Manara University
Faculty of Engineering
Department of IT

Introduction to Artificial Intelligence

Problem Solving and Search part II

Lecture 4

Prof Dr. Eng. Mariam M. Saii

Outline

- Best-first search
- A* search
- Heuristics
- Local search algorithms
- Hill-climbing search

Knowledge and Heuristics

الاجتهاد أوالاستكشاف أو الاستدلال:

هو دراسة قواعد وطرق الاكتشاف discovery والاختراع Invention استناداً إلى كلمة العالم أرخميدس الشهيرة Eureka والتي تعني بالعربية "وجدتها".

أما بالنسبة للبحث في البيانات فإن الاجتهاد يعني قواعد اختيار الفروع في فضاء الحالة للوصول إلى حل معقول للمسألة.

*intelligence comes from **heuristics** that help find promising states fast.*

Best-first search

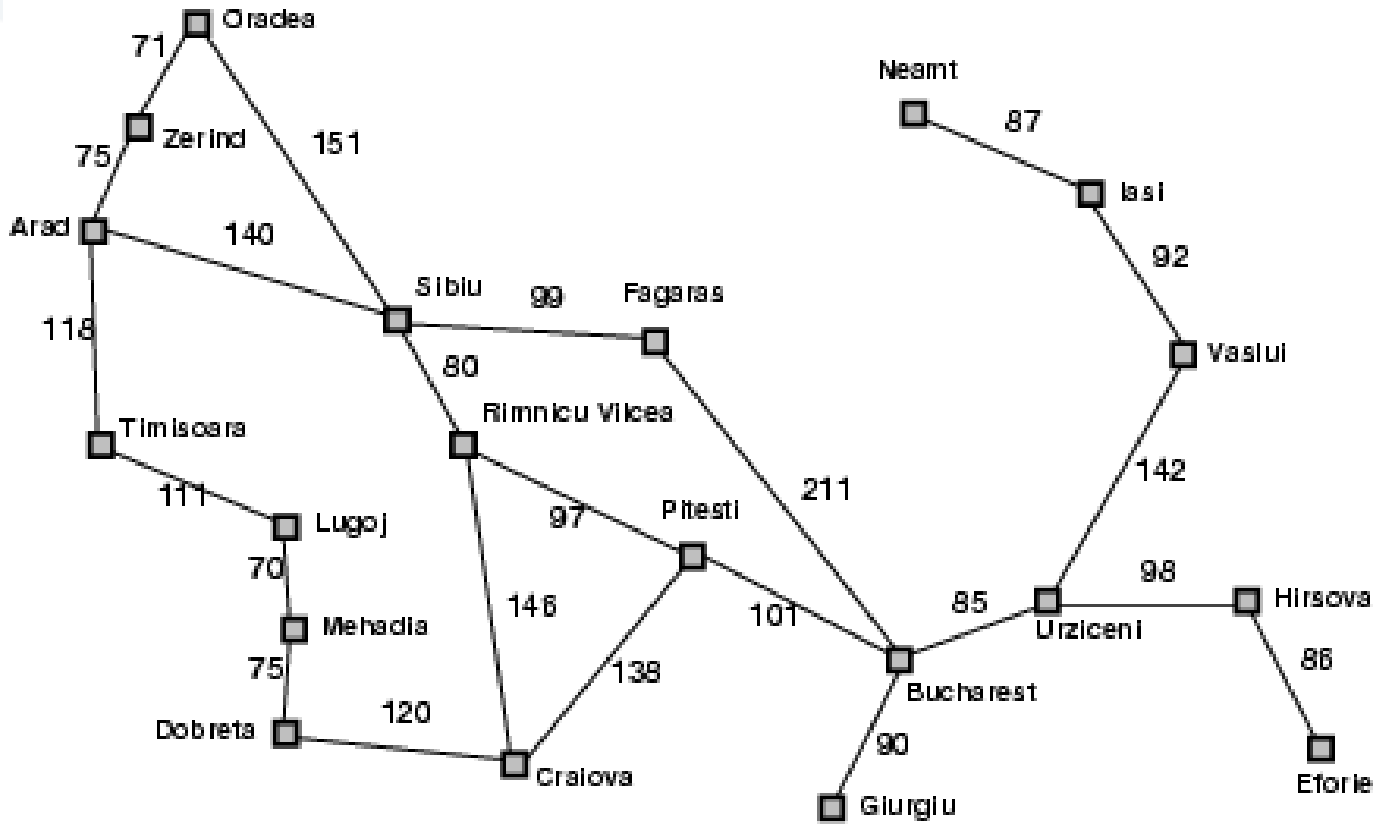
- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability"

  →Expand most desirable unexpanded node

  →
- Implementation:

  Order the nodes in frontier in decreasing order of desirability
- Special cases:
  - greedy best-first search
  - A* search
  -

Romania with step costs in km



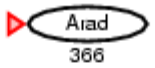
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

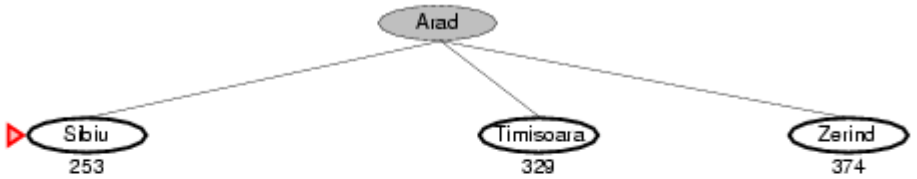
- Evaluation function
  - $f(n) = h(n)$ (heuristic)
  - = estimate of cost from n to *goal*

- e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal
-

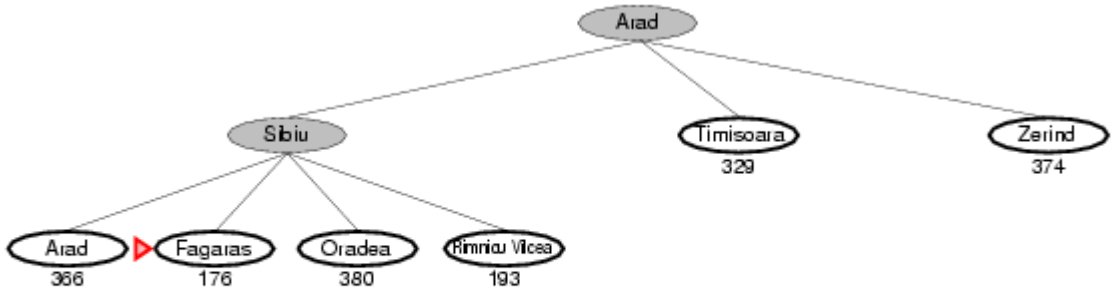
Greedy best-first search example



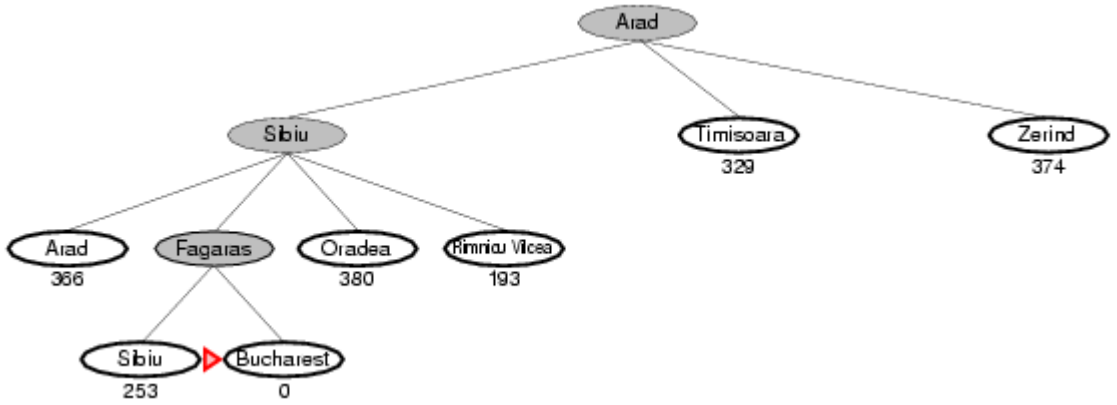
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



http://aispace.org/search/

Properties of greedy best-first search

- Complete? No – can get stuck in loops,
  - e.g. as Oradea as goal
    - Iasi → Neamt → Iasi → Neamt →
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ -- keeps all nodes in memory
- Optimal? No
-

A* search

- **Idea**: avoid expanding paths that are already expensive.
- Very important!

