21CSC206T– Artificial Intelligence

UNIT – 2

Unit 2 List of Topics

- Searching techniques – Uninformed search – General search Algorithm
- Uninformed search Methods – Breadth First Search

- Uninformed search Methods – Depth First Search
- Uninformed search Methods – Depth limited Search

- *Uniformed search Methods- Iterative Deepening search*
- *Bi-directional search*

- *Informed search- Generate and test, Best First search*
- *Informed search-A* Algorithm*

- *AO* research*
- *Local search Algorithms-Hill Climbing, Simulated Annealing*

- *Local Beam Search*
- *Genetic Algorithms*

- *Adversarial search Methods-Game playing-Important concepts*
- *Game playing and knowledge structure.*

- *Game as a search problem-Minimax Approach*
- *Minimax Algorithm*

- *Alpha beta pruning*
- *Game theory problems*

General Search Strategies

- **Blind search** □ traversing the search space until the goal nodes is found (might be doing exhaustive search).

- *Techniques* : **Breadth First, Depth first, Depth Limited, Iterative Deepening search, Uniform Cost search**.
- Guarantees solution.

- **Heuristic search** □ search process takes place by traversing search space with applied rules (information).

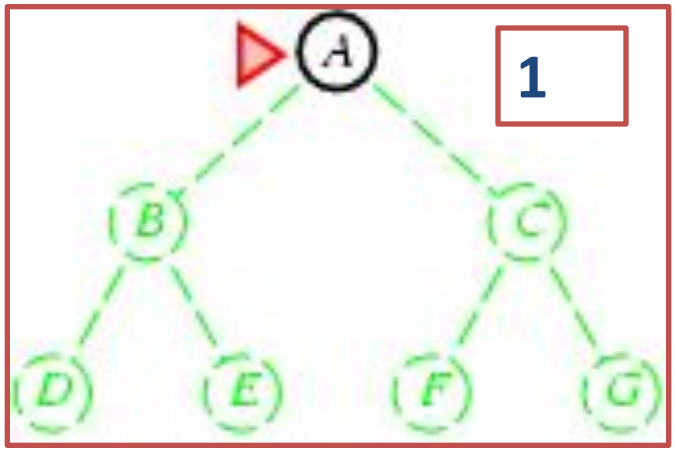
- T*echniques*: **Greedy Best First Search, A* Algorithm**

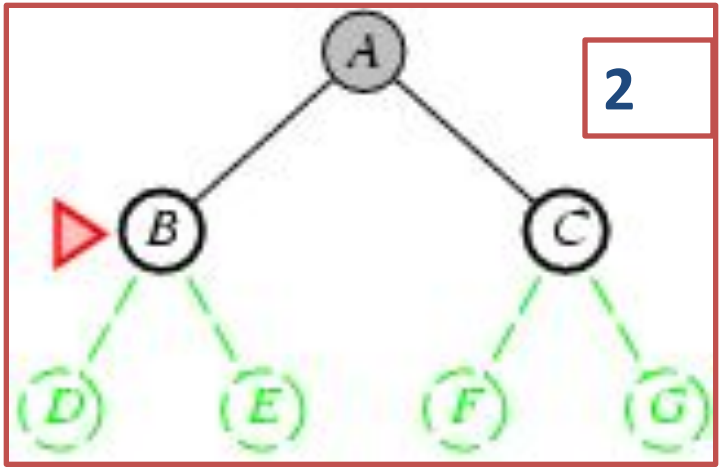
- There is no guarantee that solution is found.

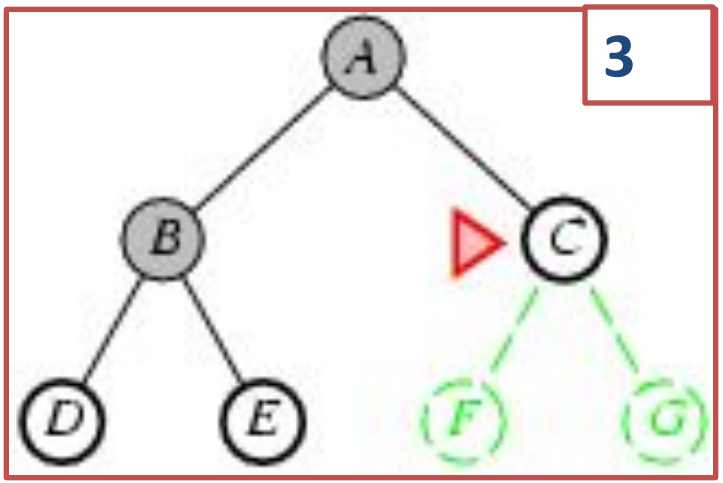
Important Terms

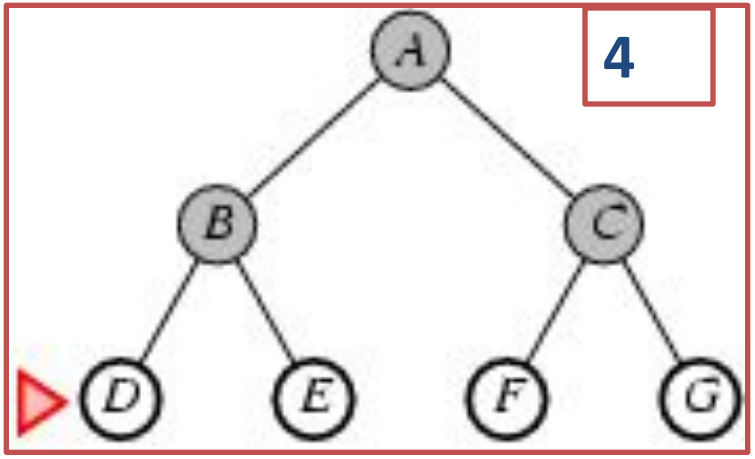
- **Search space** □ possible conditions and solutions.
- **Initial state** □ state where the searching process started.
- **Goal state** □ the ultimate aim of searching process.
- **Problem space** □ “what to solve”
- **Searching strategy** □strategy for controlling the search.
- **Search tree** □ tree representation of search space, showing possible solutions from initial state.

Blind Search : Breadth First Search









Blind Search : Breadth First Search

BFS characteristics

- Completeness: if the branching factor is 'b' and the goal node is at depth d, BFS will eventually find it.
- Optimality: BFS is optimal if path cost is a non-decreasing function of depth.[a]
- Time complexity:
  $1 + b + b^2 + b^3 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$.
- Space complexity: $O(b^{d+1})$.[b]

---

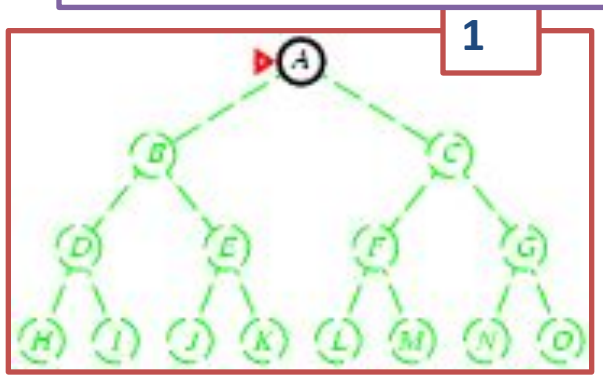
[a] Otherwise, the shallowest node may not necessarily be optimal.

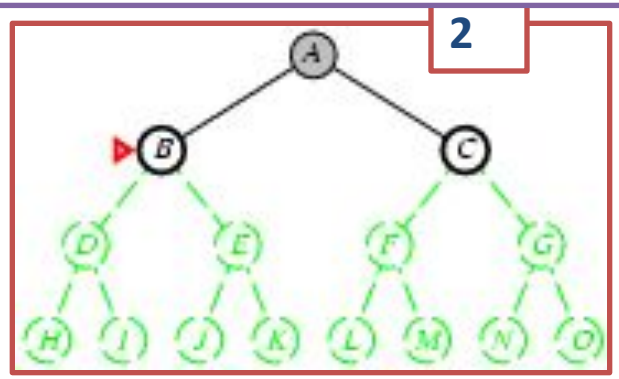
[b] b branching factor; d depth of the goal node

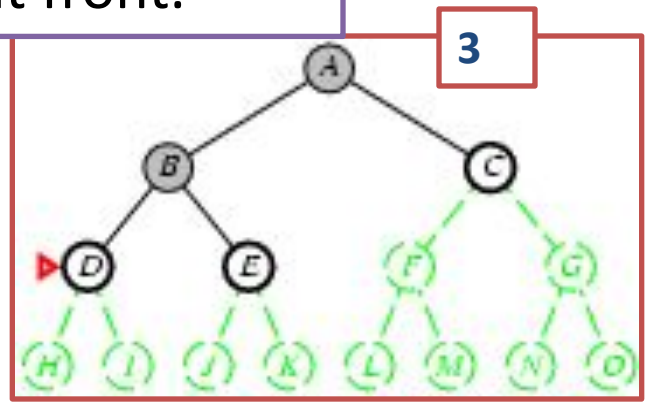
Blind Search : Depth First Search (DFS)

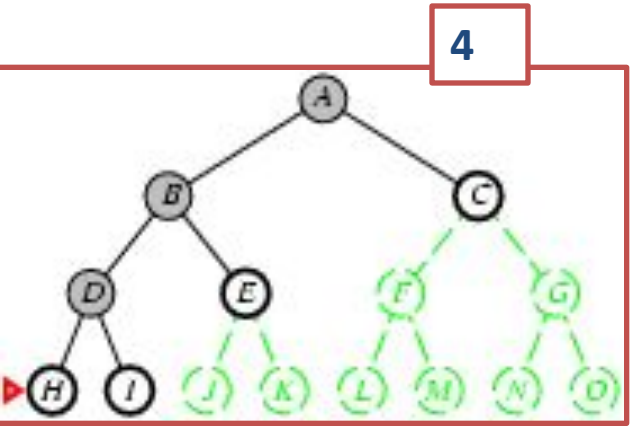
Implementation:

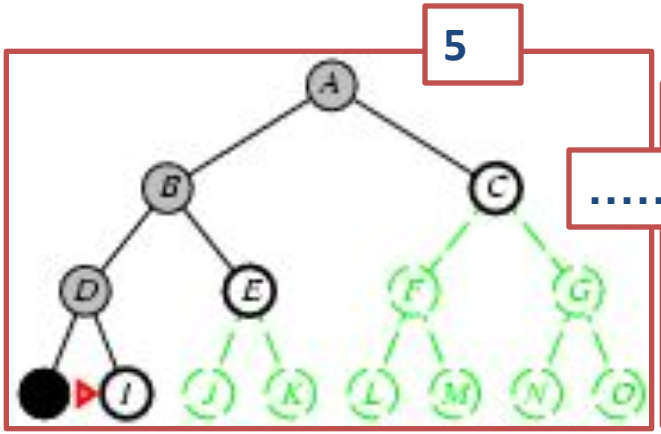
fringe = LIFO queue, i.e., put successors at front.



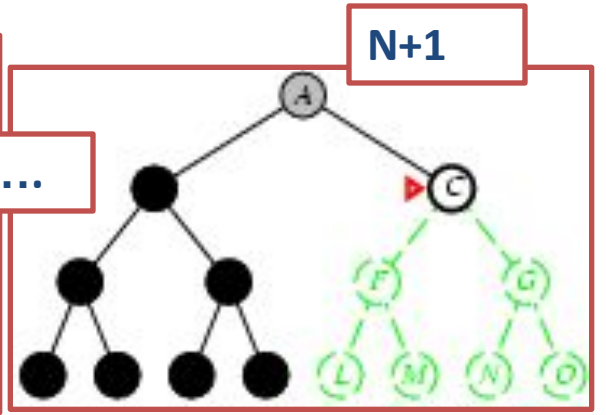








.....



Blind Search : Depth First Search (DFS)

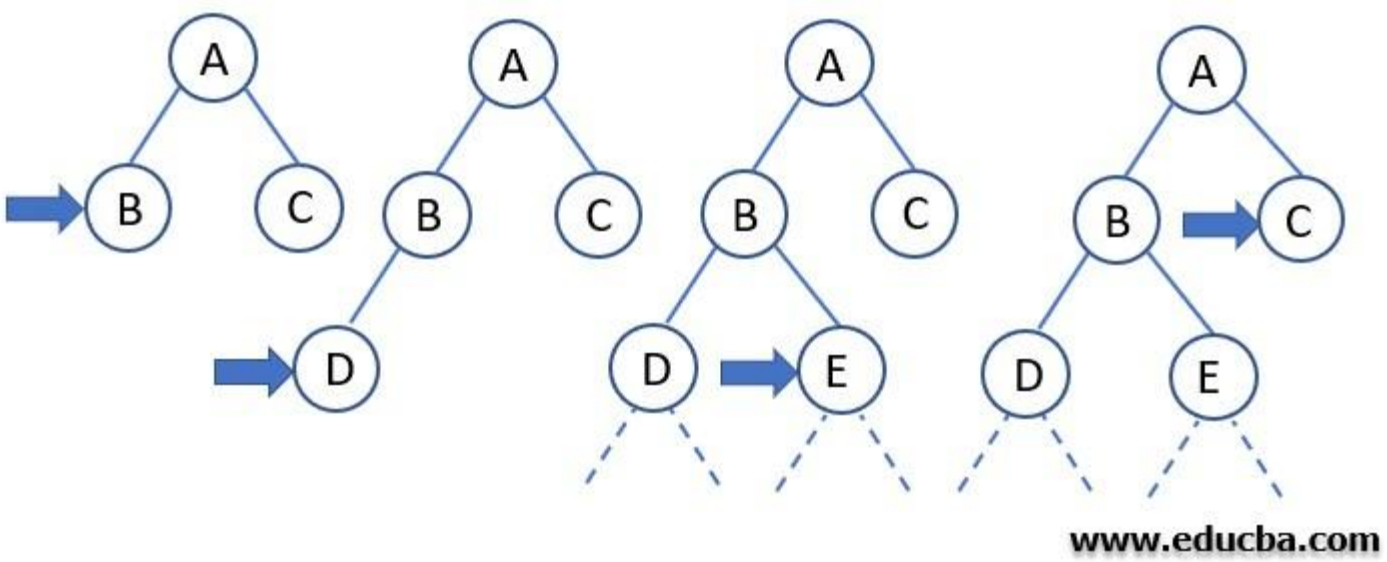
DFS characteristics

- Small space requirements: only the path to the current node and the siblings of each node in the path are stored.
- Backtracking search generates only one successor for each node.
- Completeness: no, if the expanded subtree has an infinite depth.
- Optimality: no, if a solution located deeper, but located in a subtree expanded earlier, is found.
- Time complexity: $O(b^m)$.

  Space complexity: $O(bm)$ (linear!).

Blind Search : Depth Limited Search (DLS)

In Depth Limited Search, we first set a constraint on how deep (or how far from root) will we go.



In the above example , If we fix the depth limit to 2, DLS can be carried out similarly to the DFS until the goal node is found to exist in the search domain of the tree.

