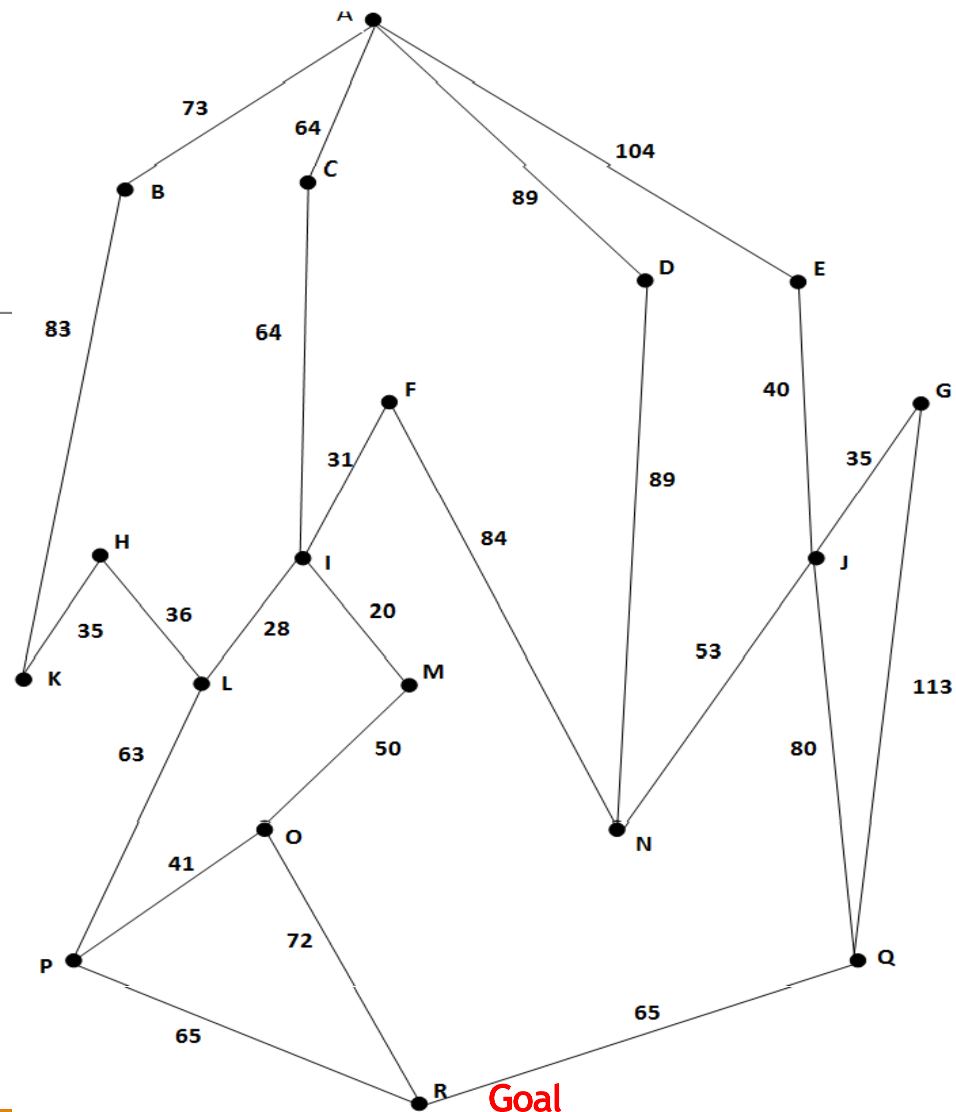


- A* is complete except if there is an infinity of nodes with f < f(G).
- A* is optimal if heuristic *h* is admissible.
- Time complexity depends on the quality of heuristic but is still exponential.
- For space complexity, A* keeps all nodes in memory. A* has worst case $O(b^d)$ space complexity, but an iterative deepening version is possible (IDA*).