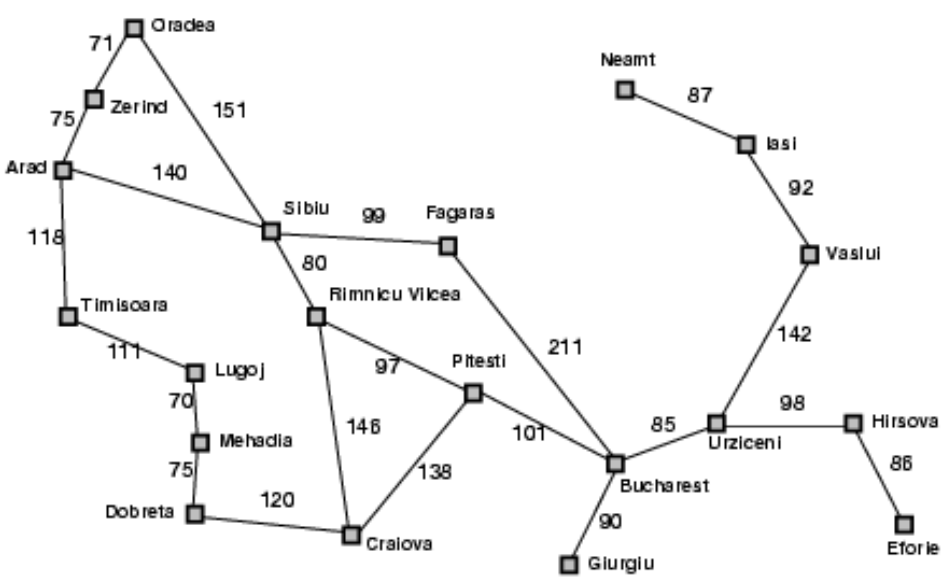
- Evaluation function $f(n) = g(n) + h(n)$

  $g(n)$ = cost so far to reach n

  $h(n)$ = estimated cost from n to goal

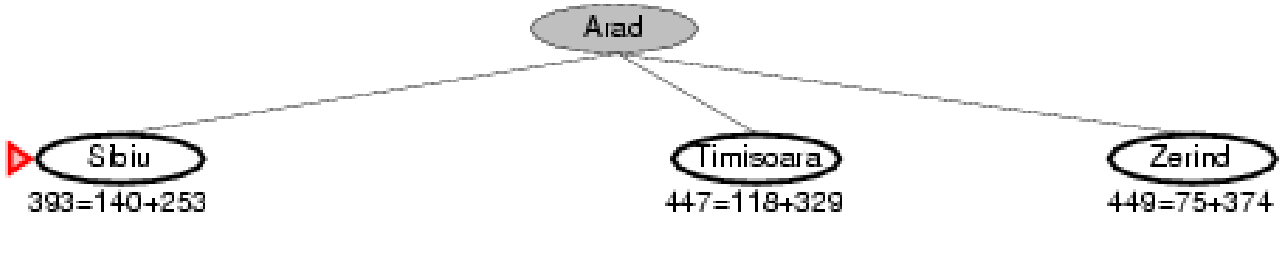
- ***f(n)* = estimated total cost of path through *n* to goal**