Unit 2 List of Topics

- Searching techniques – Uninformed search – General search Algorithm
- Uninformed search Methods – Breadth First Search

- Uninformed search Methods – Depth First Search
- Uninformed search Methods – Depth limited Search

- *Uniformed search Methods- Iterative Deepening search*
- *Bi-directional search*

- *Informed search- Generate and test, Best First search*
- *Informed search-A* Algorithm*

- *AO* research*
- *Local search Algorithms-Hill Climbing, Simulated Annealing*

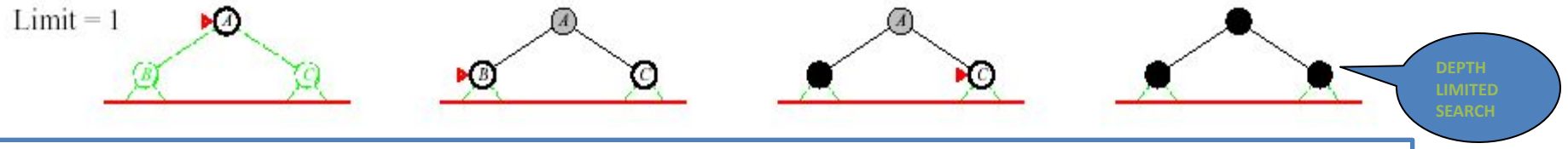
- *Local Beam Search*
- *Genetic Algorithms*

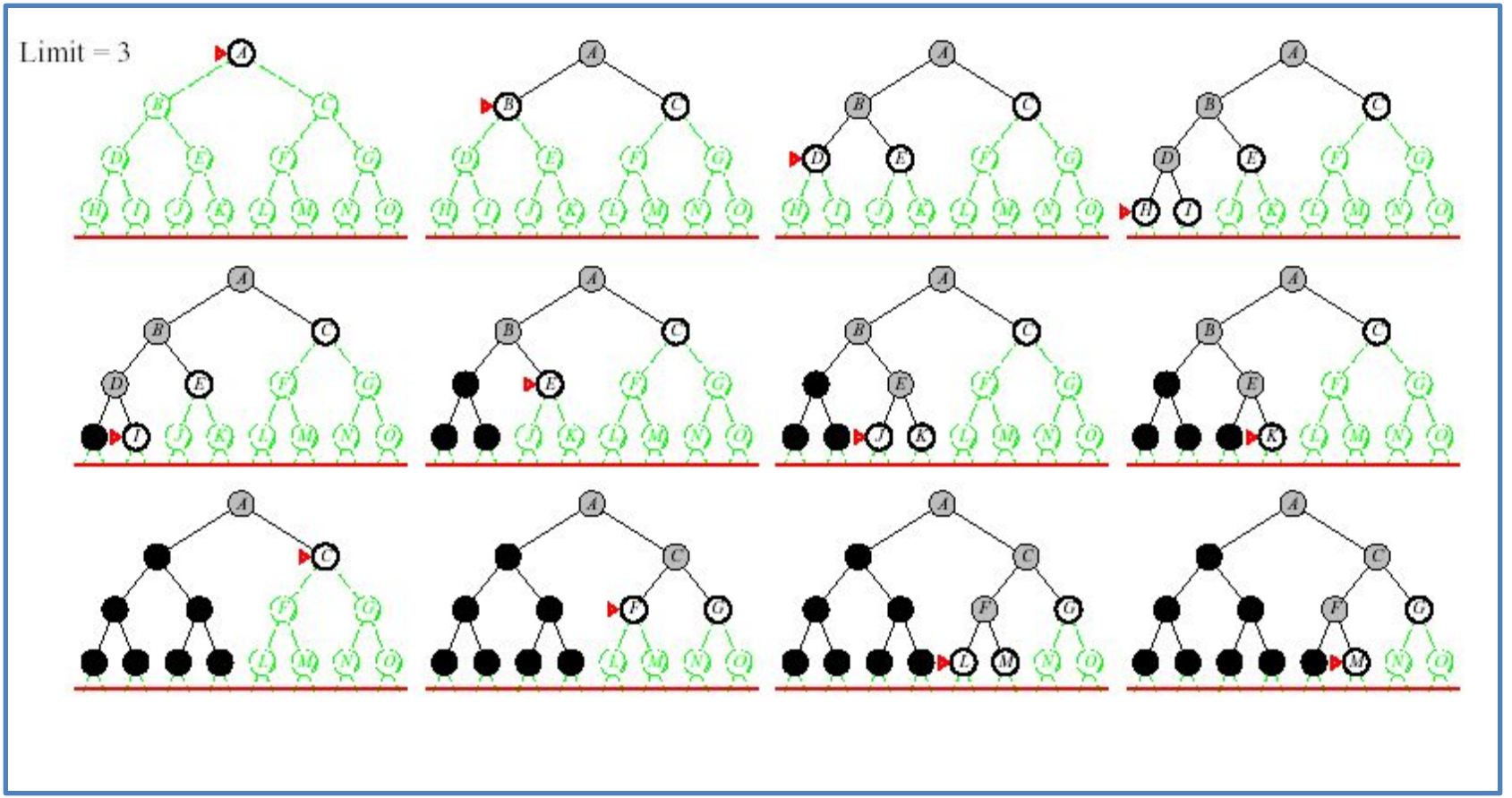
- *Adversarial search Methods-Game playing-Important concepts*
- *Game playing and knowledge structure.*

- *Game as a search problem-Minimax Approach*
- *Minimax Algorithm*

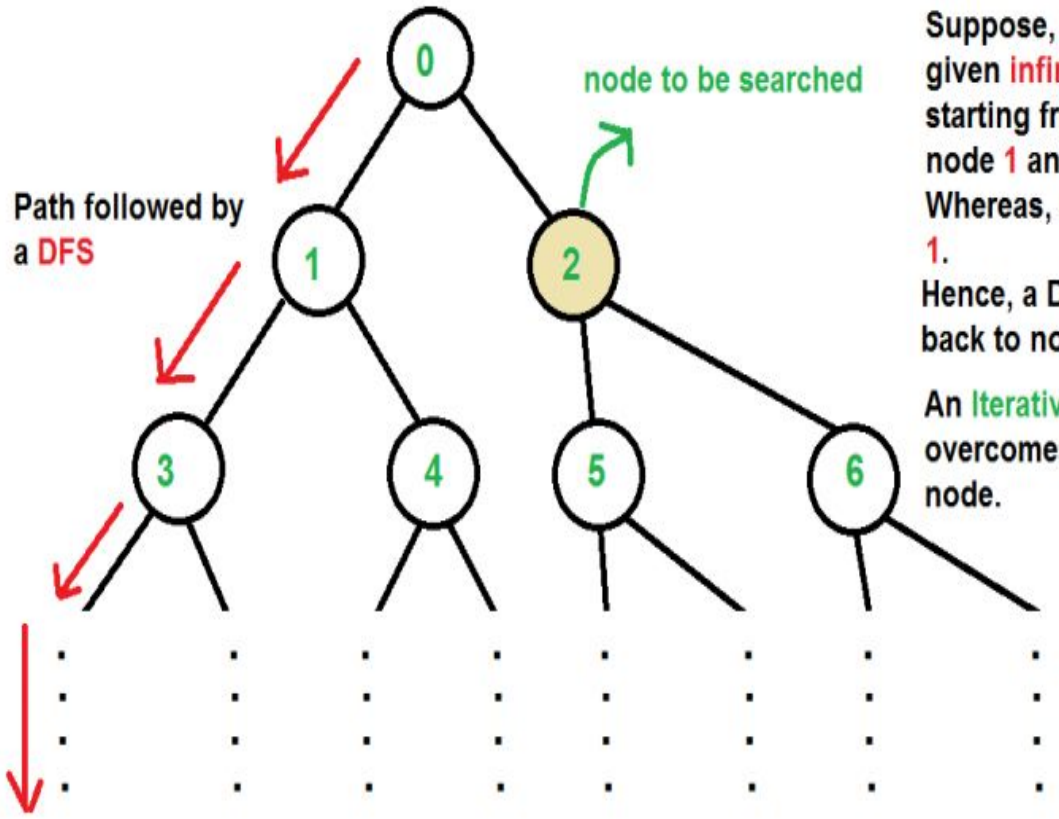
- *Alpha beta pruning*
- *Game theory problems*

Blind Search : Iterative Deepening DFS (ID-DFS)





Blind Search : Iterative Deepening DFS (ID-DFS)



Suppose, we want to find node- '2' of the given infinite undirected graph/tree. A DFS starting from node- 0 will dive left, towards node 1 and so on.
Whereas, the node 2 is just adjacent to node 1.
Hence, a DFS wastes a lot of time in coming back to node 2.

An Iterative Deepening Depth First Search overcomes this and quickly find the required node.

Blind Search : Iterative Deepening DFS (ID-DFS)

IDS characteristics

- Completeness: yes.
- Optimality: yes, if step cost = 1.
- Time complexity:

  $(d + 1)b^0 + db^1 + (d - 1)b^2 + \ldots + b^d = O(b^d)$.

- Space complexity: O(bd).

Numerical comparison for b = 10, d = 5

$$N(IDS) = 50 + 400 + 3000 + 20000 + 100000 = 123450$$

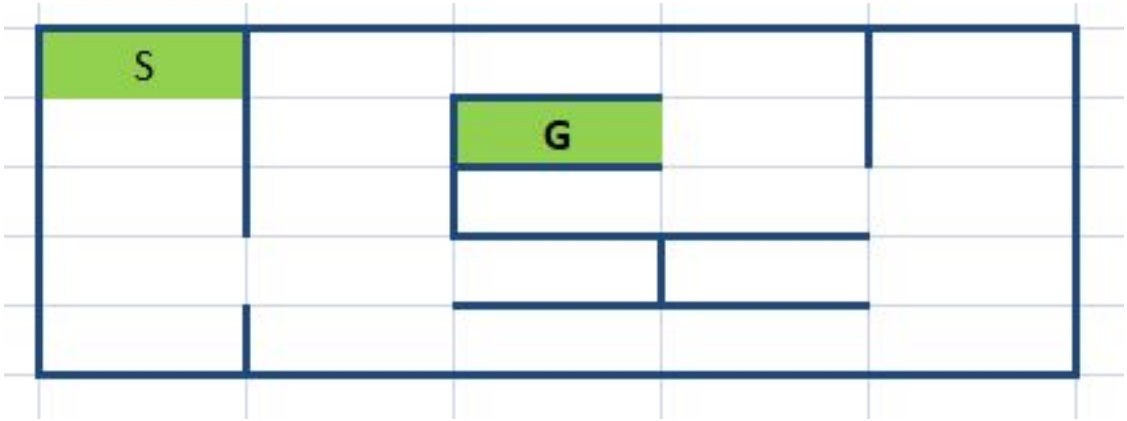
$$N(BFS) = 10 + 100 + 1000 + 10000 + 100000 + 999990 = 1111100$$

Conclusion

IDS exhibits better performance, because it does not expand other nodes at depth d.

APPLICATION: MAZE GAME

- HOW TO REACH TO THE GOAL?



APPLICATION: MAZE GAME

- BFS SOLUTION?
  - S-1-2-3-5-8-10-12-14-16-19-G
  - SEARCH SOLUTION FOUND IN **12 STEPS**

- DFS SOLUTION?
  - S-1-2-3-6-5-8-9-10-11-13-16-18-G
  - SEARCH SOLUTION FOUND IN **14 STEPS**

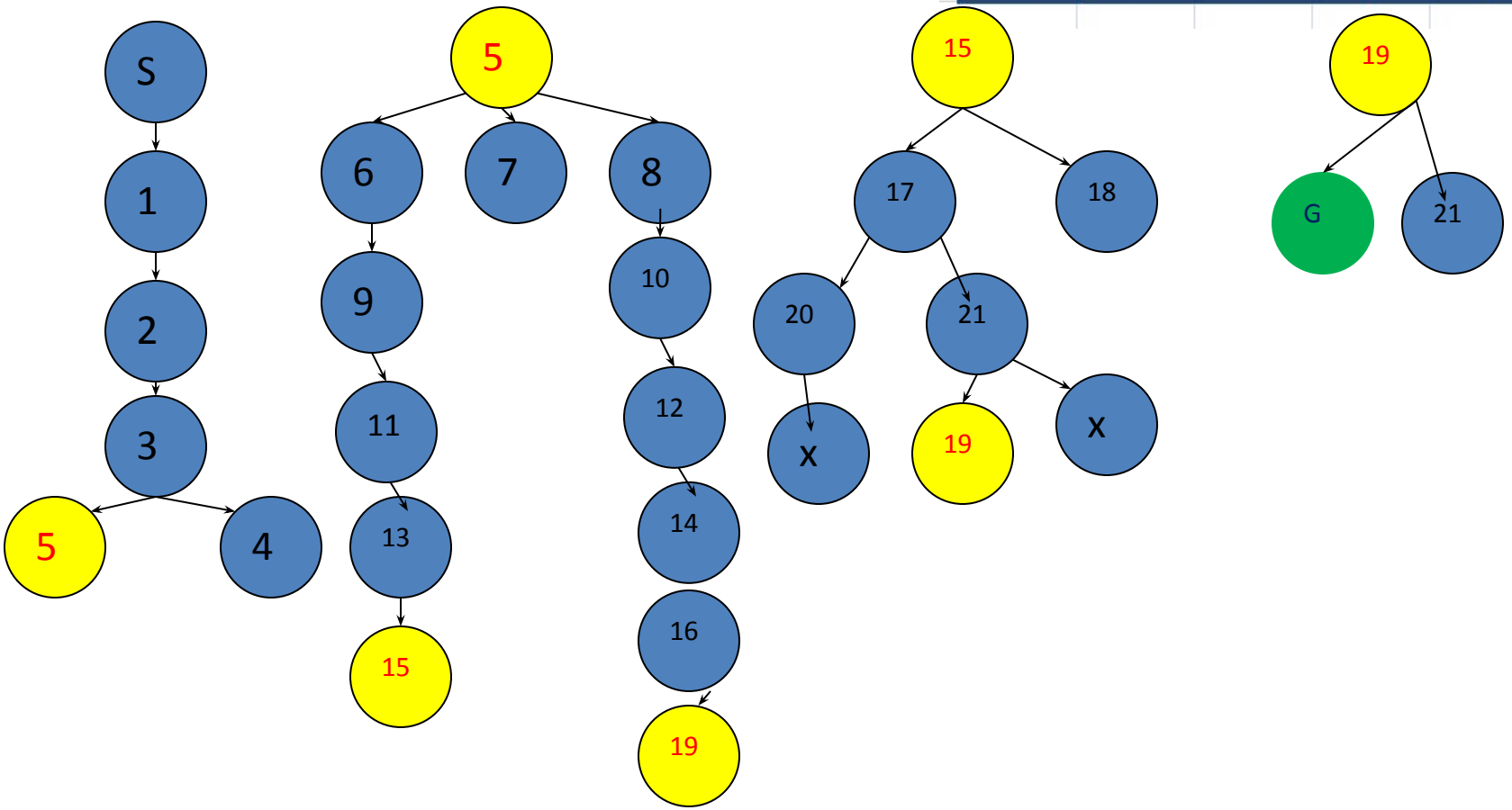
S	12	14	16	X
1	10	G	19	20
2	8	X	21	17
3	5	7	18	15
4	6	9	11	13