Conclusions

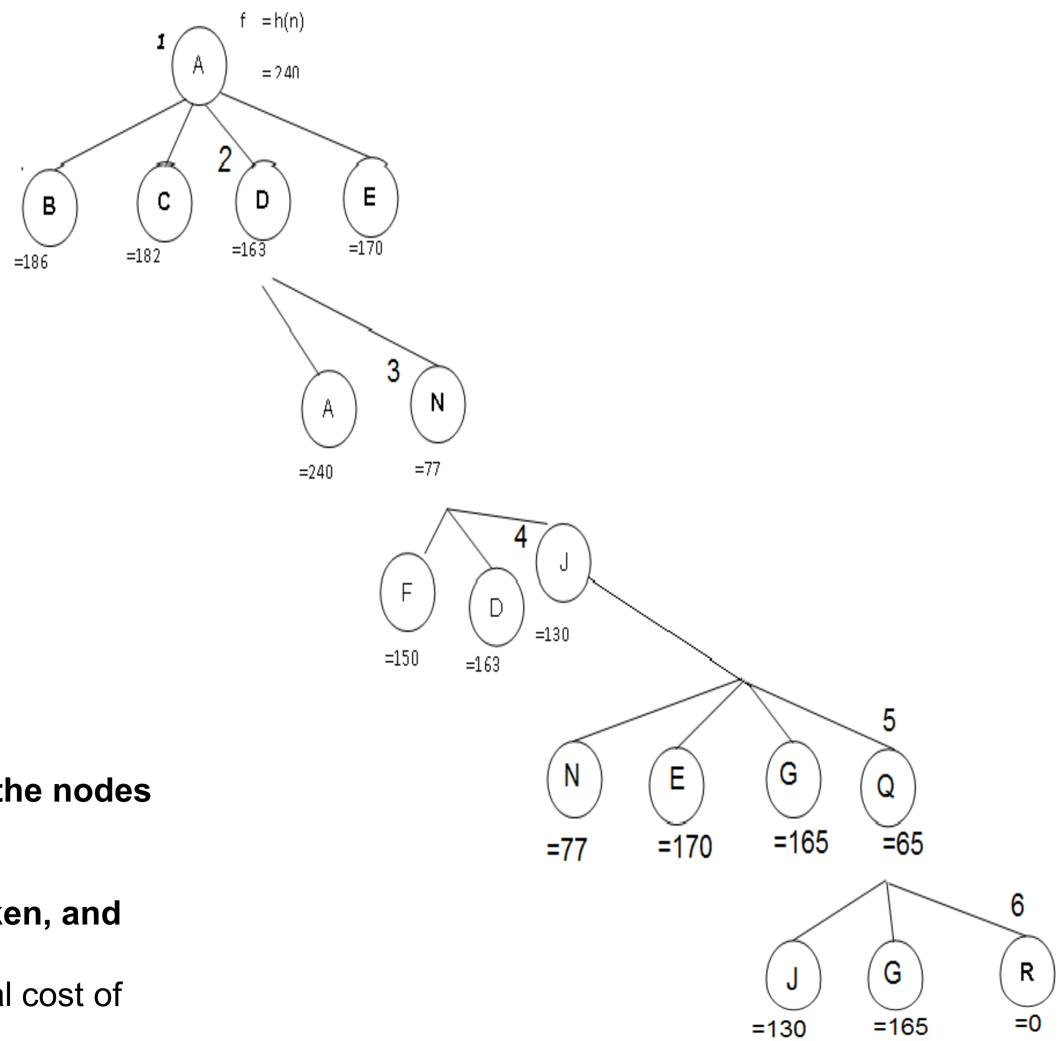
- Frustration with *uninformed* search led to the idea of using domain-specific knowledge in a search so that one can intelligently explore only the relevant part of the search space that has a good chance of containing the goal state. These new techniques are called informed (heuristic) search strategies.
- Even though heuristics improve the performance of informed search algorithms, they are still time-consuming, especially for large-size instances.