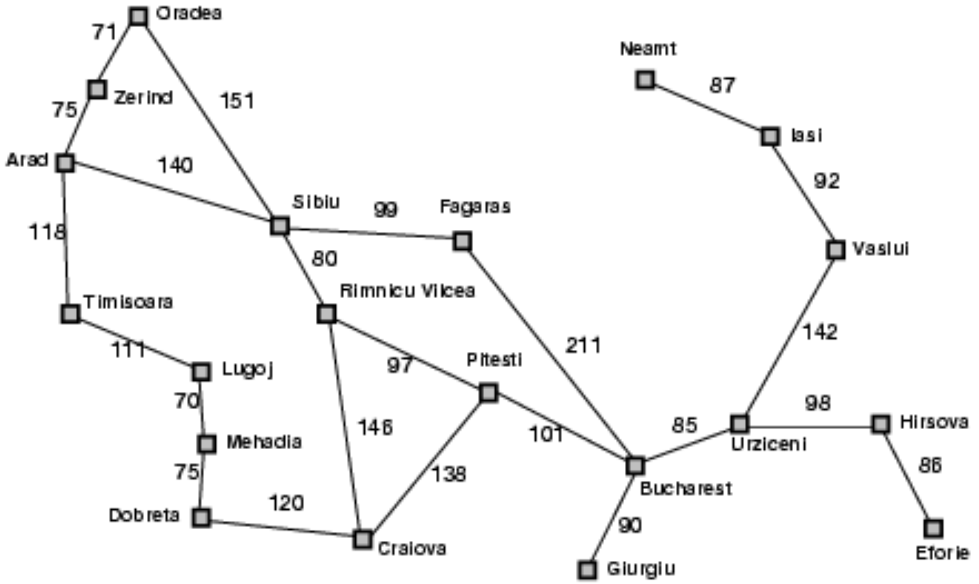
A* search example



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

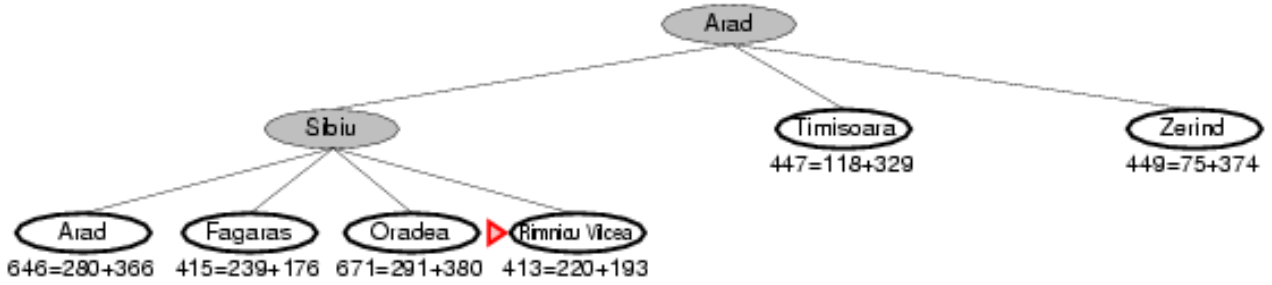
A* search example

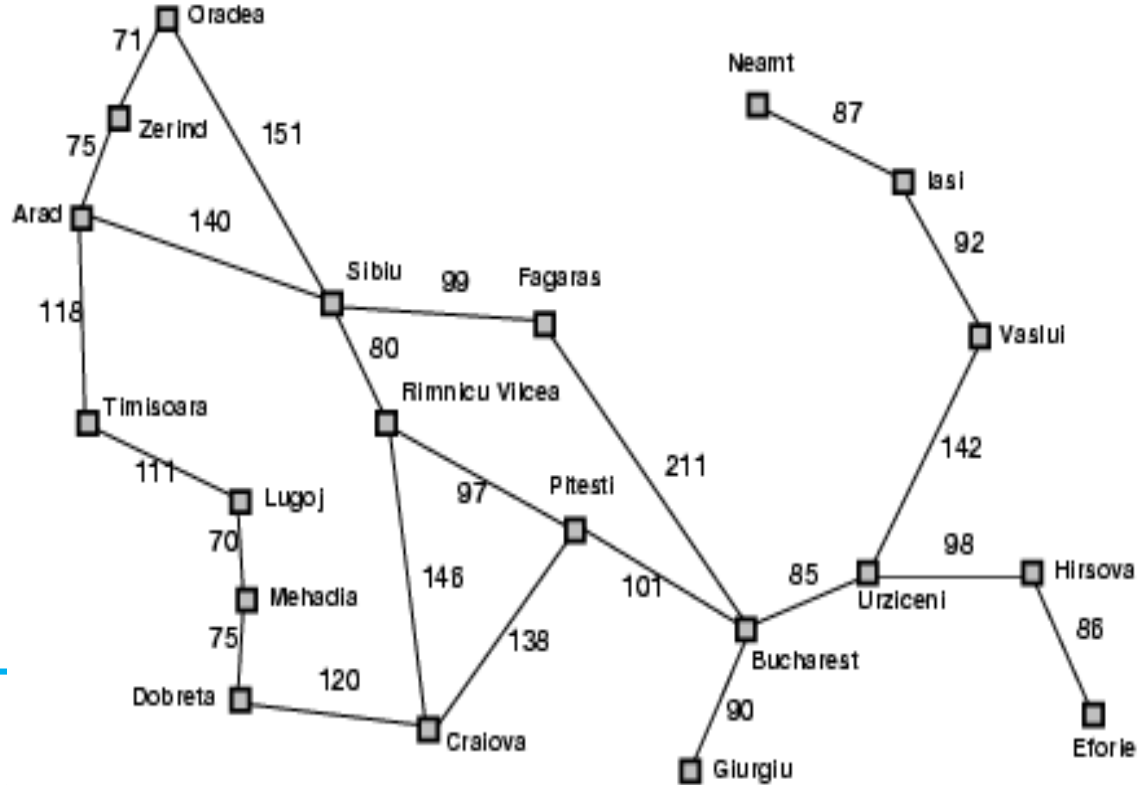




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

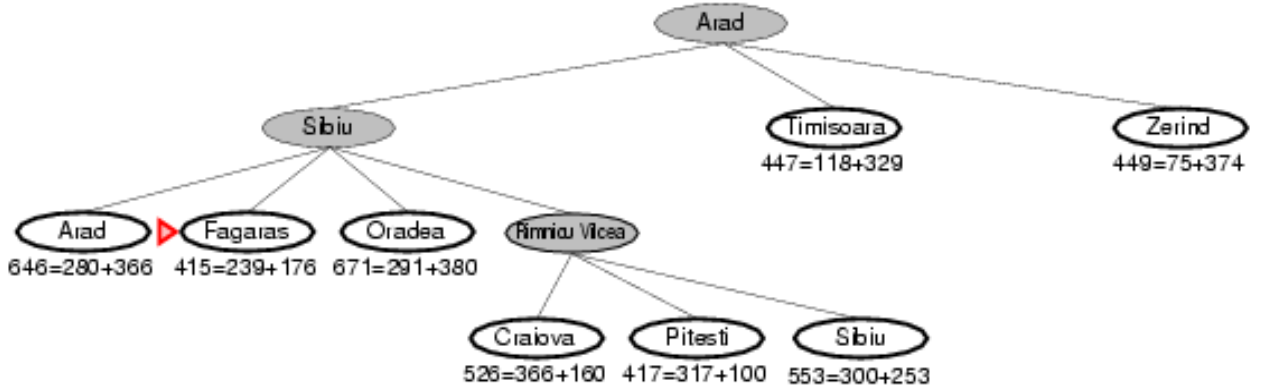


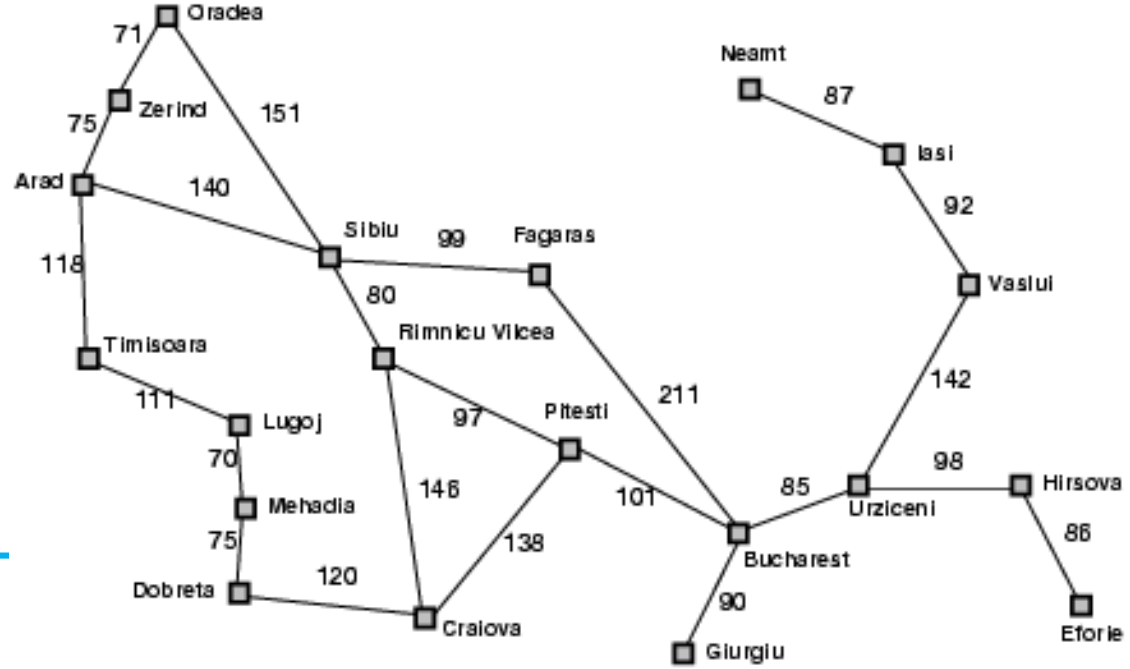


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

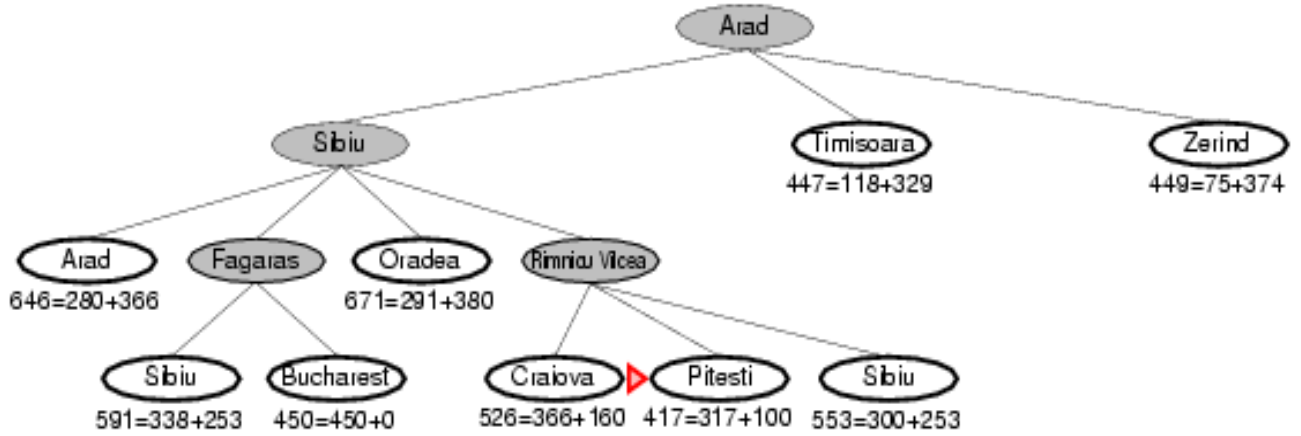
A* search example

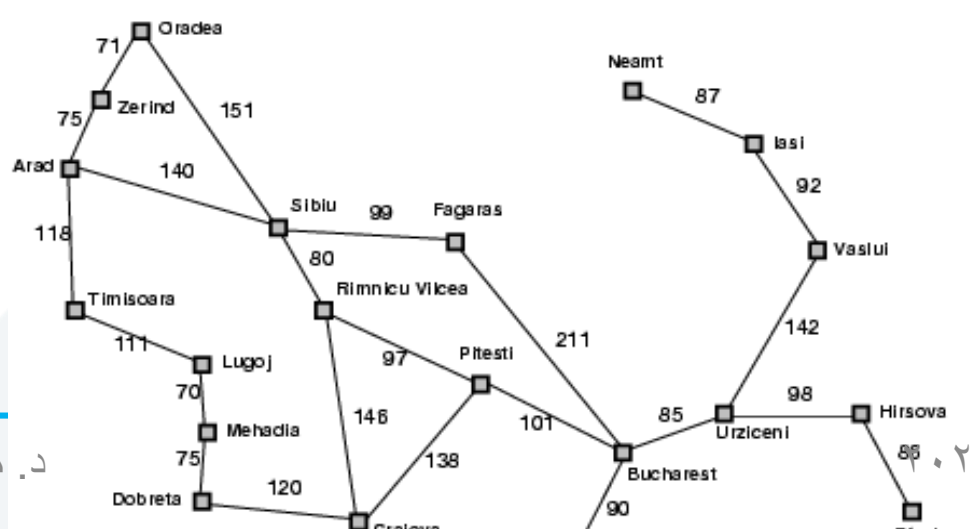




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

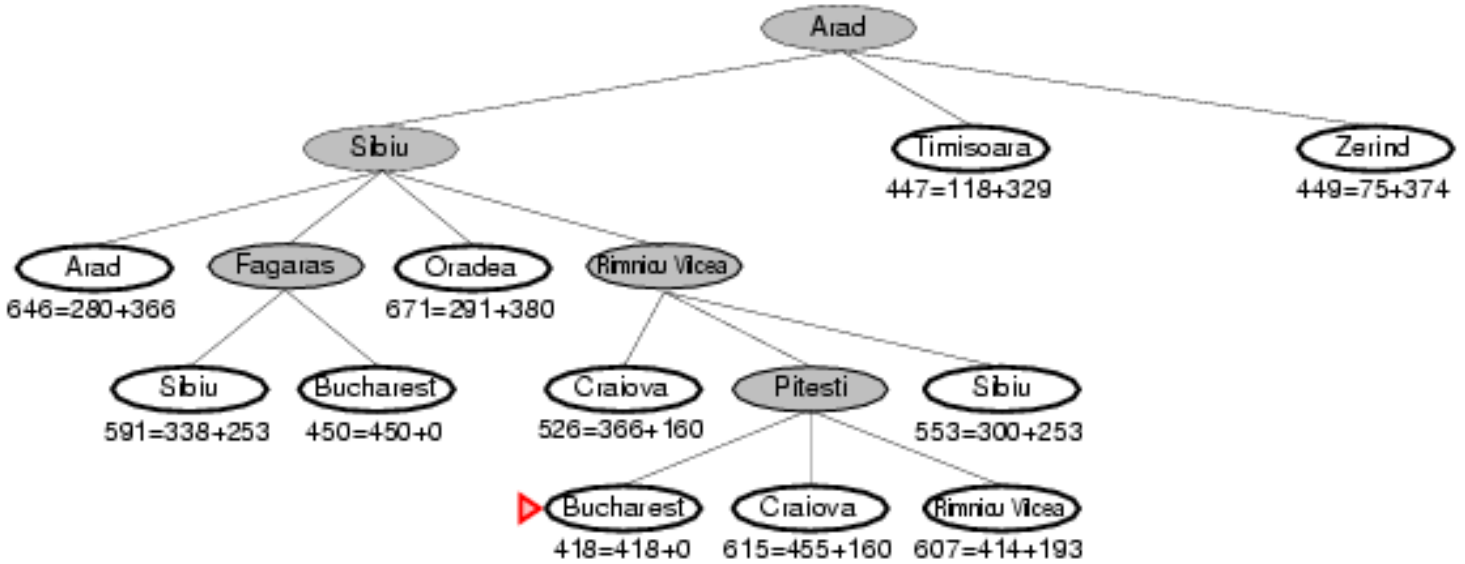
A* search example





Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199

A* search example



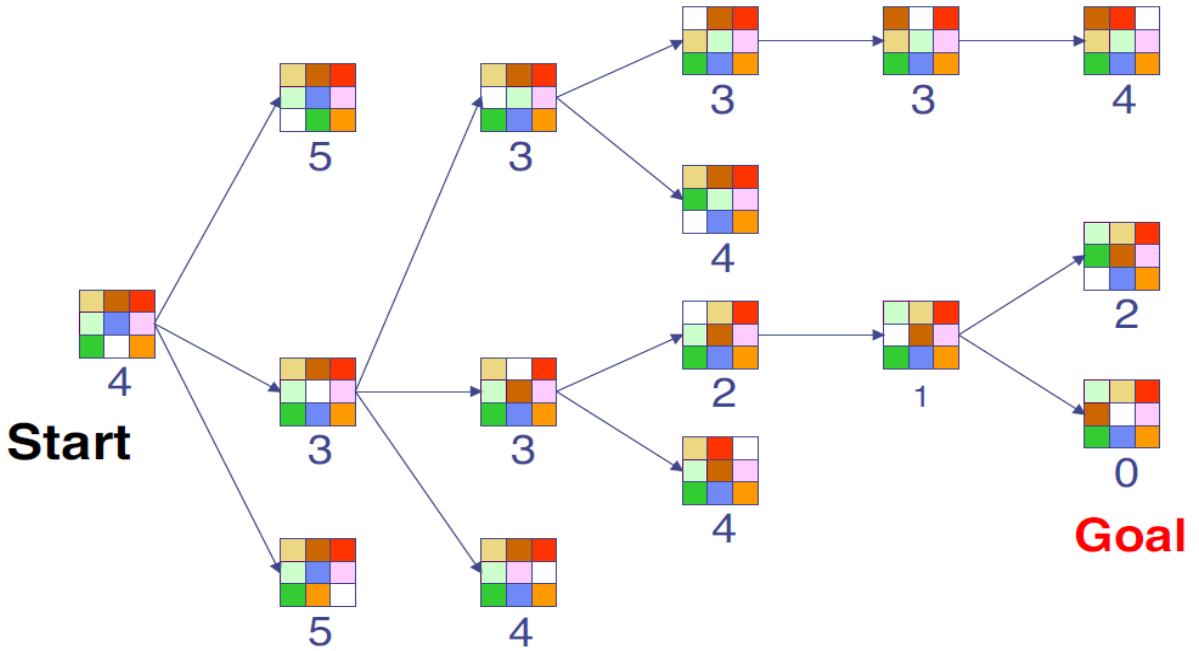
http://aispace.org/search/

- We stop when the node with the lowest f-value is a goal state.
- Is this guaranteed to find the shortest path?

greedy

8-Puzzle

f(N) = h(N) = number of misplaced numbered tiles



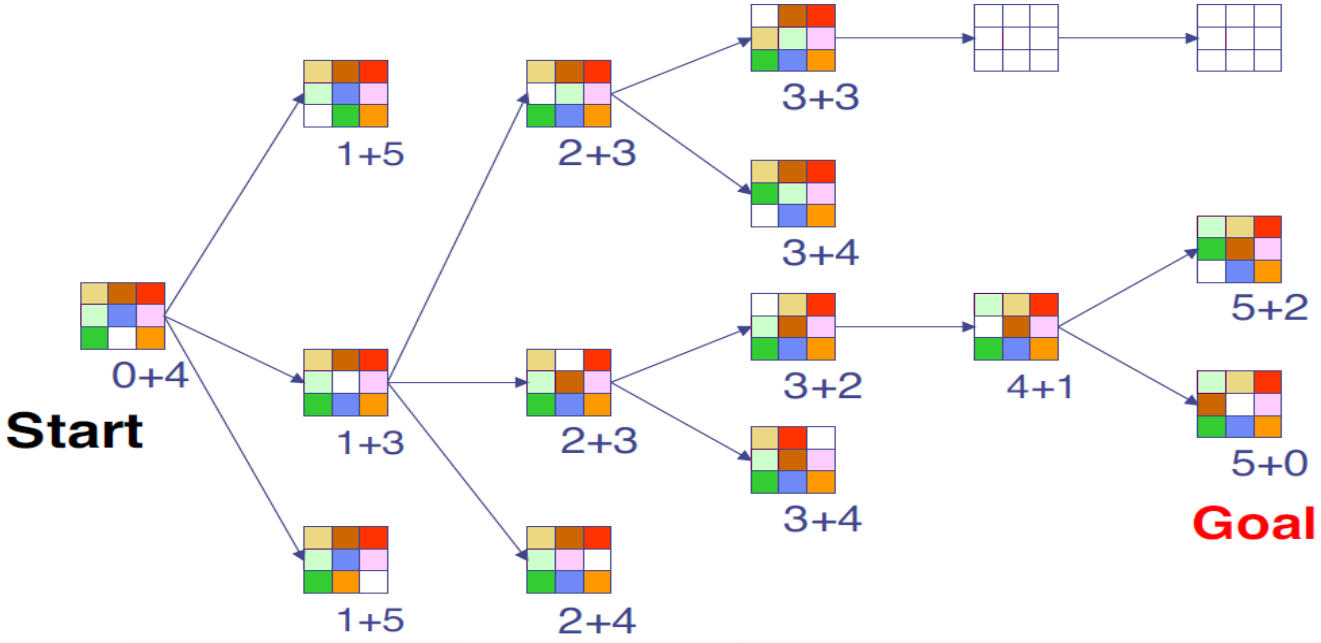
The white tile is the empty tile

A*

8-Puzzle

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

with h(N) = number of misplaced numbered tiles



Properties of A*

- Complete? Yes (unless there are infinitely many nodes with f $\leq f(G)$)
-
- Time? Exponential
-
- Space? Keeps all nodes in memory
-
- Optimal? Yes
-