S	21	20	19	15
1	22	G	18	14
2	23	17	16	13
3	6	7	12	11
4	5	8	9	10

APPLICATION: MAZE GAME

- BFS SOLUTION?
  - S-1-2-3-5-8-10-12-14-16-19-G
  - SEARCH SOLUTION FOUND IN **12 STEPS**

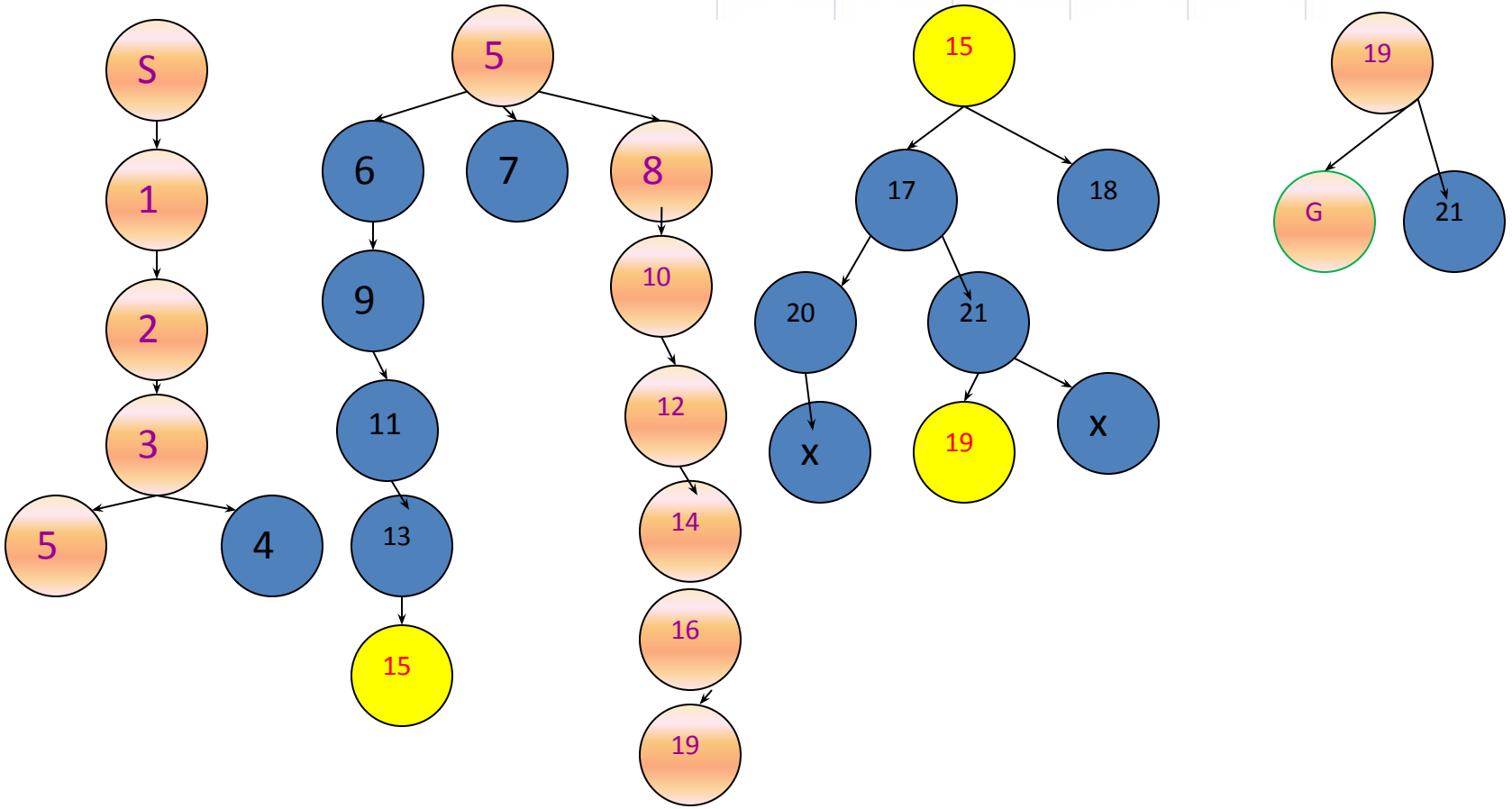
S	12	14	16	X
1	10	G	19	20
2	8	X	21	17
3	5	7	18	15
4	6	9	11	13



APPLICATION: MAZE GAME

- BFS SOLUTION?
  - S-1-2-3-5-8-10-12-14-16-19-G
  - SEARCH SOLUTION FOUND IN **12 STEPS**

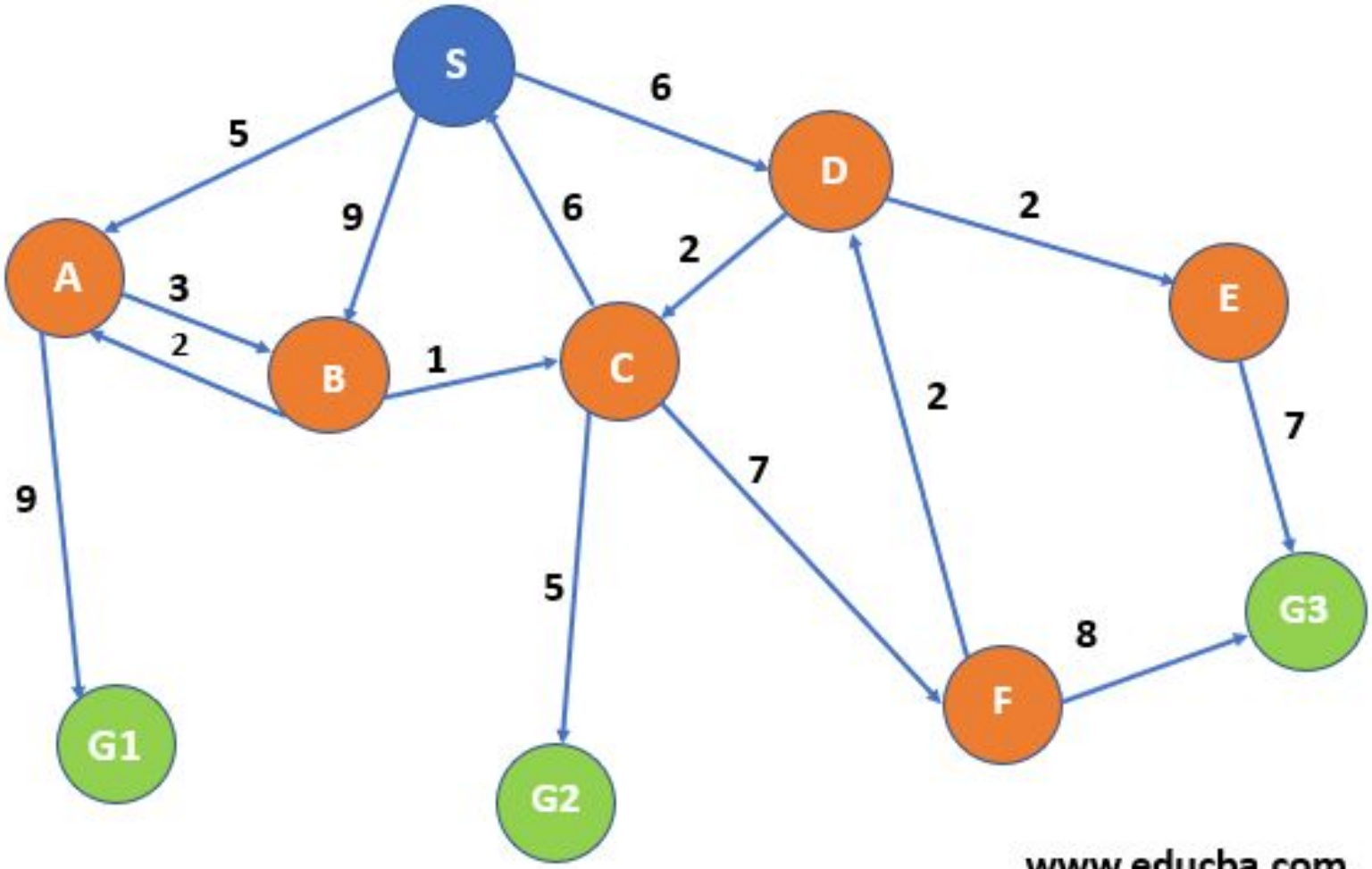
S	12	14	16	X
1	10	G	19	20
2	8	X	21	17
3	5	7	18	15
4	6	9	11	13



Blind Search : Uniform Cost Search

- This algorithm comes into play when a different cost is available for each edge.
- The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost.
- Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand.
- A uniform-cost search algorithm is implemented by the priority queue.
- It gives maximum priority to the lowest cumulative cost.
- Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

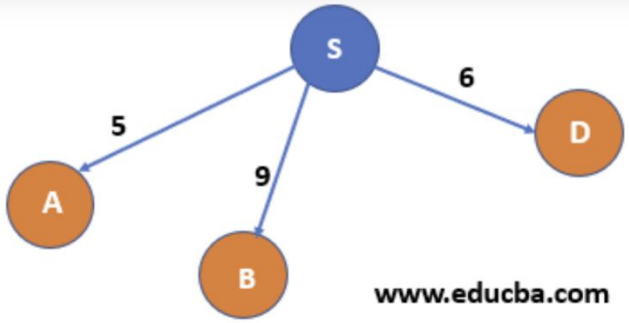
Uniform Cost Search - Example



Uniform Cost Search - Example

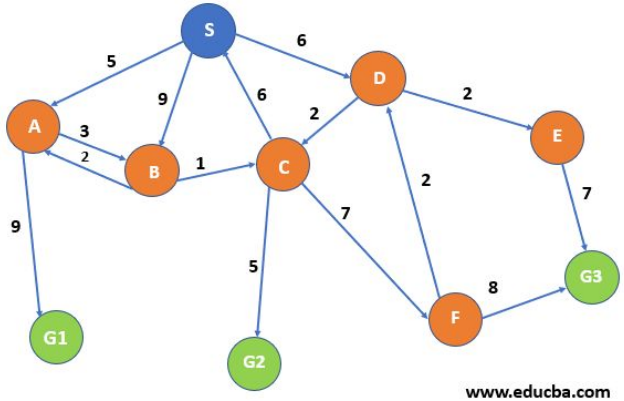
Explanation	Flow	Visited List
Step 1- We will start with start node and check if we have reached any of the destination nodes i.e No thus continue.		

Uniform Cost Search - Example

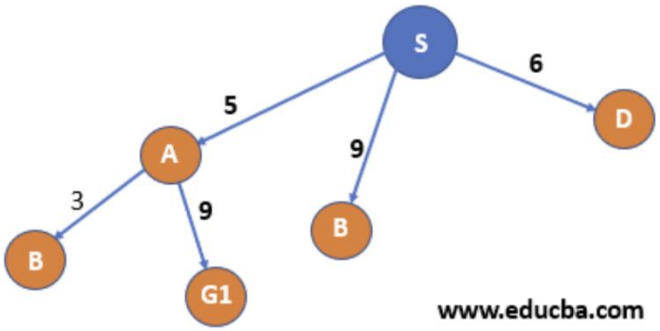


Step 2– We reach all the nodes that can be reached from S I .eA, B, D. And Since node S has been visited thus added to the visited List. Now we select the cheapest path first for further expansion i.e. A

S



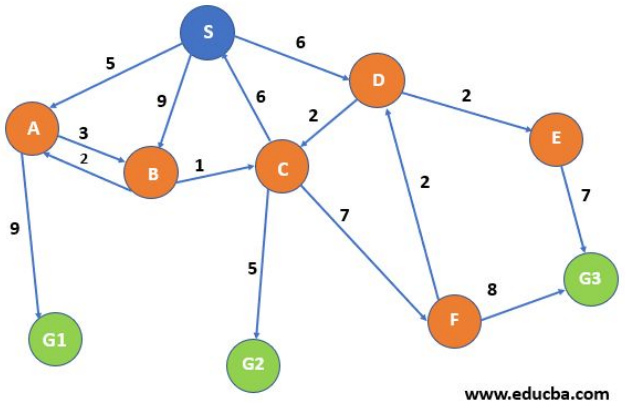
Uniform Cost Search - Example



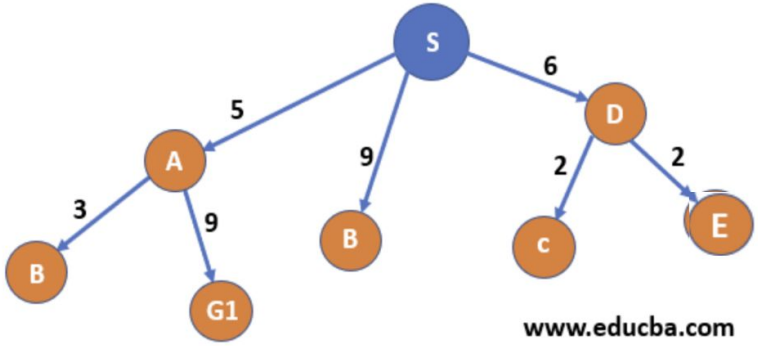
Step 3 – Node B and G1 can be reached from A and since node A is visited thus move to the visited list.

S, A

Since G1 is reached but for the optimal solution we need to consider every possible case thus we will expand the next cheapest path i.e. S->D.

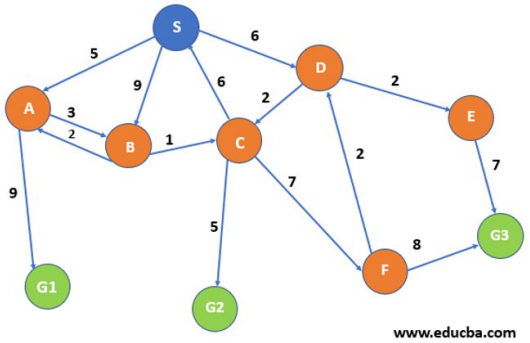


Uniform Cost Search - Example

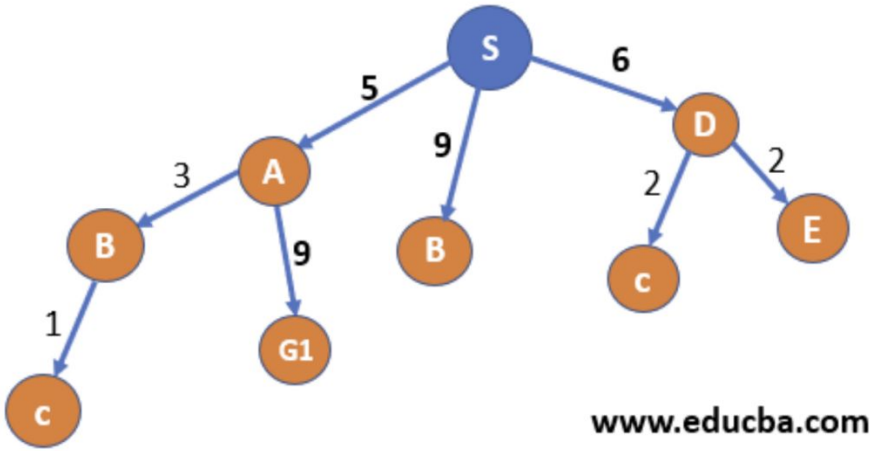


Step 4– Now node D has been visited thus it goes to visited list and now since we have three paths with the same cost, we will choosealphabetically thus will expand node B

S,A,D

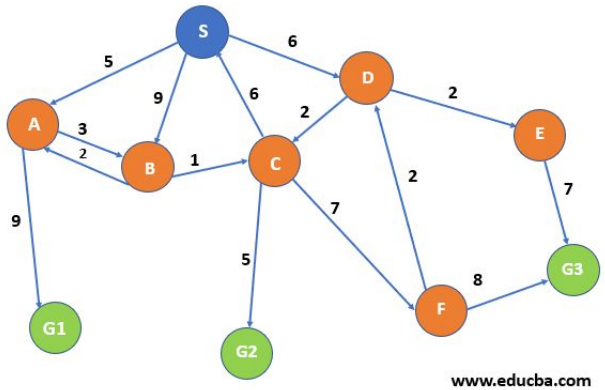


Uniform Cost Search - Example

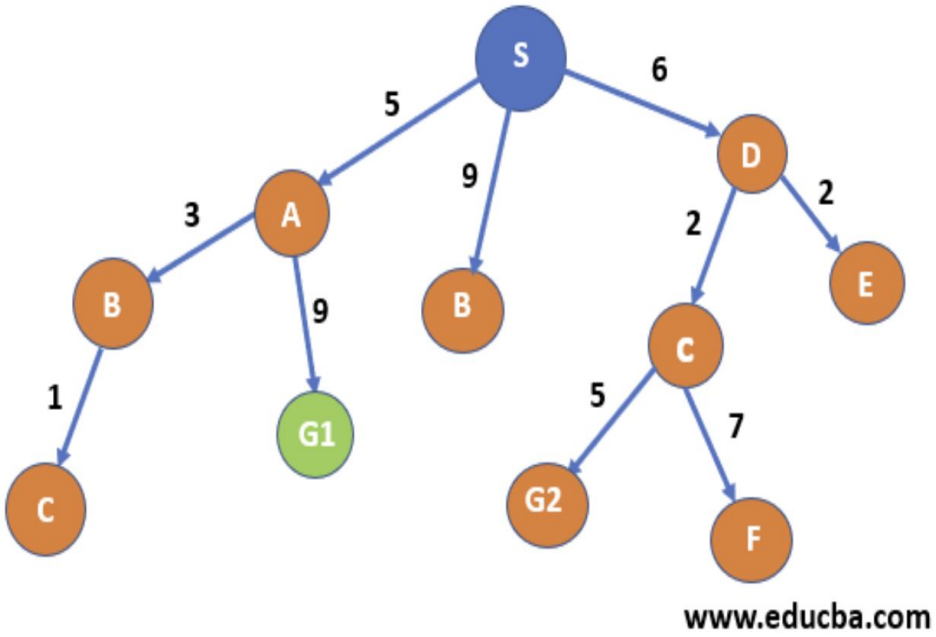


Step 5-: From B we can only reach node C. Now the path with minimum weight is S->D->C i.e 8. Thus expand C. And B is now visited node.