Exercise



A	240
B	186
C	182
D	163
E	170
F	150
G	165
H	139
I	120
J	130
K	122
L	104
M	100
N	77
O	72
P	65
Q	65
R	0

The Search Tree Using Greedy best first Algorithm



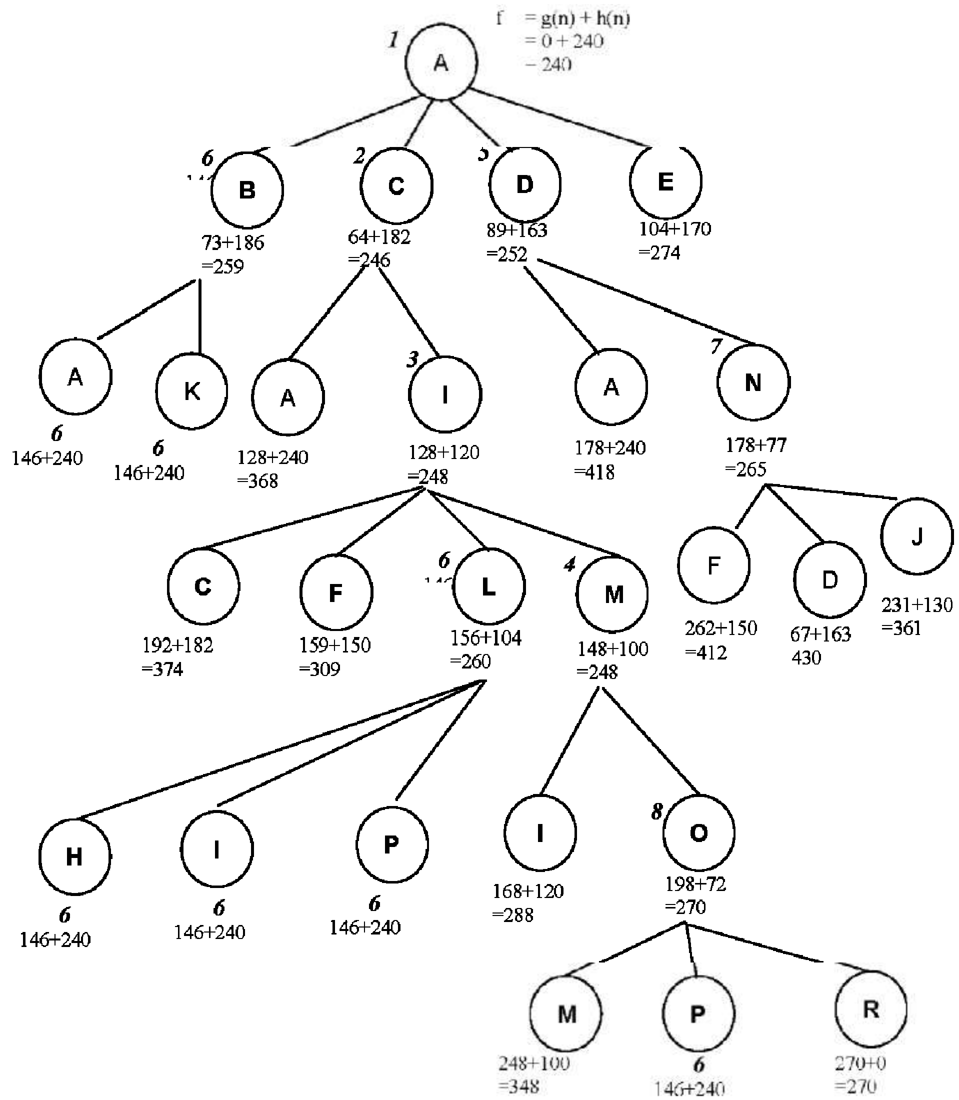
State the order in which the nodes were expanded

A, D, N, J, Q, R

State the route that is taken, and give the total cost

A, D, N, J, Q, R, with a total cost of 376

The Search Tree Using A*



State the order in which the nodes were expanded

A, C, I, M, D, B, N, O, L

State the route that is taken, and give the total cost

A, C, I, M, O, R, with a total cost of 270