- **Heuristic functions estimate costs of shortest paths**
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands **lowest h**
  - incomplete and not always optimal
- A∗ search expands lowest **g + h**
  - complete and optimal
  - also optimally efficient (up to tie-breaks)
- Admissible heuristics can be derived from exact solution of relaxed problems

Optimality

- Admissible heuristics
- Consistent heuristics (monotonicity)

Admissible heuristics

- A heuristic $h(n)$ 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.**
- **h(n)<=c(c,g)**

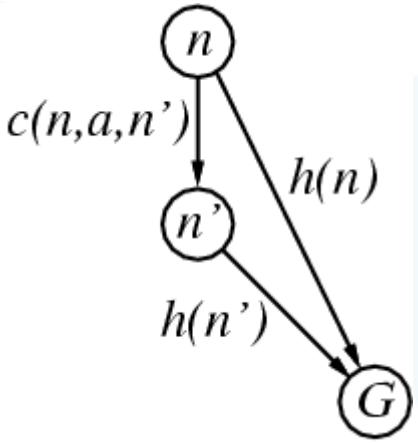
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)
- Negative Example: Fly heuristic: if wall is dark, then distance from exit is large.

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

Consistent heuristics

- A heuristic is consistent if for every node *n*, every successor *n'* of *n* generated by any action *a*,
-

h(n) ≤ c(n,a,n') + h(n')



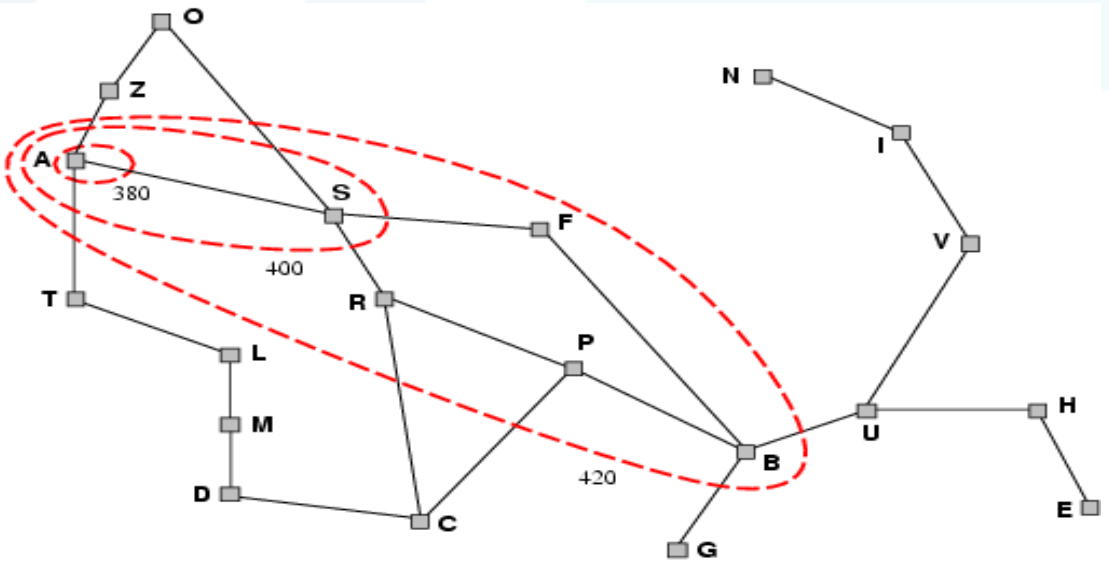
- Intuition: can't do worse than going through n'.
- If *h* is consistent, we have
-

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

- i.e., ***f(n)*** **is non-decreasing along any path.**
-
- **Theorem: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal**
-

Optimality of A*

- A* expands nodes in order of increasing *f* value
-
- http://aispace.org/search/
- Gradually adds "*f*-contours" of nodes
- Contour *i* has all nodes with $f=f_i$, where $f_i < f_{i+1}$

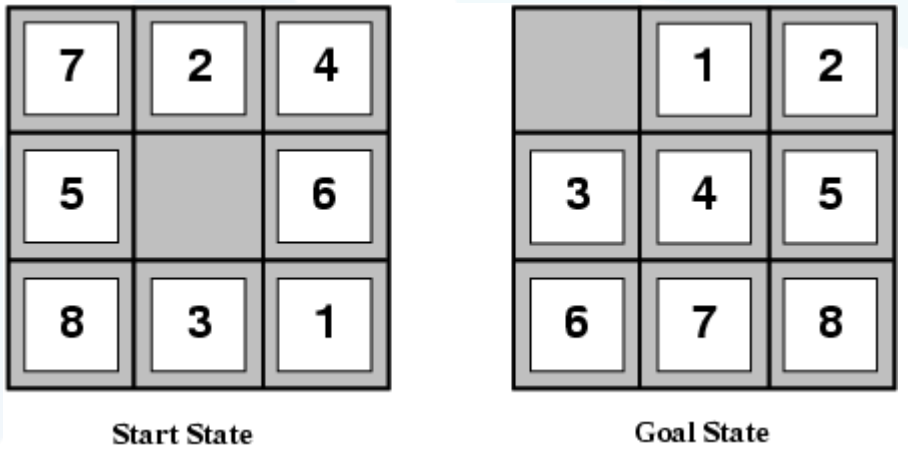


Admissible heuristics

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)



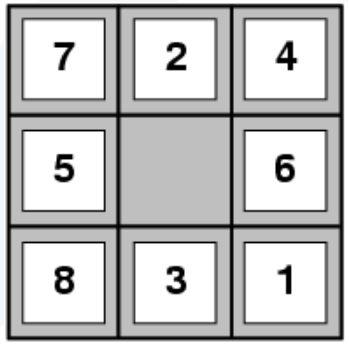
- $h_1(S) = ?$
- $h_2(S) = ?$

Admissible heuristics

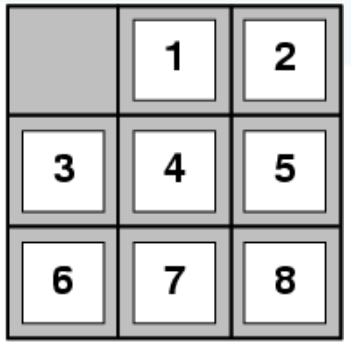
E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)