S,A,D,B



Uniform Cost Search - Example



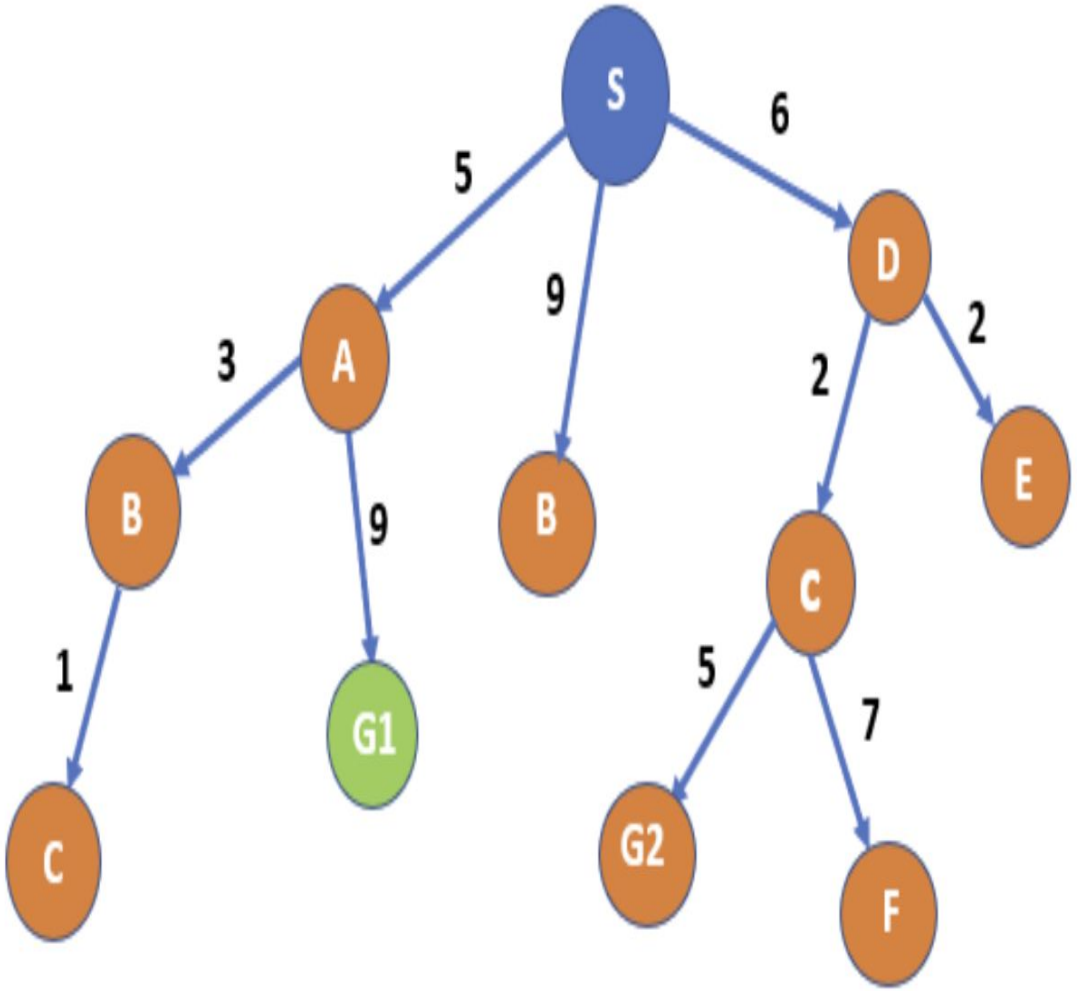
Step 6:- From C we can reach G2 and F node with 5 and 7 weights respectively.

S,A,D,B,C

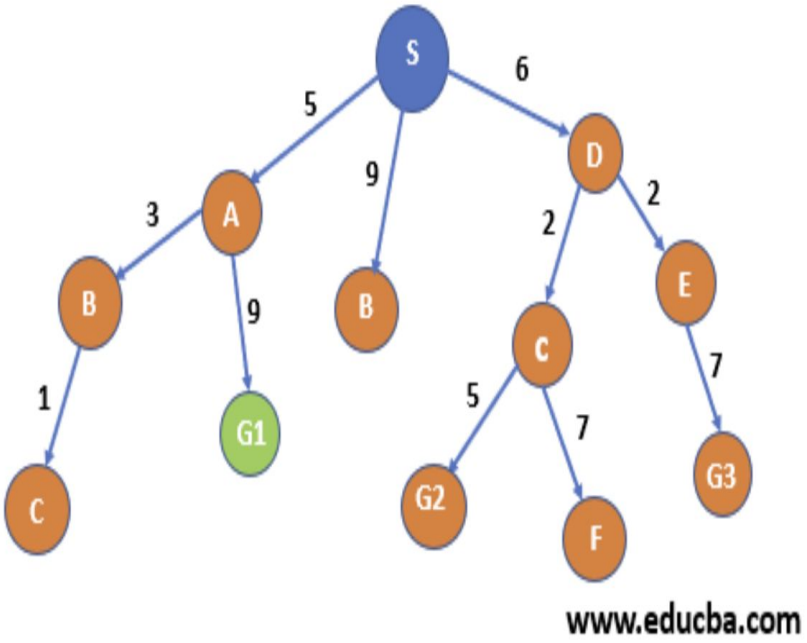
Uniform Cost Search - Example

Since S is present in the visited list thus we are not considering the C->S path.

Now C will enter the visited list. Now the next node with the minimum total path is S->D->E i.e 8.Thus we will expand E.



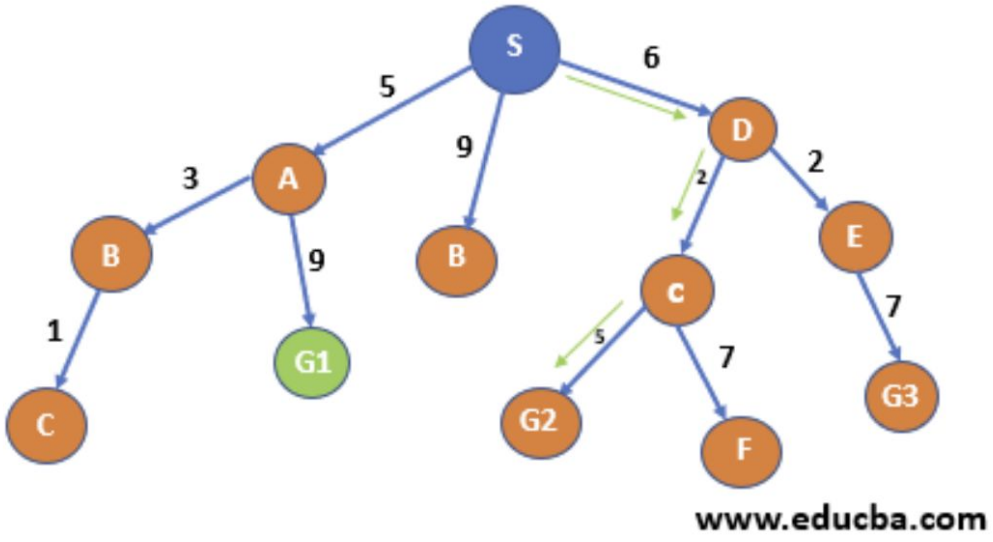
Uniform Cost Search - Example



Step 7:- From E we can reach only G3. E will move to the visited list.

S,A,D,B,C,E

Uniform Cost Search - Example



S,A,D,B,C,E

Minimum is S->D->C->G2
And also G2 is one of the destination nodes thus we found our path.

In this way we can find the path with the minimum cumulative cost from a start node to ending node – S->D->C->G2 with cost total cost as 13(marked with green color).

Uniform Cost Search

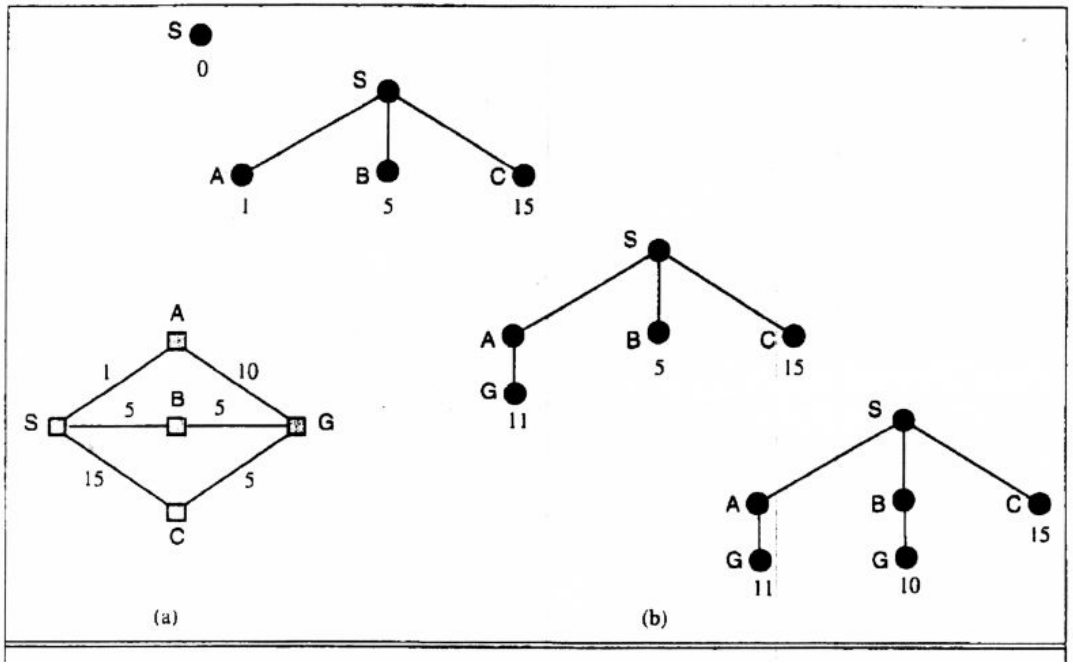
Implementation: *fringe* = queue ordered by path cost

Equivalent to breadth-first if all step costs all equal.

Breadth-first is only optimal if step costs is increasing with depth.

(e.g. constant). Can we guarantee optimality for any step cost?

Uniform-cost Search: Expand node with smallest path cost g(n).



Uniform Cost Search

Uniform-cost search strategy

- Expands the node with the lowest cost from the start node rst.[a]
- Completeness: yes, if cost of each step $\geq s > 0$.
- Optimality: same as above nodes are expanding in increasing order of cost.
- Time and space complexity: $O(b^{1+\lfloor C^*/s \rfloor})$.[b]

---

[a] If all step costs are equal, UCS=BFS. C

[b] * cost of optimal solution

Uniform Cost Search

Time Complexity:

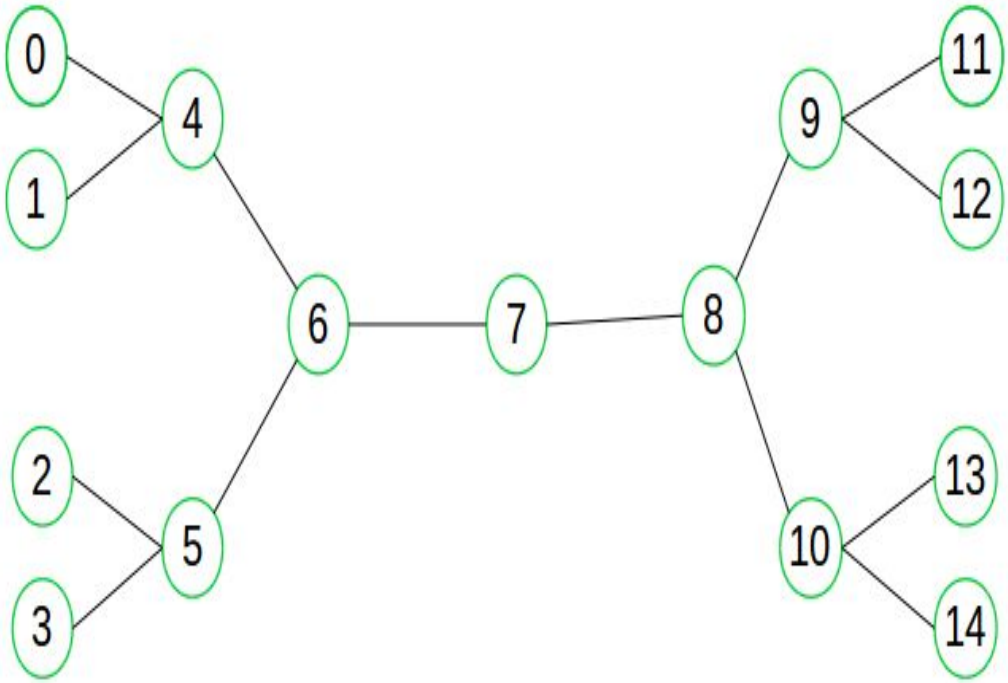
Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

Hence, the worst-case time complexity of Uniform-cost search is$O(b^{1 + [C^*/\varepsilon]})$/.

Space Complexity:

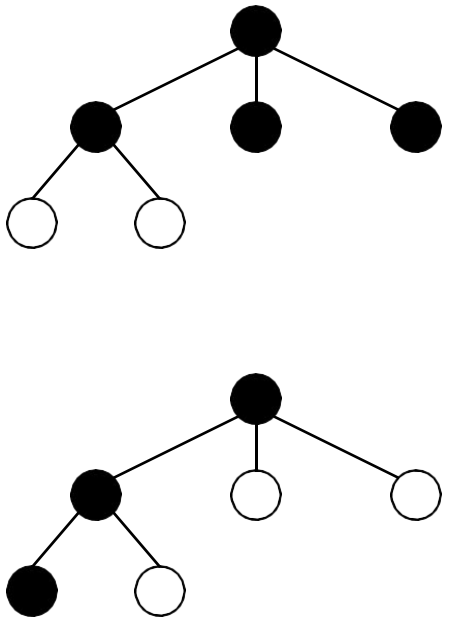
The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is $O(b^{1 + [C^*/\varepsilon]})$.

Blind search -Bidirectional Search



Summary of Blind Search Algorithms

Criterion	Breadth-First	Depth-First
Time	b^d	b^m
Space	b^d	bm
Optimal?	Yes	No
Complete?	Yes	No



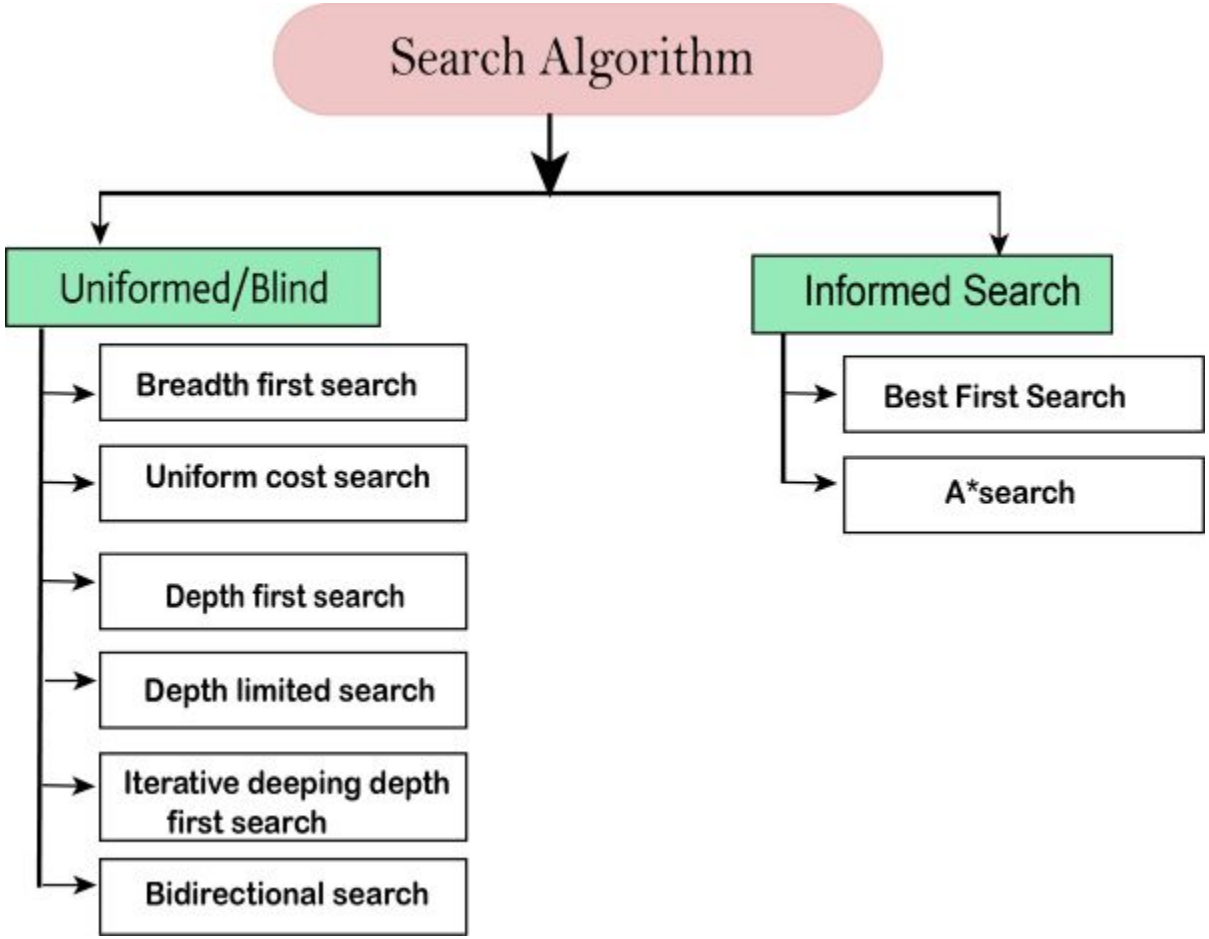
b: branching factor d: solution depth m: maximum depth

Summary of Blind Search Algorithms

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

Algorithm	Space	Time	Complete	Optimal
BFS	Theta (b^d)	Theta(b^d)	Yes	Yes
DFS	Theta(d)	Theta(b^m)	No	No
UCS	Theta(b^ (ceil(C*/e))	Theta(b^d)	Yes	Yes
DLS	Theta(l)	Theta(b^l)	No	No
IDS	Theta(d)	Theta(b^d)	Yes	Yes

Summary of Search Algorithms



Unit 2 List of Topics

- Searching techniques – Uninformed search – General search Algorithm
- Uninformed search Methods – Breadth First Search
- Uninformed search Methods – Depth First Search
- Uninformed search Methods – Depth limited Search
- *Uniformed search Methods- Iterative Deepening search*
- *Bi-directional search*
- *Informed search- Generate and test, Best First search*
- *Informed search-A* Algorithm*
- *AO* search*
- *Local search Algorithms-Hill Climbing, Simulated Annealing*
- *Local Beam Search*
- *Genetic Algorithms*
- *Adversarial search Methods-Game playing-Important concepts*
- *Game playing and knowledge structure.*
- *Game as a search problem-Minimax Approach*
- *Minimax Algorithm*
- *Alpha beta pruning*
- *Game theory problems*

Informed Search Algorithms

- Generate and Test
- Best-first search
- Greedy best-first search
- A^* search
- Heuristics

Generate-and-test

- Very simple strategy - just keep guessing.

```
do while goal not accomplished
   generate a possible solution
   test solution to see if it is a goal
```

- Heuristics may be used to determine the specific rules for solution generation.

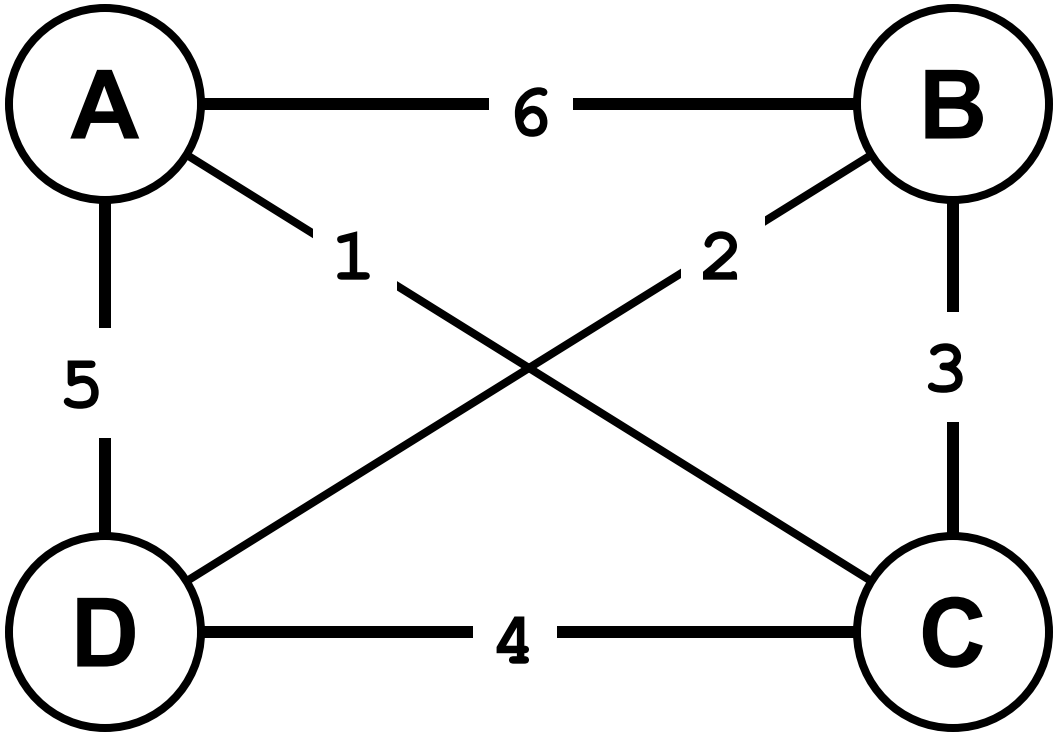
Generate-and-test

Example - Traveling Salesman Problem (TSP)

- Traveler needs to visit *n* cities.
- Know the distance between each pair of cities.
- Want to know the shortest route that visits all the cities once.
- *n=80* will take millions of years to solve exhaustively!

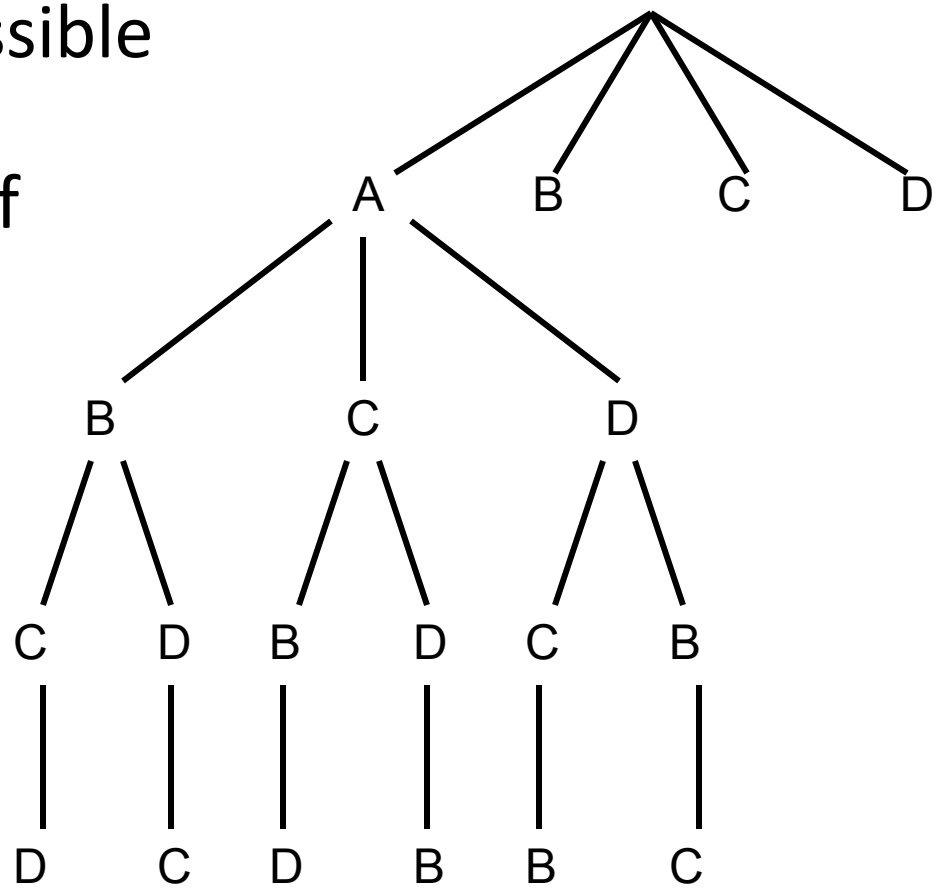
Generate-and-test

TSP Example



Generate-and-test Example

- TSP - generation of possible solutions is done in lexicographical order of cities:
  1. A - B - C - D
  2. A - B - D - C
  3. A - C - B - D
  4. A - C - D - B
  ...



Best First Search Algorithms

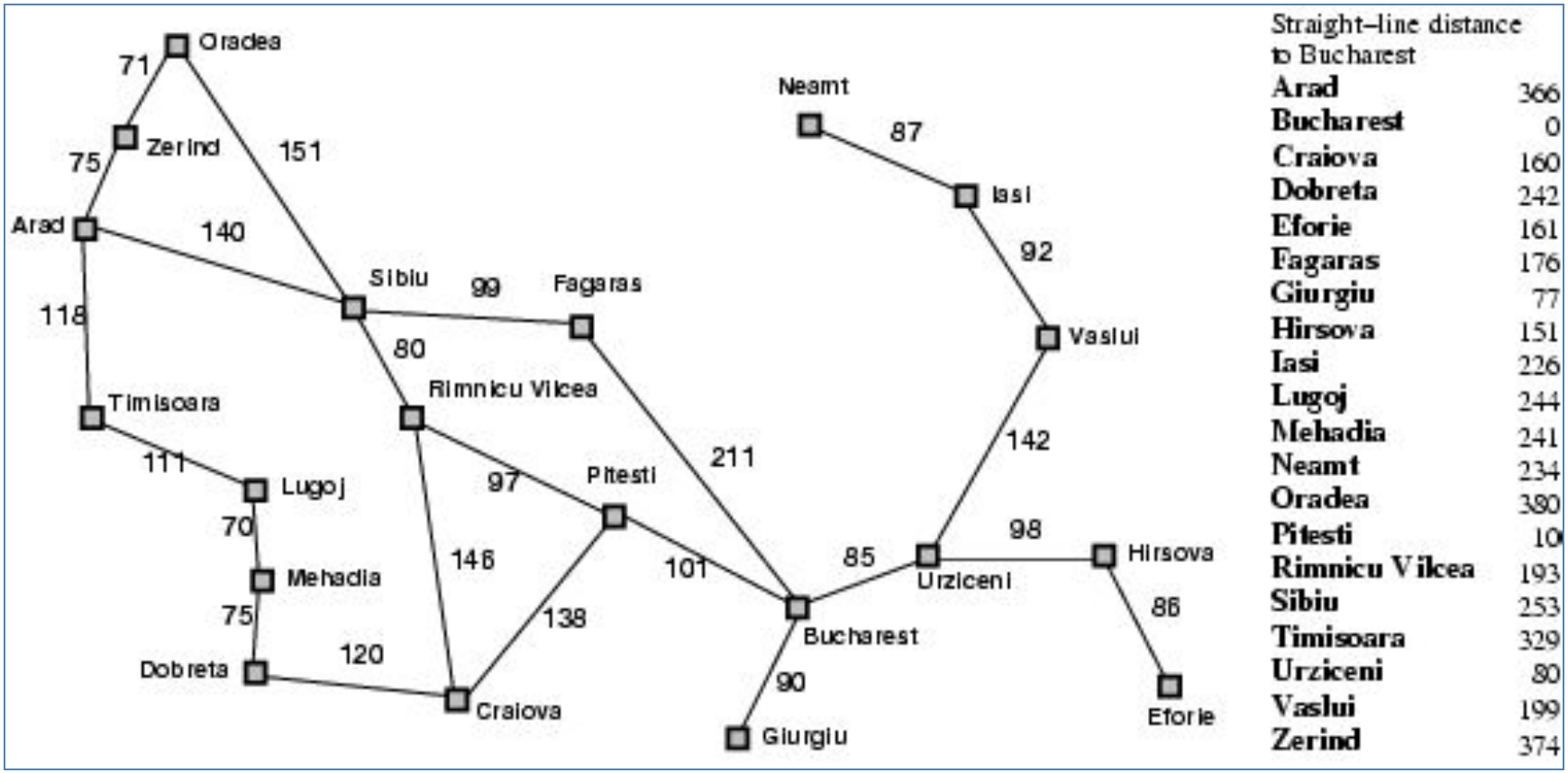
- Idea: use an evaluation function *f(n)* for each node
  - f(n) provides an estimate for the total cost.
  - Expand the node n with smallest f(n).

- Implementation:
  Order the nodes in fringe increasing order of cost.

- Special cases:
  - greedy best-first search
  - A* search

- UCS = g(n) : Min g(n)
- BFS (Best First Search) = h(n), Optimal
- A* = g(n)+h(n) = f(n), Optimal

Romania with straight-line dist.

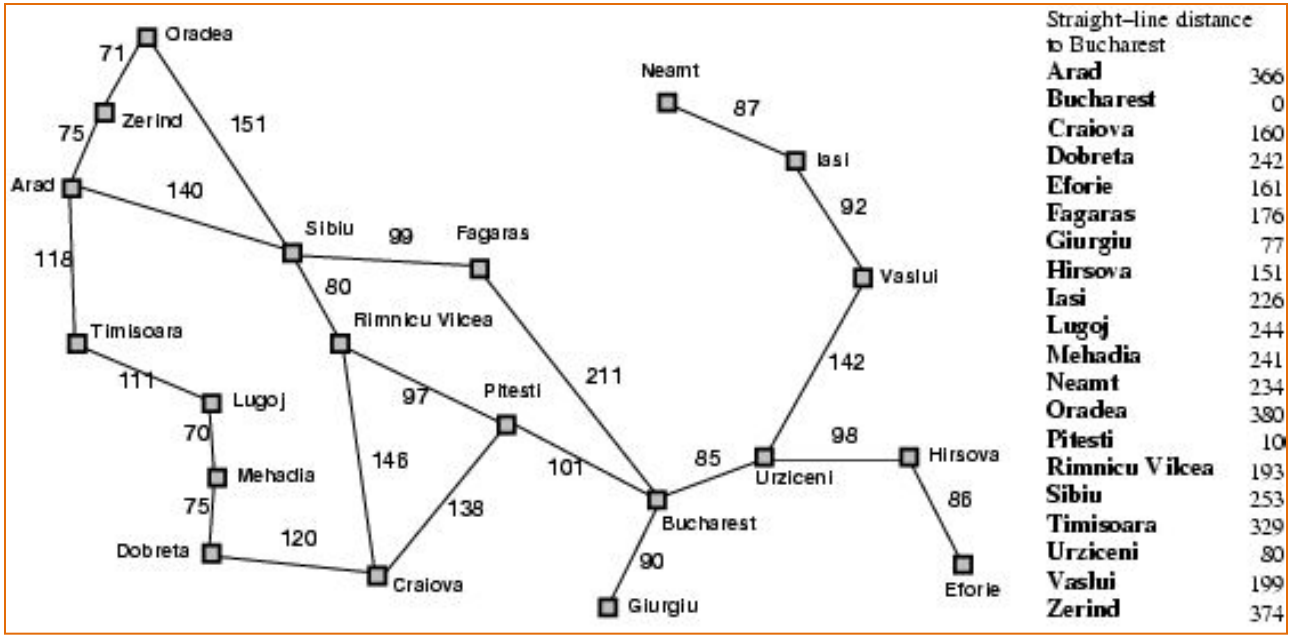


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

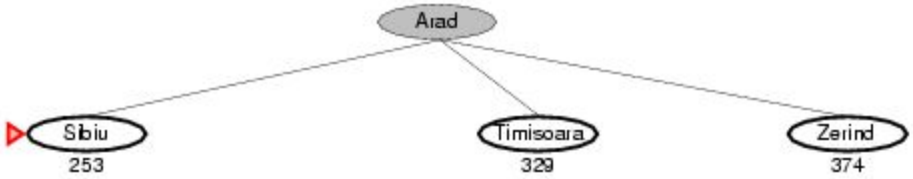
- f(n) = estimate of cost from *n* to *goal*
- e.g., *f(n)* = straight-line distance from *n* to Bucharest
- Greedy best-first search expands the node that appears to be closest to goal.