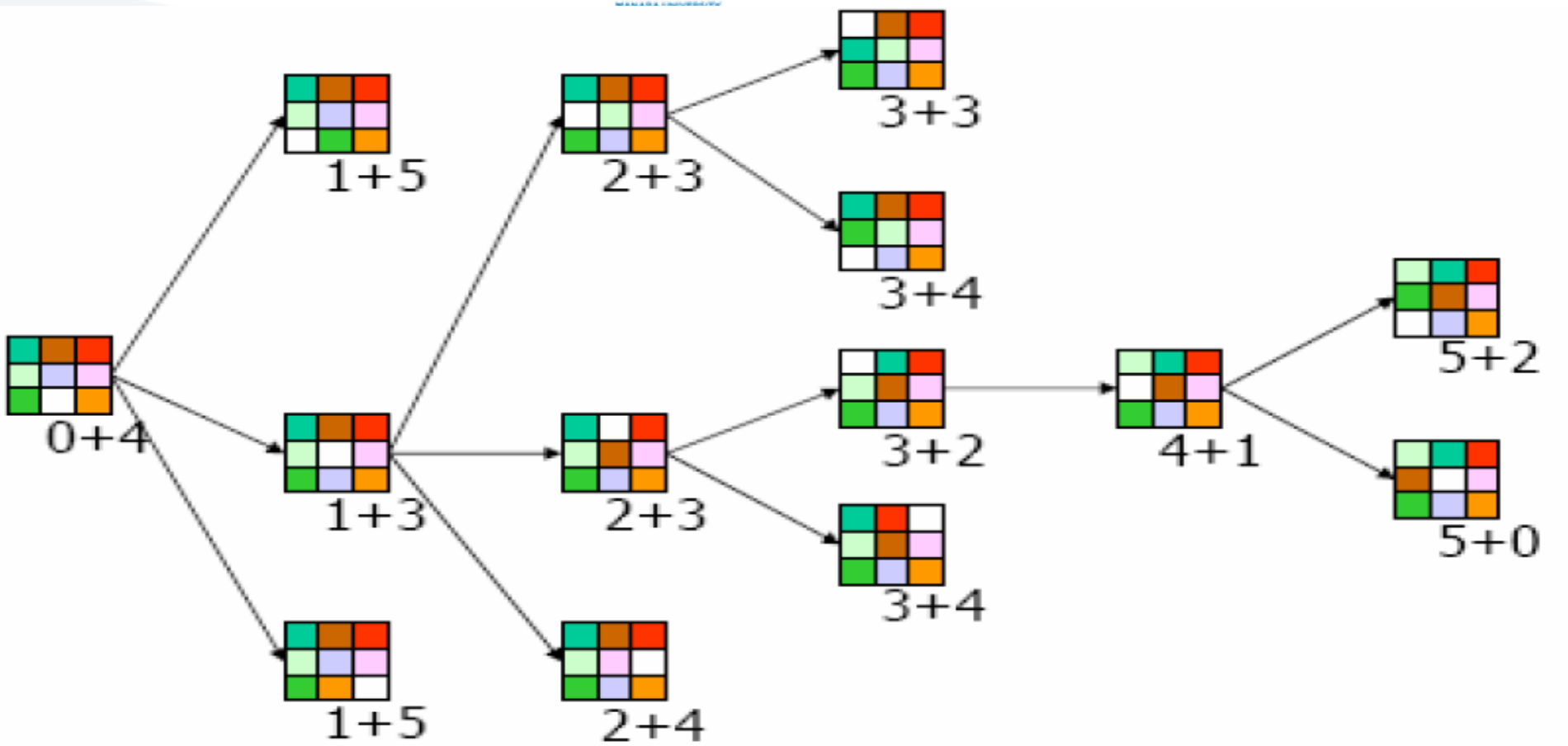


Start State



Goal State

- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18



F(n)=g(n)+h(n)

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
-

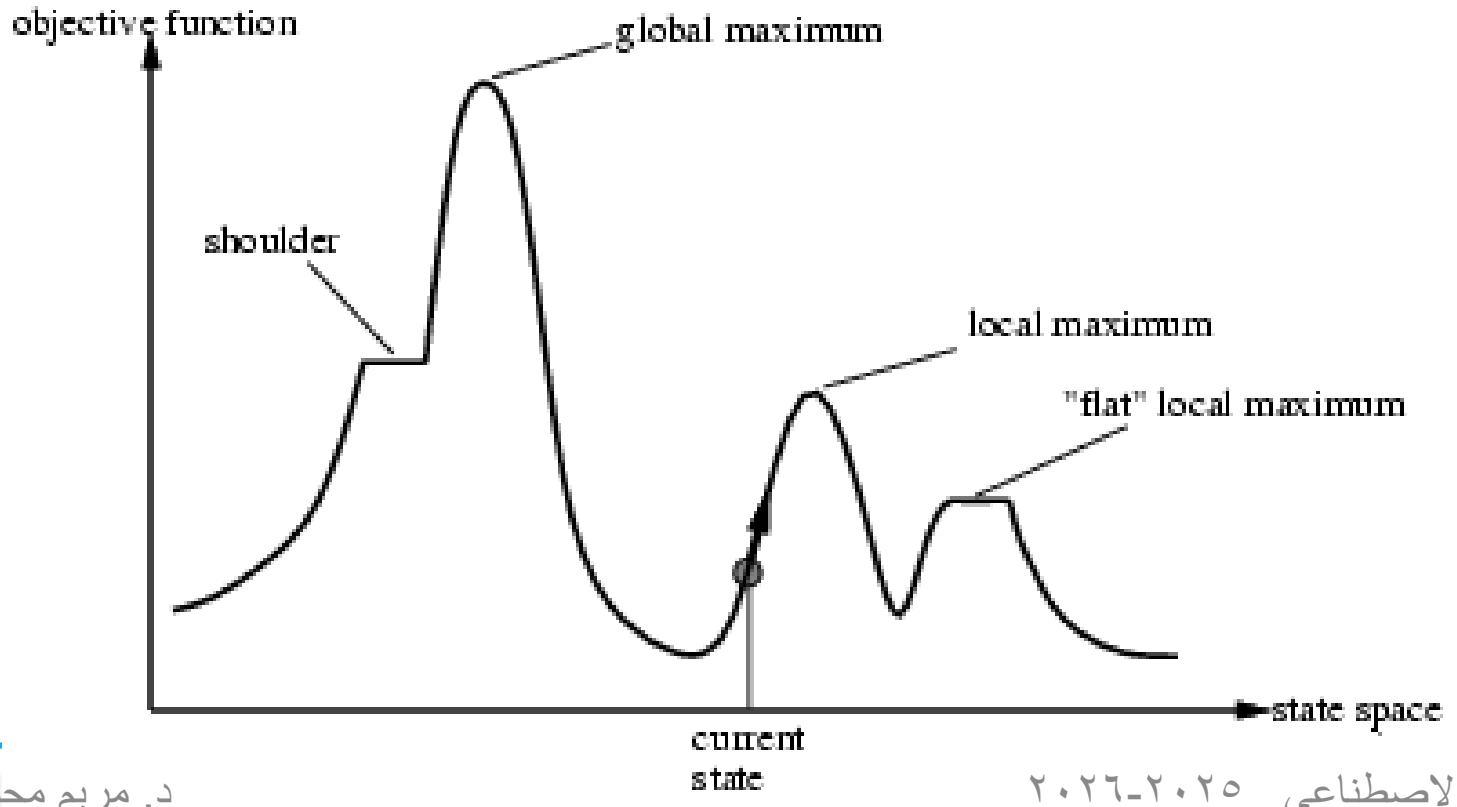
Hill-climbing search

- "Like climbing Everest in thick fog with amnesia"

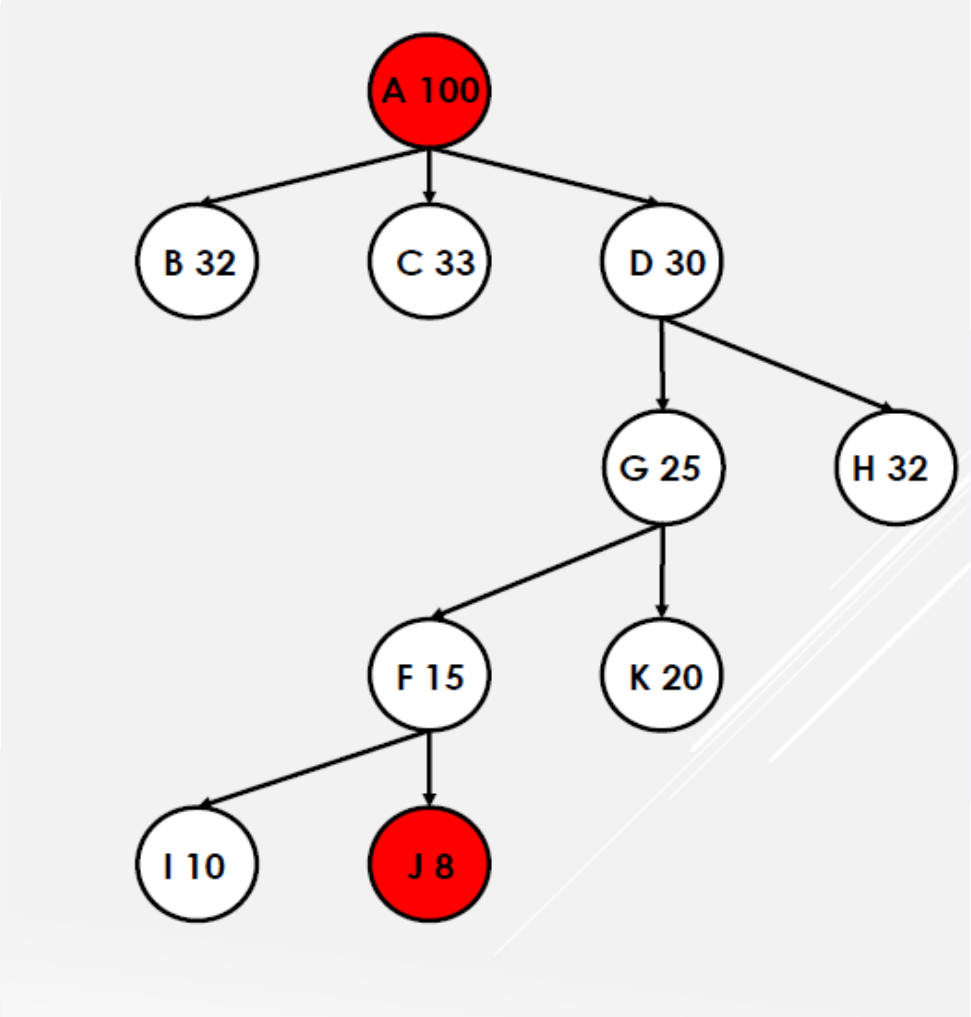
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Hill-climbing search