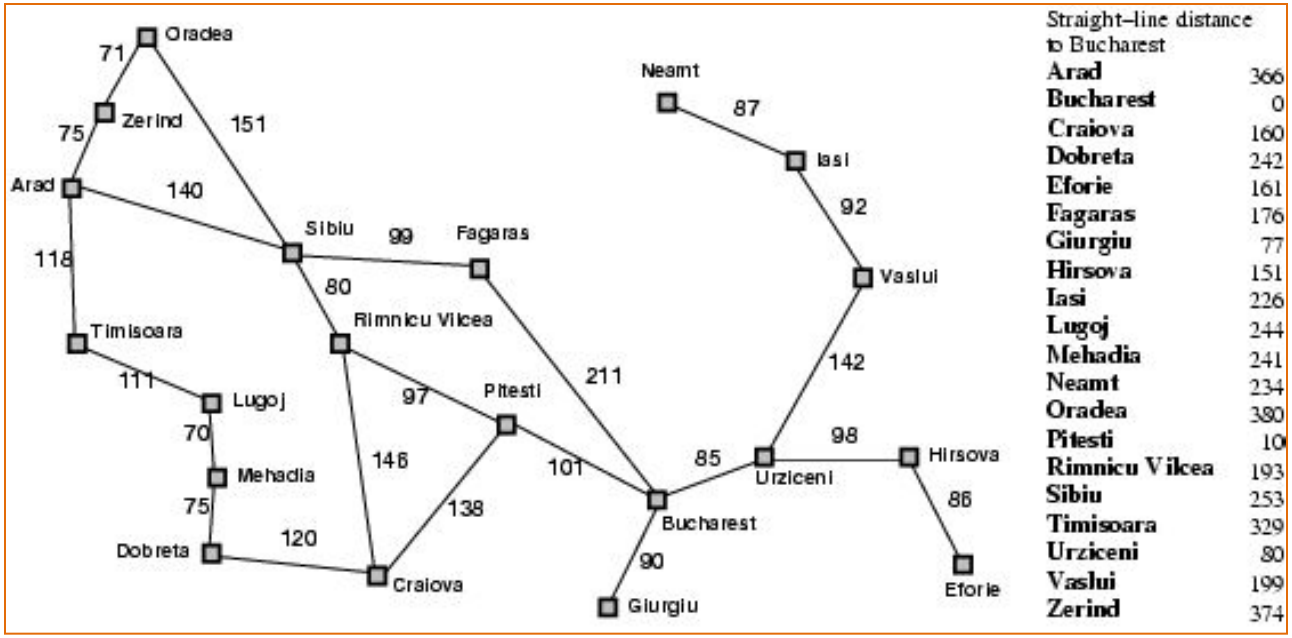
Greedy best-first search example



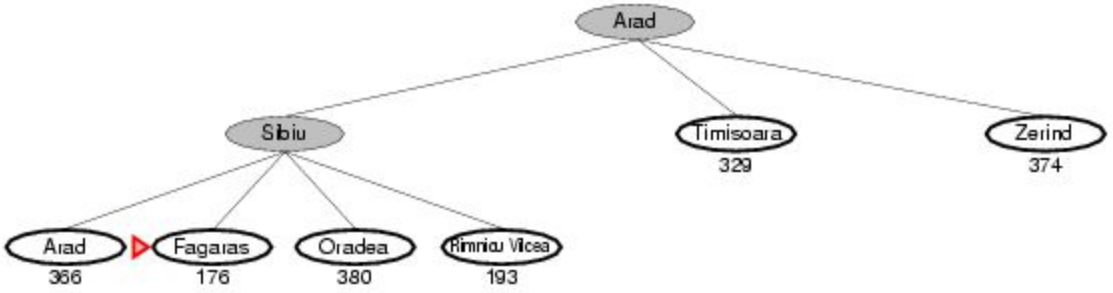
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

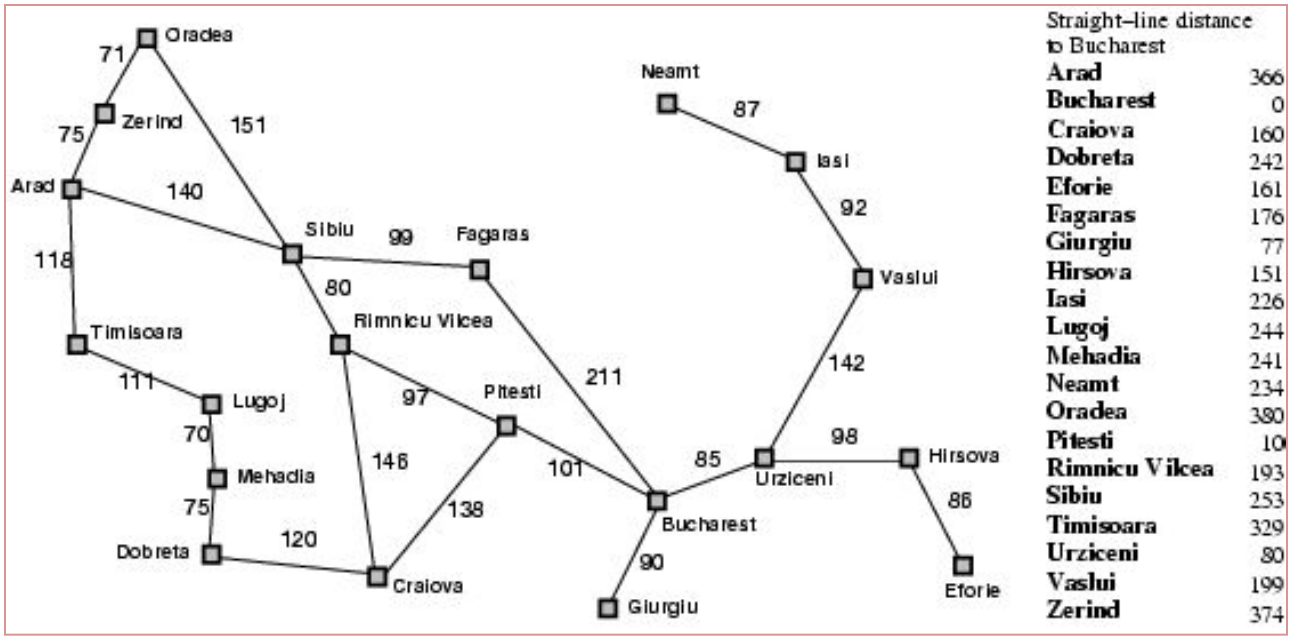
Greedy best-first search example





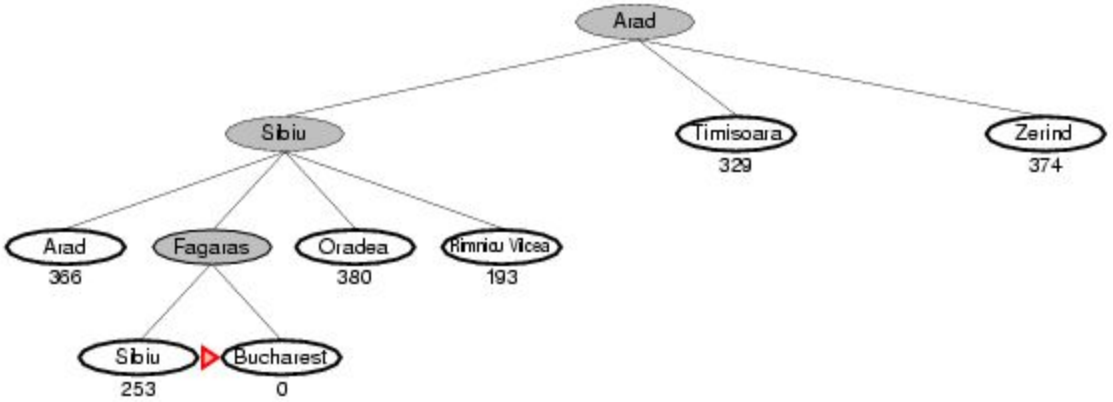
Greedy best-first search example

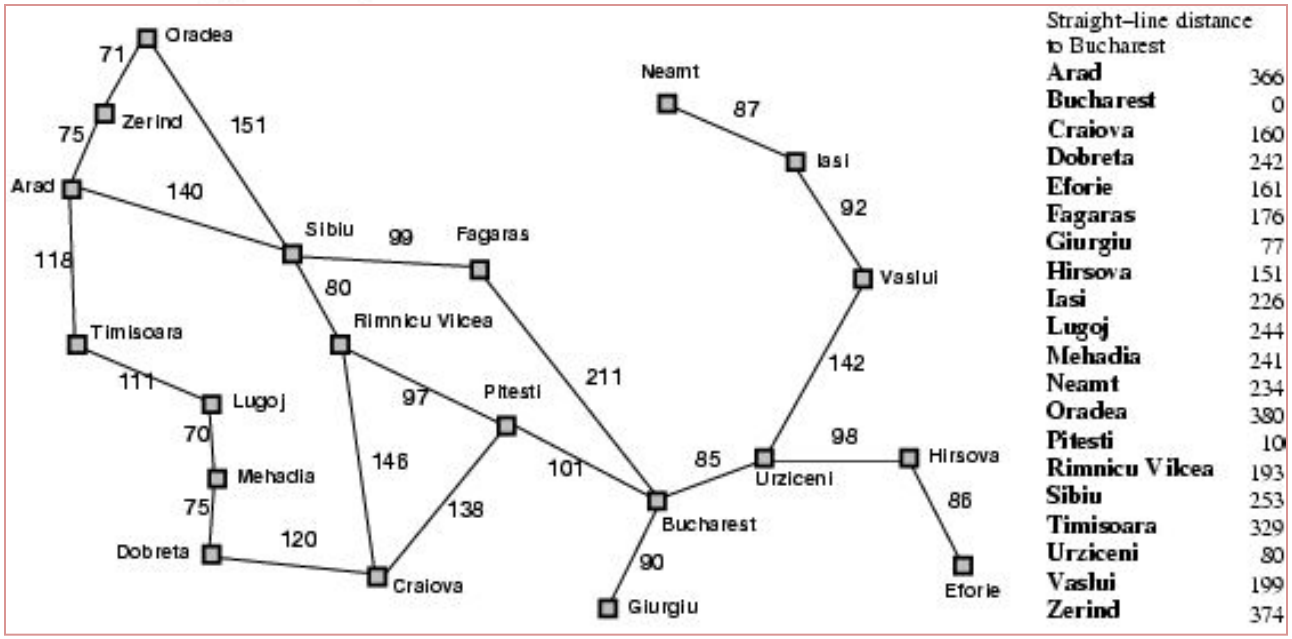




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

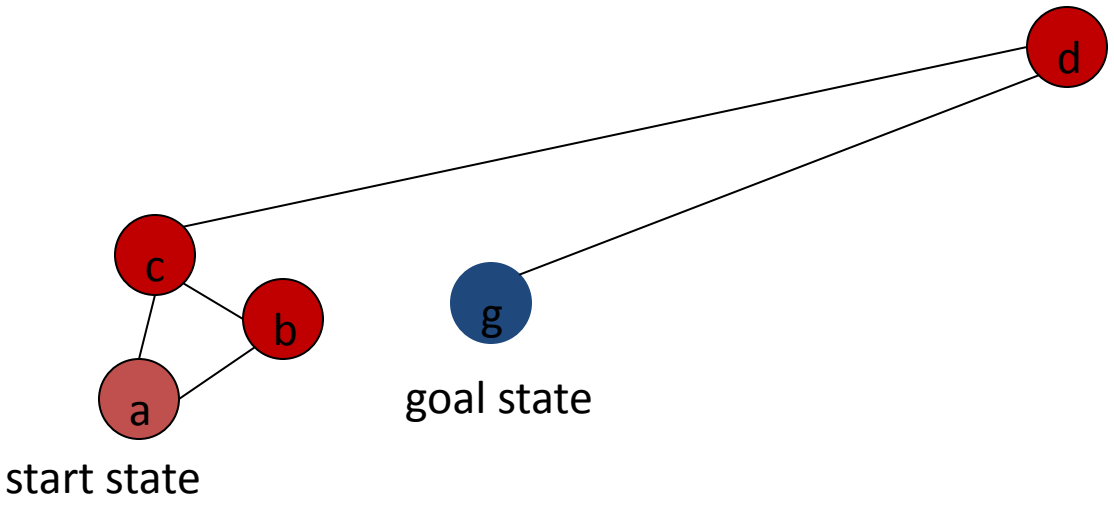
Greedy best-first search example





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

GBFS is not complete



f(n) = straightline distance

Properties of greedy best-first search

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

e.g. Arad □ Sibiu □ Rimnicu
Virea □ Pitesti □ Bucharest is shorter!

Unit 2 List of Topics

- Searching techniques – Uninformed search – General search Algorithm
- Uninformed search Methods – Breadth First Search

- Uninformed search Methods – Depth First Search
- Uninformed search Methods – Depth limited Search

- *Uniformed search Methods- Iterative Deepening search*
- *Bi-directional search*

- *Informed search- Generate and test, Best First search*
- *Informed search-A* Algorithm*

- *AO* search*
- *Local search Algorithms-Hill Climbing, Simulated Annealing*

- *Local Beam Search*
- *Genetic Algorithms*

- *Adversarial search Methods-Game playing-Important concepts*
- *Game playing and knowledge structure.*

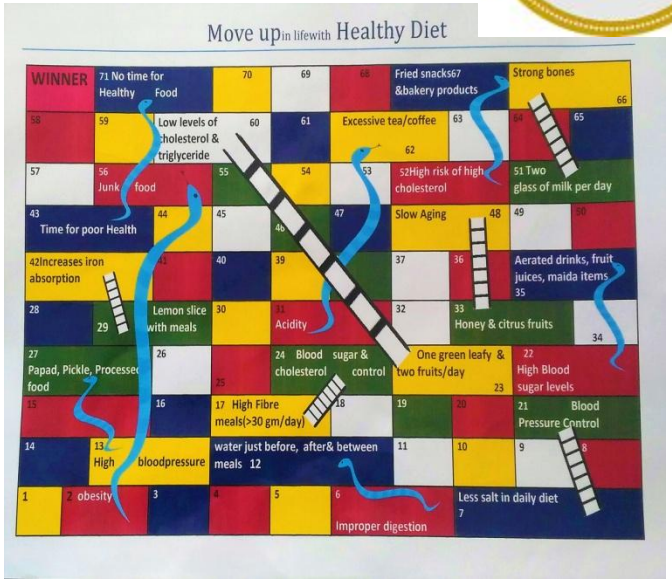
- *Game as a search problem-Minimax Approach*
- *Minimax Algorithm*

- *Alpha beta pruning*
- *Game theory problems*

A* search

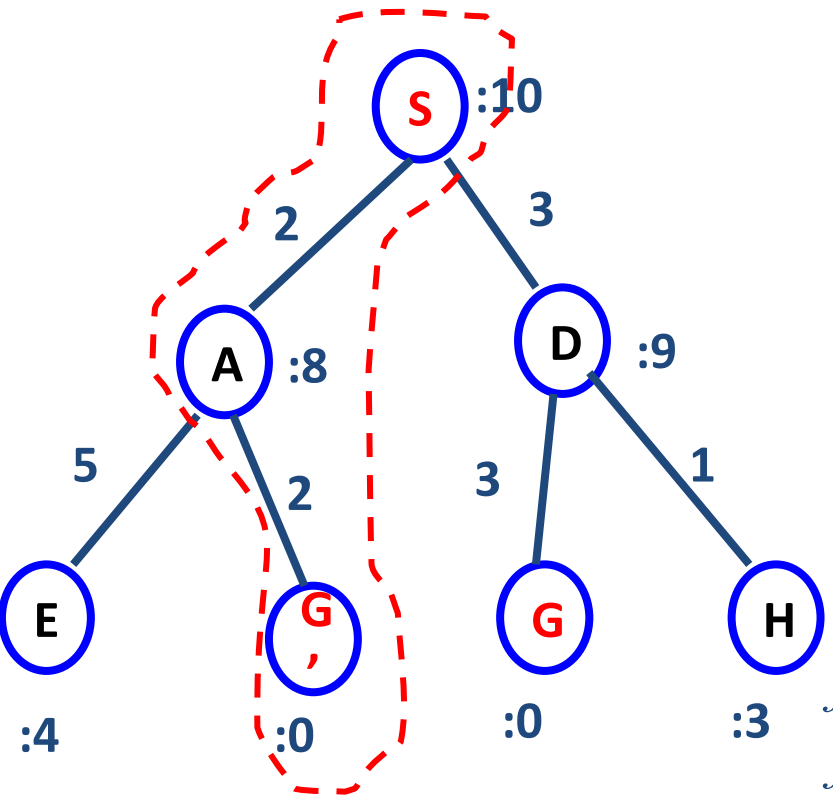
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal
- Best First search has $f(n)=h(n)$
- Uniform Cost search has $f(n)=g(n)$

Applications of Heuristic Search : A* Algorithm



- Widely known algorithm – (pronounced as "A star" search).
- Evaluates nodes by combining g(n) "cost to reach the node" and h(n) "cost to get to the goal"
- $f(n) = g(n) + h(n)$, $f(n)$ □ estimated cost of the cheapest solution.
- Complete and optimal ~ since evaluates all paths.

Heuristic Search : A* Algorithm



* Path S-A-G' is chosen = Lowest cost

Path cost for S-D-G

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

$= 0 + 10 = 10$

$f(D) = (0+3) + 9 = 12$

$f(G) = (0+3+3) + 0 = 6$

Total path cost $= f(S)+f(D)+f(G) = 28$

Path cost for S-A-G'

$f(S) = 0 + 10 = 10$

$f(A) = (0+2) + 8 = 10$

$f(G') = (0+2+2) + 0 = 4$

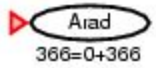
Total path cost $= f(S)+f(A)+f(G') = 24$

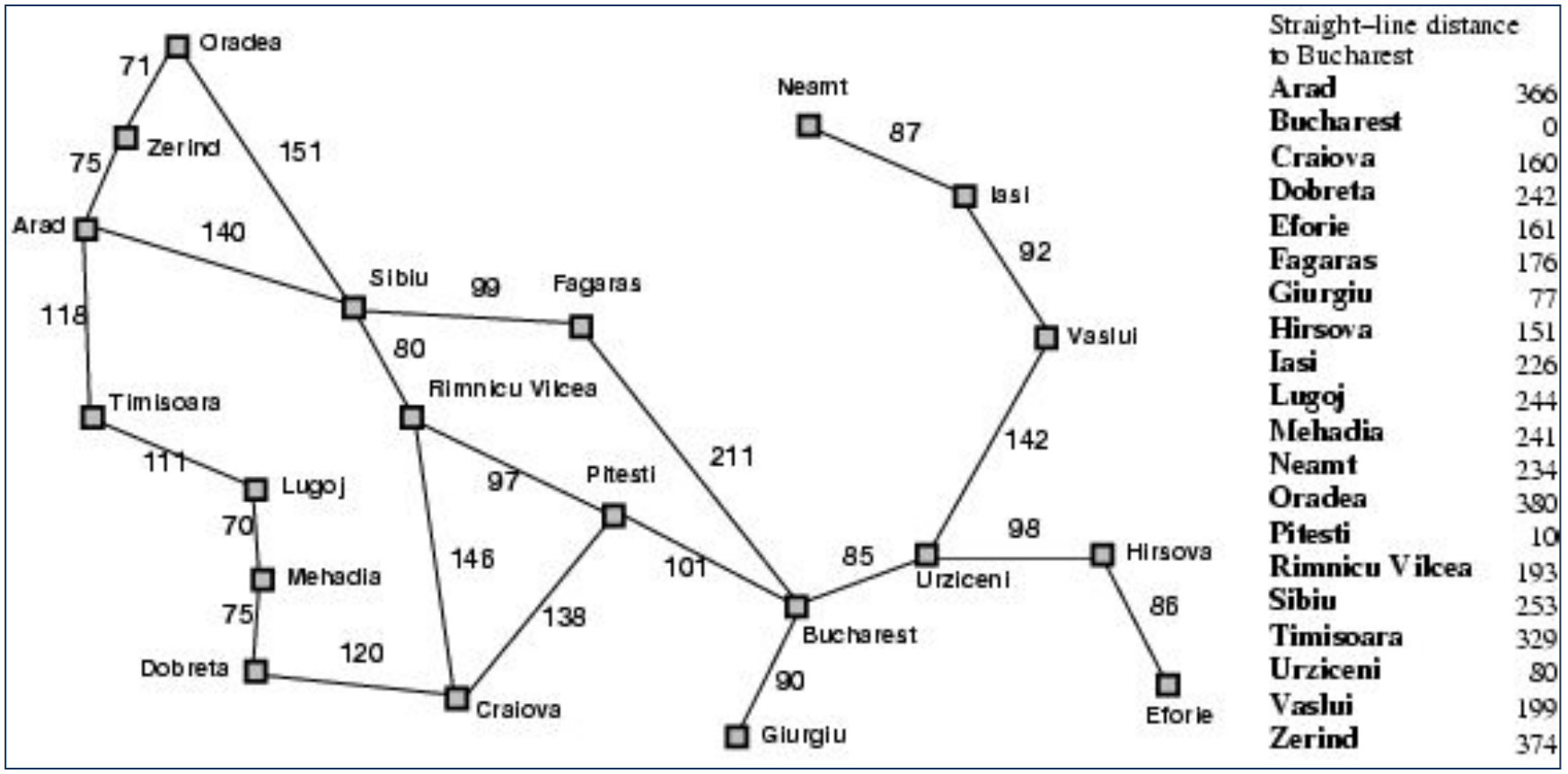
Application of Heuristic Search : A* Algorithm – Snake & Ladder

12 h(n)=0	11	10 g(n)=1, h(n)=2, f(n)=3	9 g(n)=8, h(n)=8, f(n)=16
8	7		5
1 g(n)=0, h(n)=11, f(n)=11	2 g(n)=1, h(n)=2, f(n)=3	3 g(n)=2, h(n)=9, f(n)=11	4 g(n)=3, h(n)=8, f(n)=11

1	2 (10)	3	4	5	6	7
2 (10)	3	4	5	6	7	8
3	4	5	6	7	8	9 (4)
4	5	6	7	8	9 (4)	10
5	6	7	8	9 (4)	10	11
6	7	8	9 (4)	10	11	12
7	8	9 (4)	10	11	12	X
8	9 (4)	10	11	12	X	
9 (4)	10	11	12	X		
10	11	12	X			
11	12	X				
12	X					

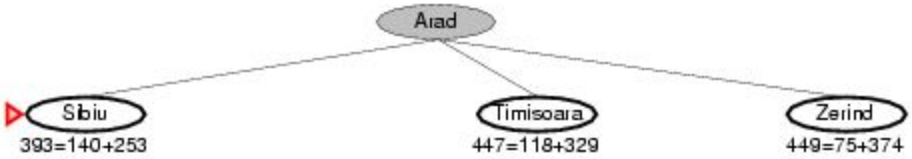
A* search example

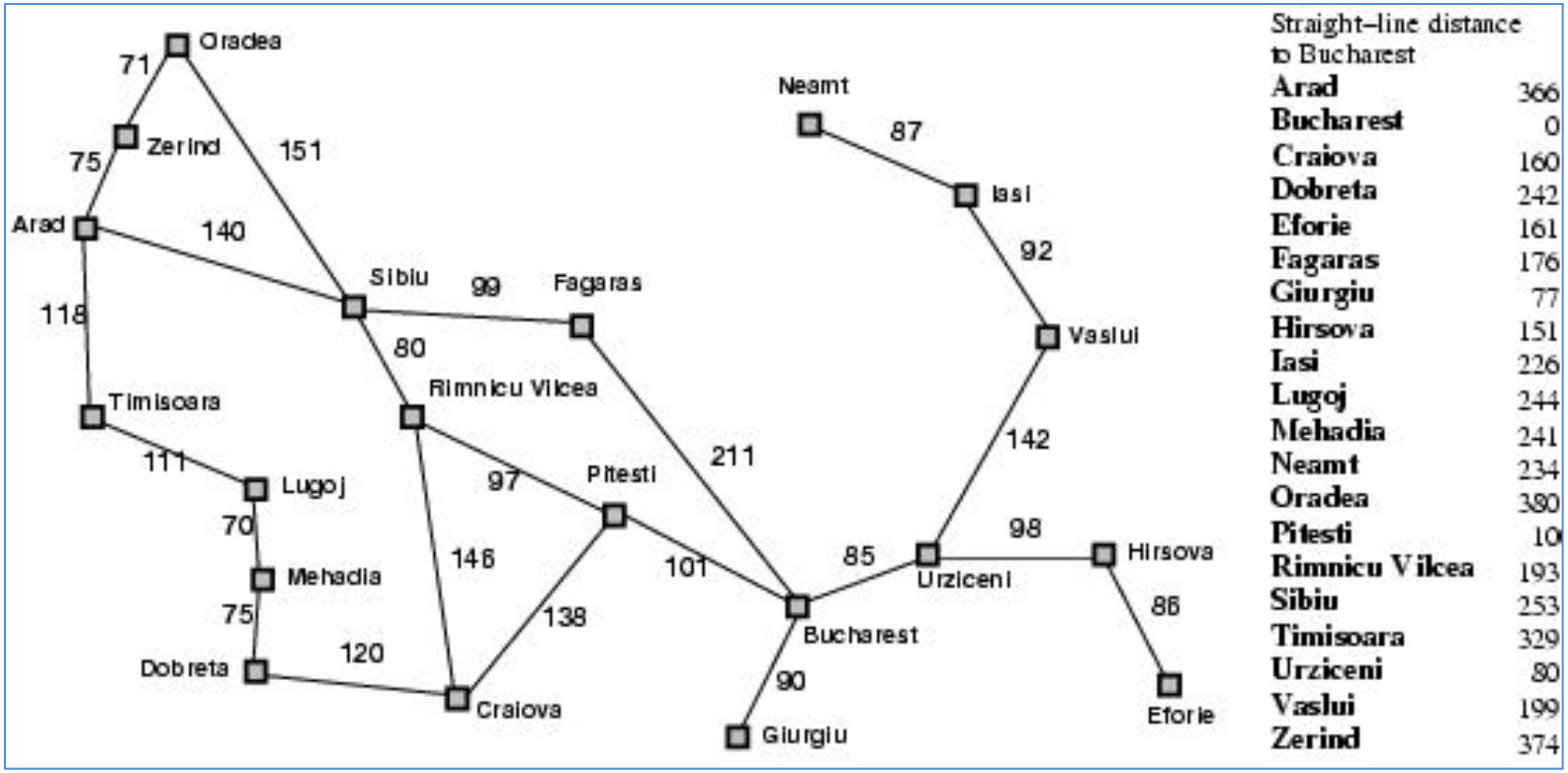




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

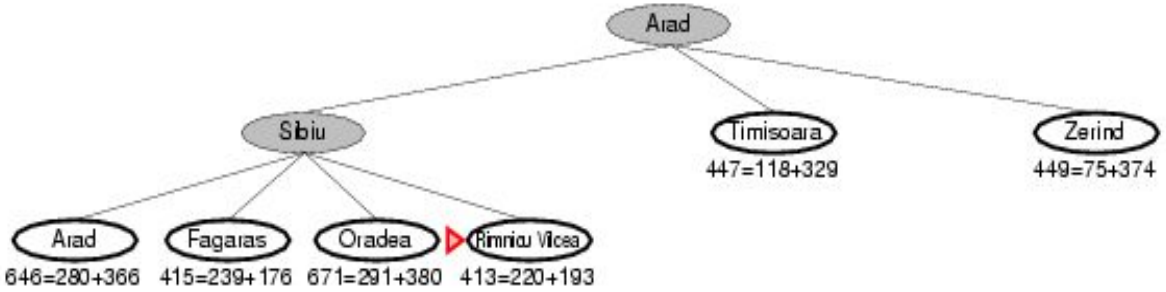
A* search example

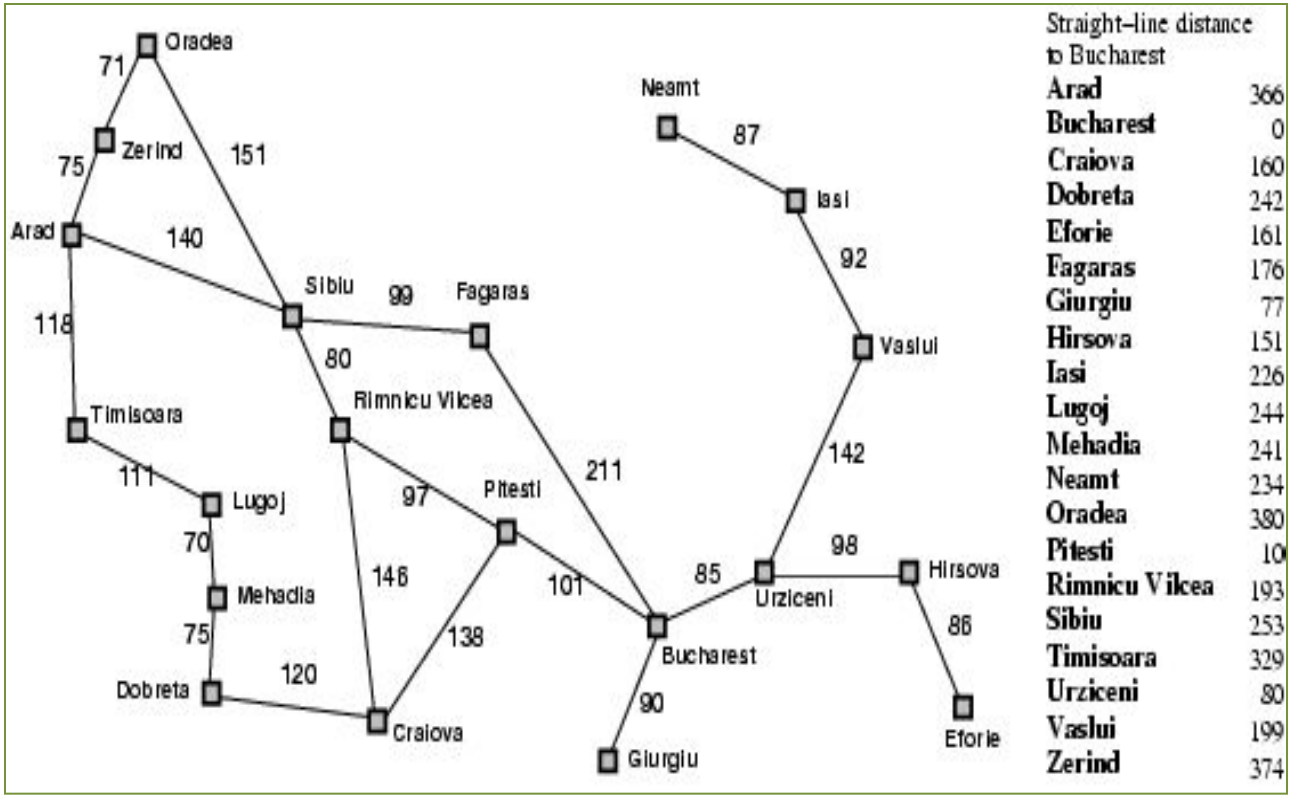




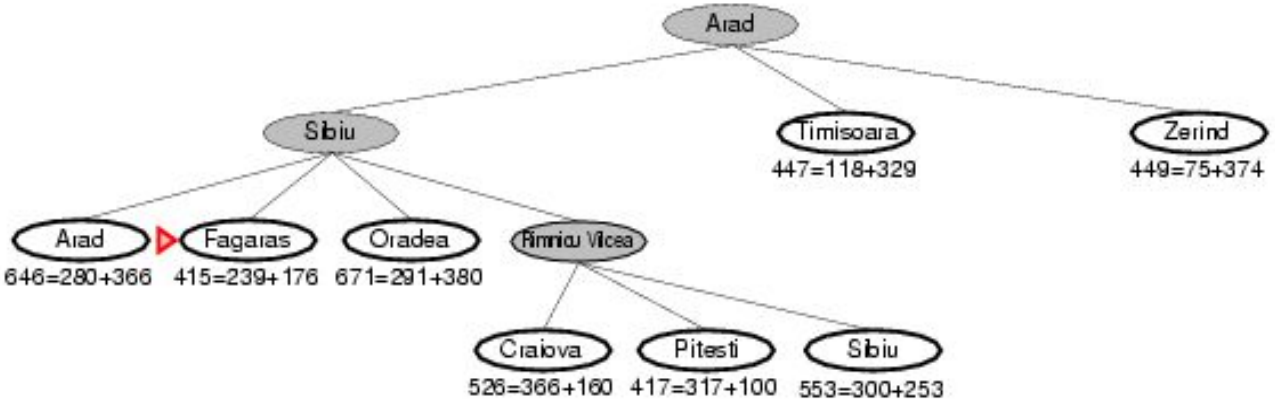
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