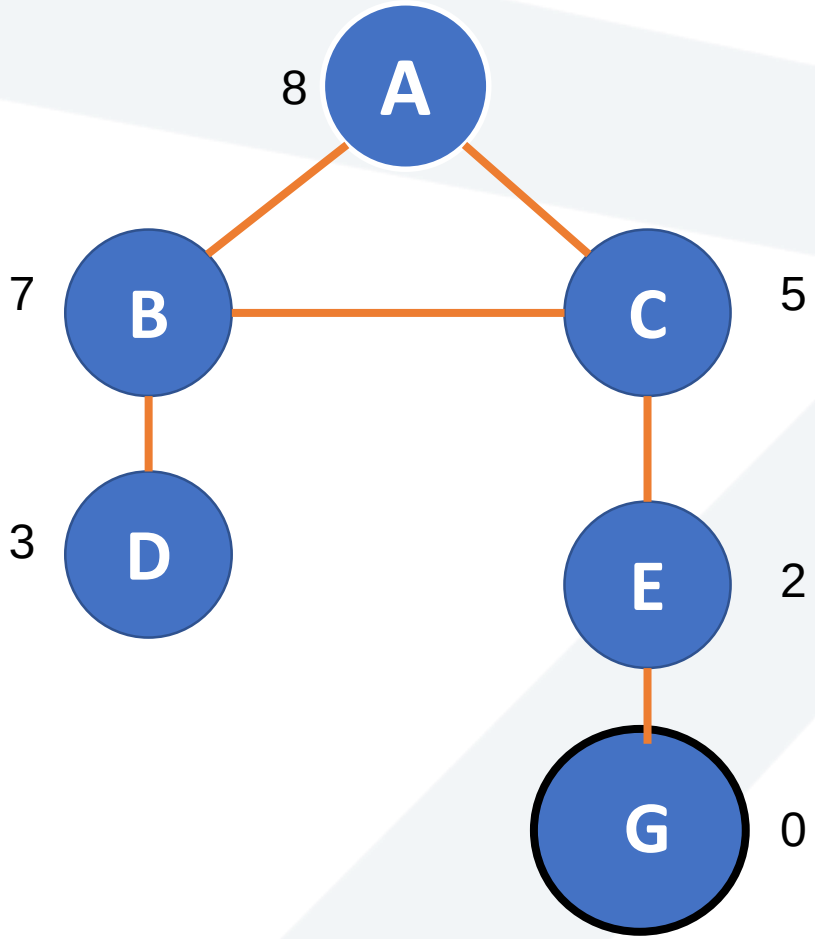
- Problem: depending on initial state, can get stuck in local maxima
-



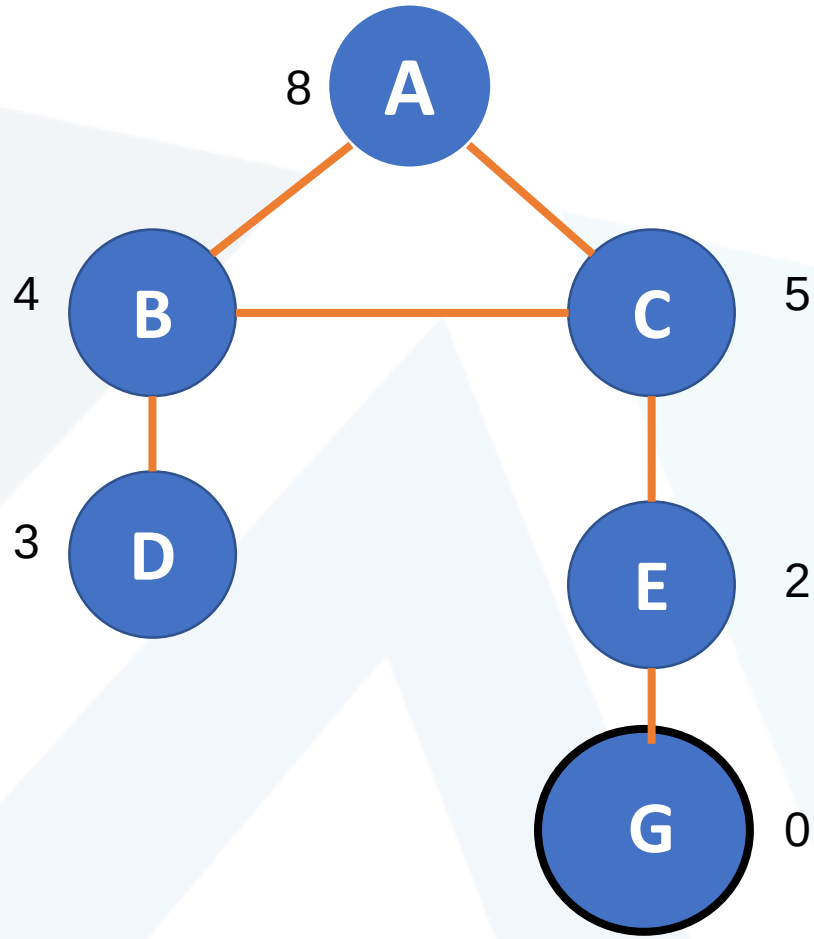
Hill climbing algorithm



Example



A C E G

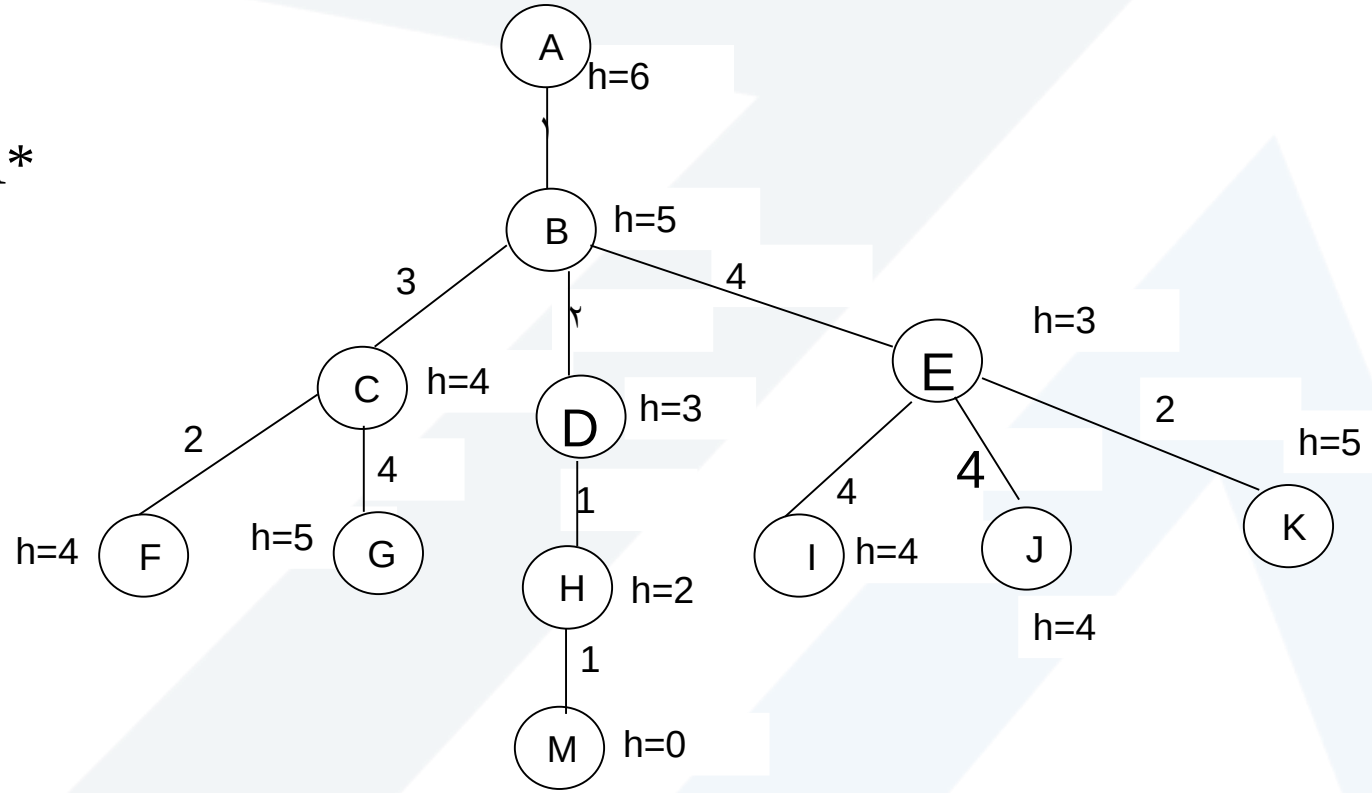


A B D

OPEN- CLOSED List Example

Using open and closed list Determine the optimum path and the best approach.

- DFS.
- BFS.
- Algorithm A*

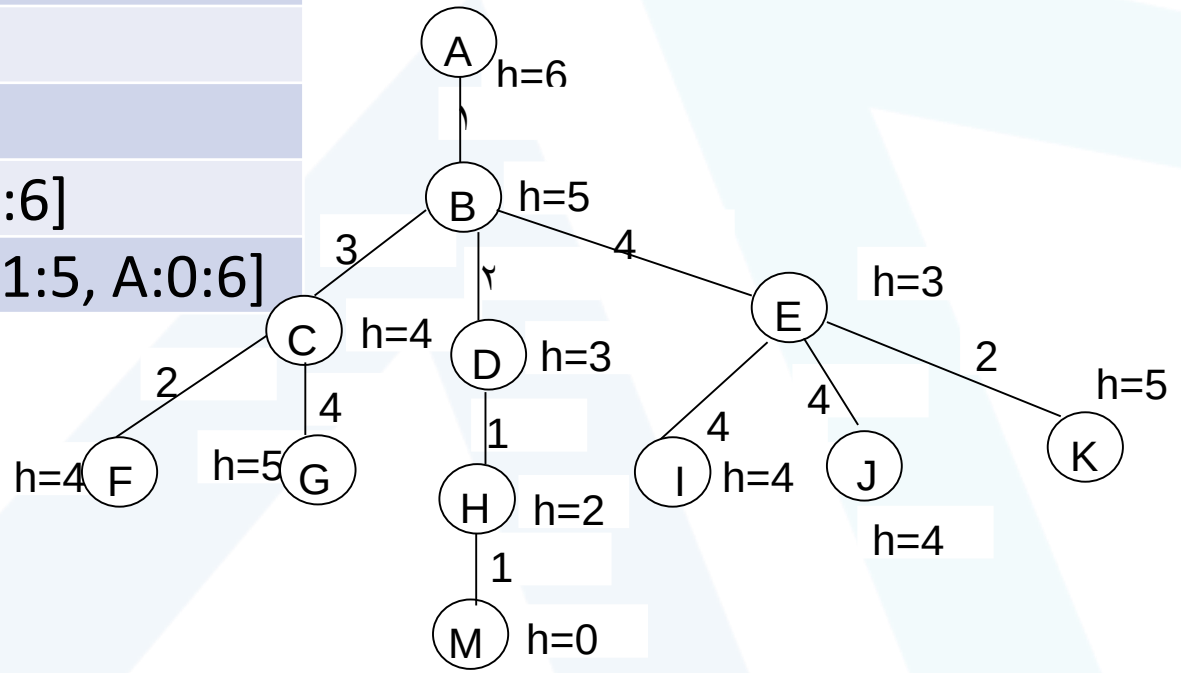


OPEN- CLOSED List Example- Solution

DFS			BFS		
تتالي	OPEN	CLOSED (VISITSD)	تتالي	OPEN	CLOSED (VISITSD)
1	[A];	[]	1	[A];	[]
2	[B];	[A]	2	[B];	[A]
3	[C, D, E];	[B, A]	3	[C, D, E];	[B, A]
4	[F,G , D, E];	[C, B, A]	4	[D, E, F,G];	[C, B, A]
5	[G , D, E];	[F, C, B, A]	5	[E, F,G,H];	[D, C, B, A]
6	[D, E];	[G, F, C, B, A]	6	[F,G,H, I, J, K];	[E, D, C, B, A]
7	[H, E];	[D,G, F, C, B, A]	7	[G,H, I, J, K];	[F, E, D, C, B, A]
8	[M, E];	[H, D,G, F, C, B, A]	8	[H, I, J, K];	[G, F, E, D, C, B, A]
9	[E];	[M, H, D,G, F, C, B, A]	9	[I, J, K, M];	[H, G, F, E, D, C, B, A]
			10	[J, K, M];	[I, H, G, F, E, D, C, B, A]
			11	[K, M];	[J, I, H, G, F, E, D, C, B, A]
			12	[];	[M, K, J, I, H, G, F, E, D, C, B, A]

OPEN- CLOSED List Example- **Solution**

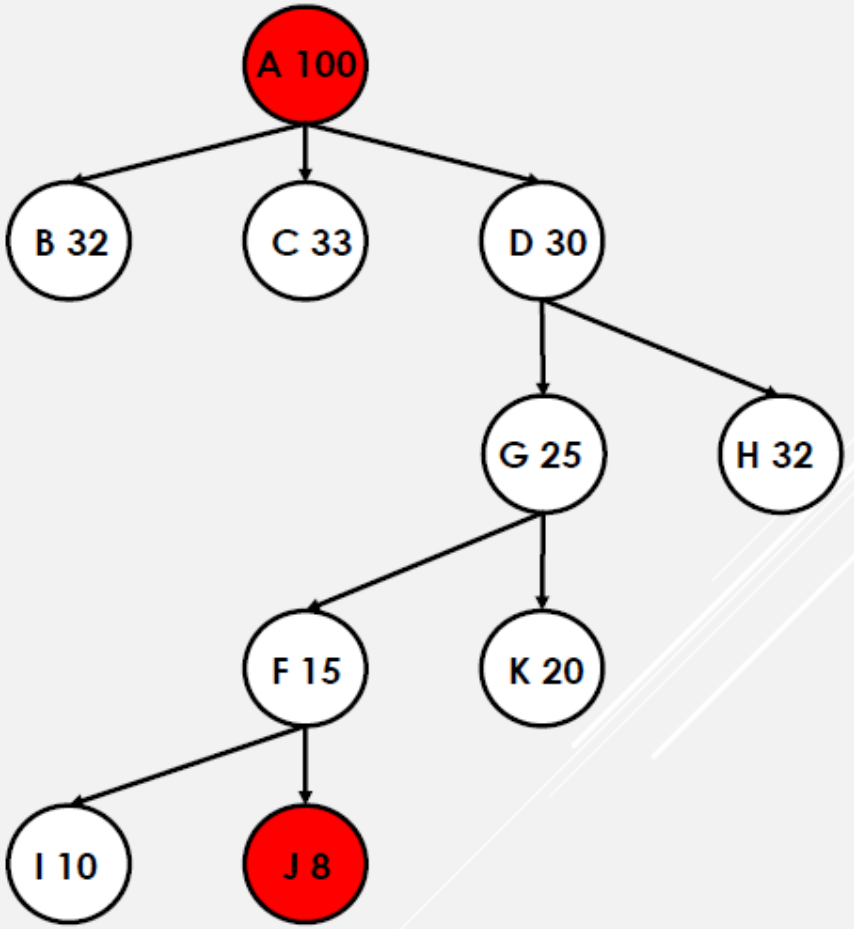
A*		
تتالي	OPEN	CLOSED (VISITSD)
0	[A:0:6]	[]
1	[B:1:5]	[A:0:6]
2	[D:3:3, C4:3, E5:3]	[B:1:5, A:0:6]
3	[H:4:2, C4:3, E5:3]	[D:3:3, B1:5, A:0:6]
4	[M:5:0, C4:3, E5:3]	[H:4:2, D3:3, B1:5, A:0:6]
5	[C4:3, E5:3]	[M:5:0, H:4:2, D3:3, B1:5, A:0:6]



Hill climbing

Open	Close
[A 100]	[ ]
[D 30, ~~B32~~, ~~C33~~]	[A 100]
[G 25, ~~H 32~~]	[A 100, D 30]
[F 15, ~~K 20~~]	[A 100, D 30, G 25]
[J 8, I 10]	[A 100, D 30, G 25, F15]
Stop	[A 100, D 30, G 25, F15, J 8]

Path is A 100, D 30, G 25, F 15, J8



example

Apply search algorithms:
A*
Greedy search