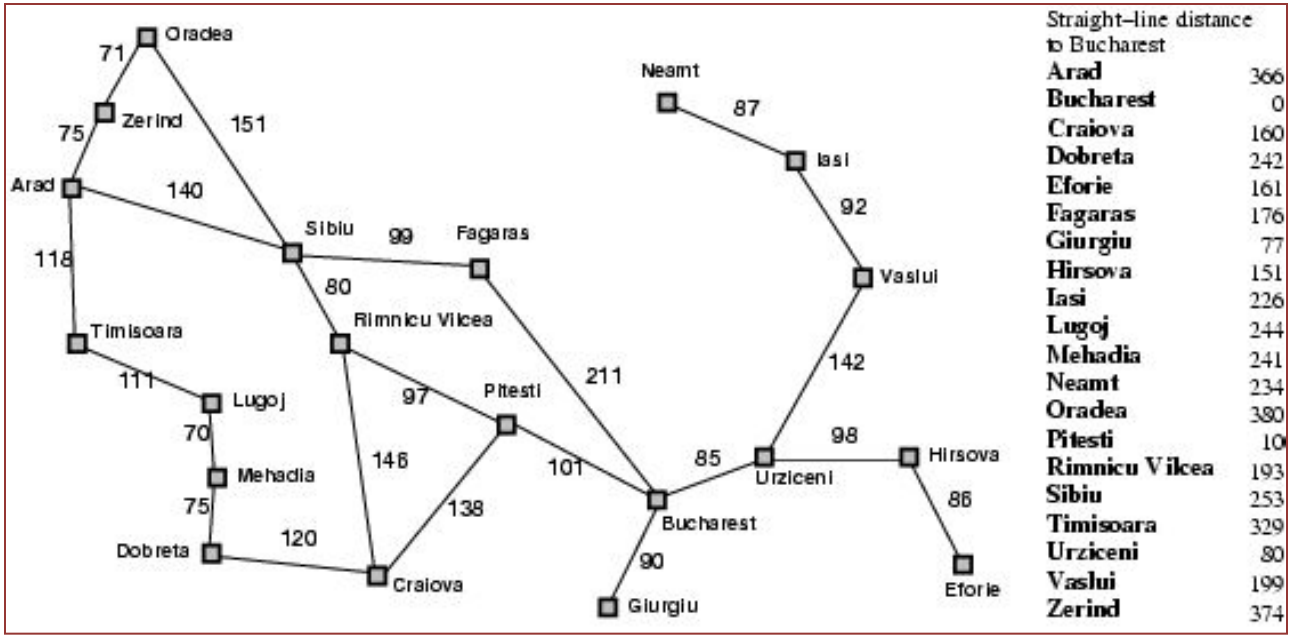
A* search example





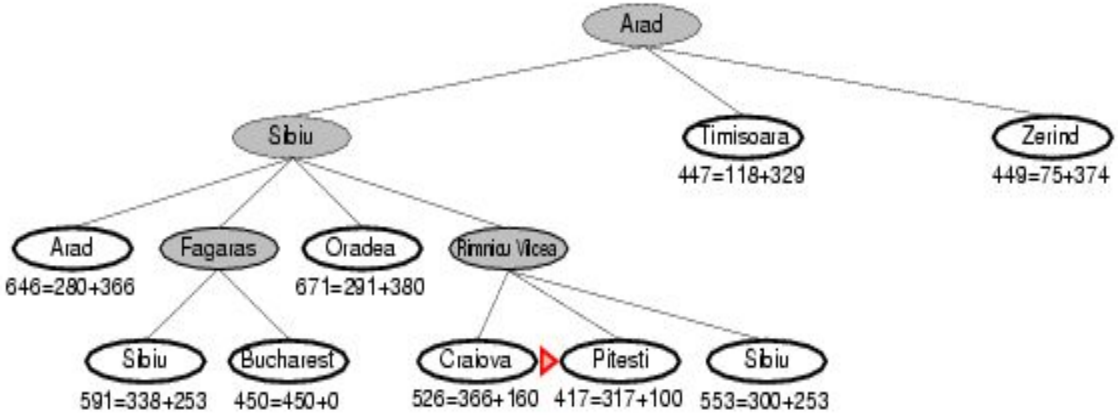
A* search example

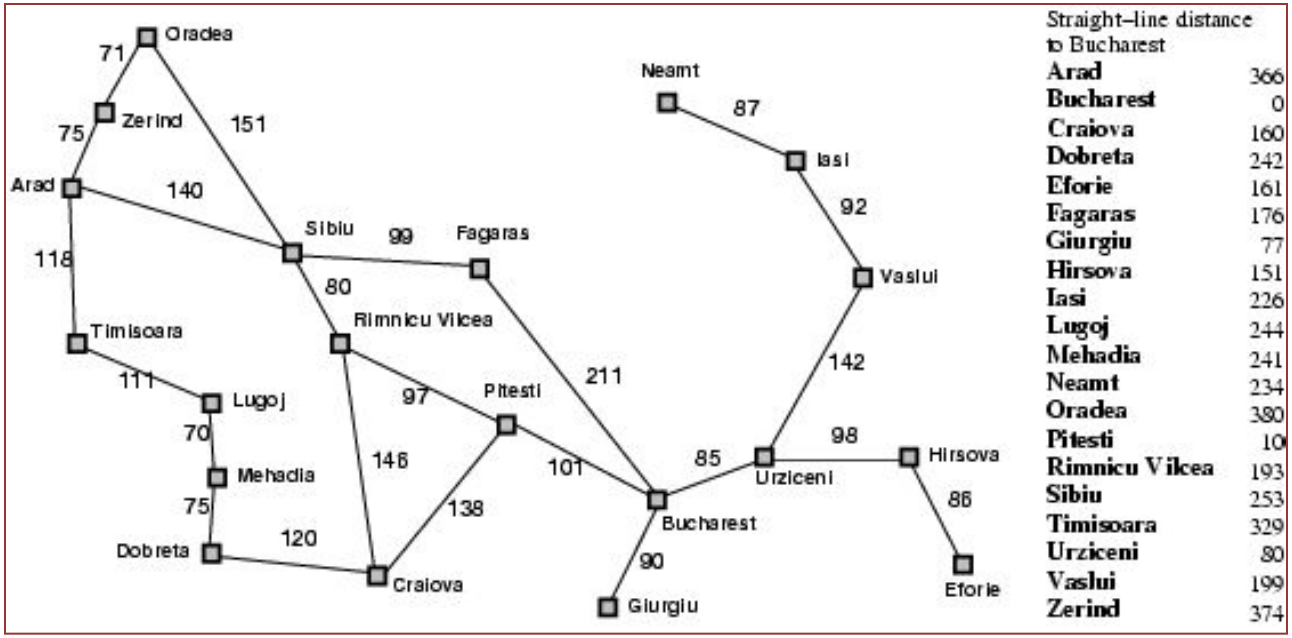




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

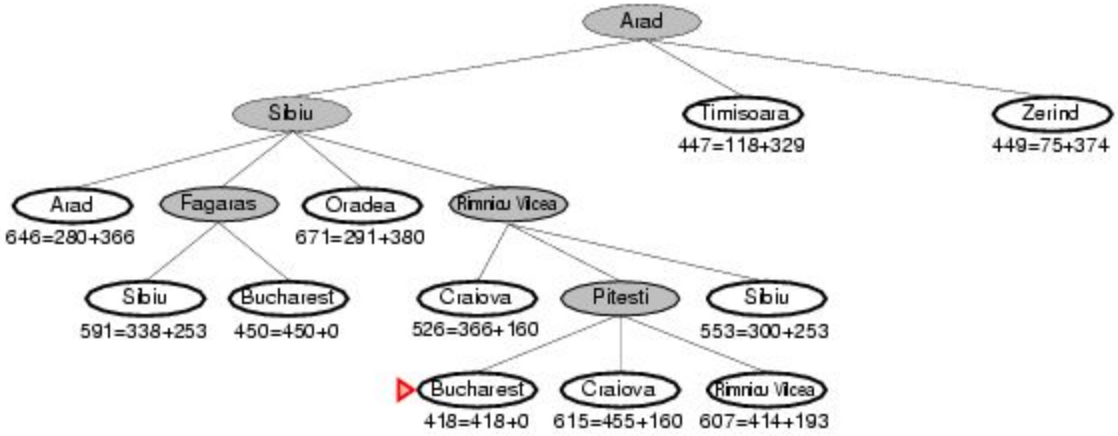
A* search example

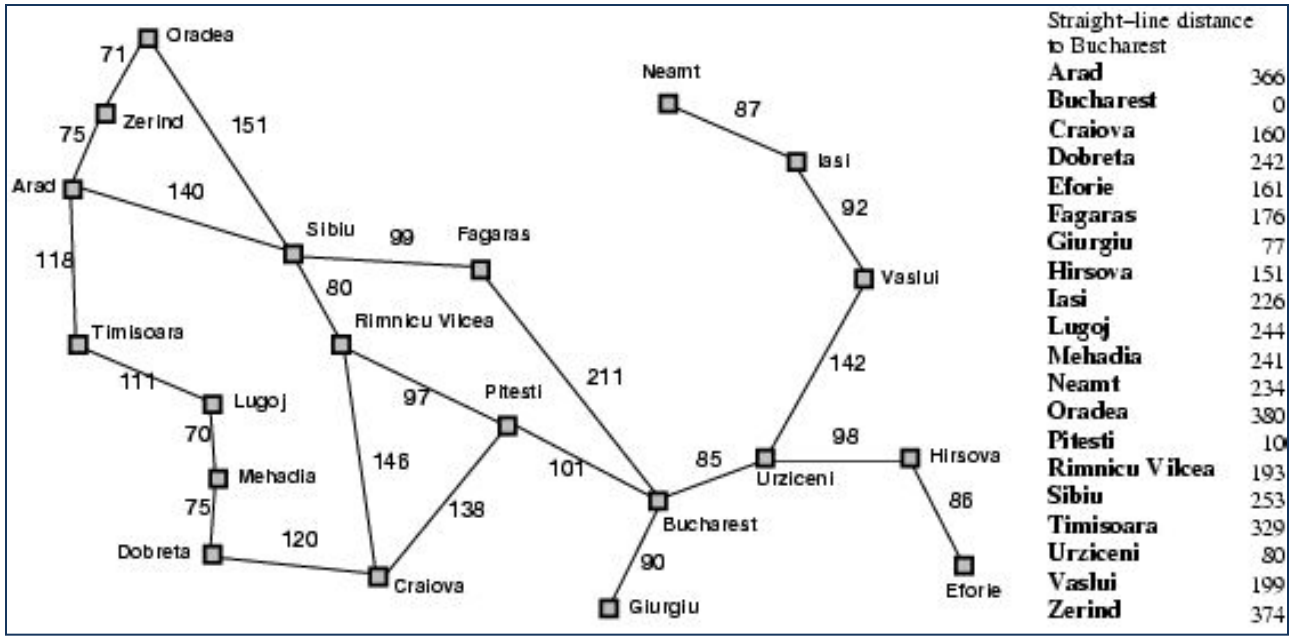




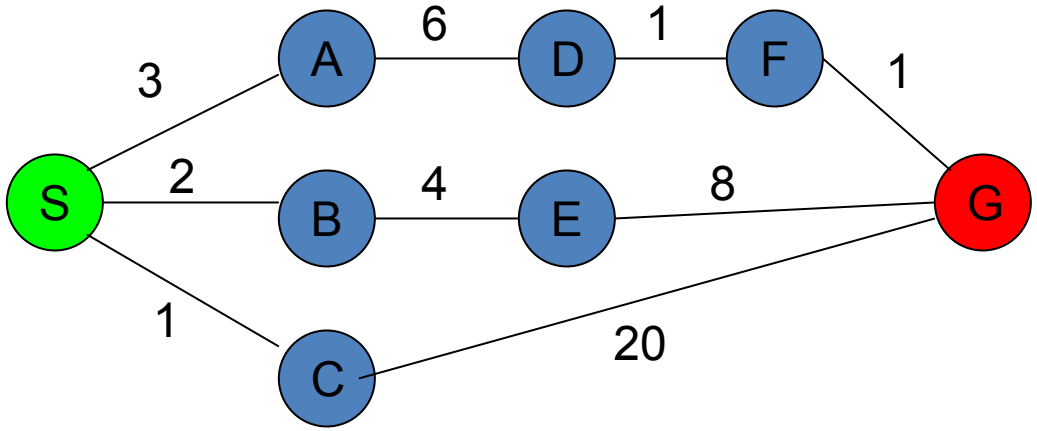
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example





try yourself !



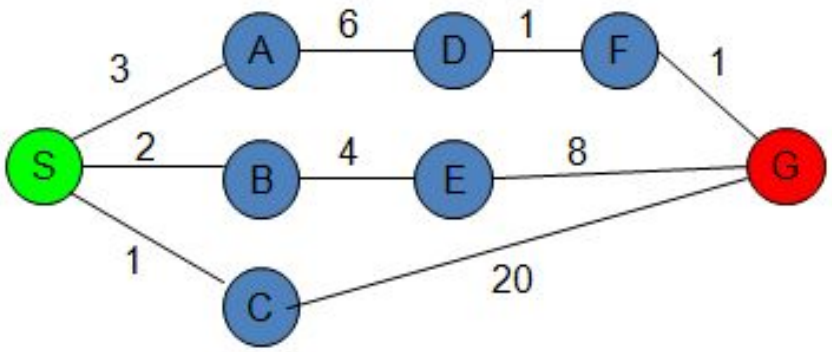
straight-line distances

h(S-G)=10
h(A-G)=7
h(D-G)=1
h(F-G)=1
h(B-G)=10
h(E-G)=8
h(C-G)=20
h(G-G)=0

The graph above shows the step-costs for different paths going from the start (S) to the goal (G). On the right you find the straight-line distances.

1. Draw the search tree for this problem. *Avoid repeated states*.

1. Give the order in which the tree is searched (e.g. S-C-B...-G) for A* search.
   Use the straight-line dist. as a heuristic function, i.e. h=SLD,
   and indicate for each node visited what the value for the evaluation function, f, is.



straight-line distances
h(S-G)=10
h(A-G)=7
h(D-G)=1
h(F-G)=1
h(B-G)=10
h(E-G)=8
h(C-G)=20
h(G-G)=0

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

F(S) = 0 + 10 = 10

F(A) = 3 + 7 = 10

F(B) = 2 + 10=12

F© = 1+20 = 21

F(D) = 9 + 1 = 10

F(F) = 10+1 = 11

F(G) = 11+0 = 11

S-A-D-F-G

Admissible heuristics

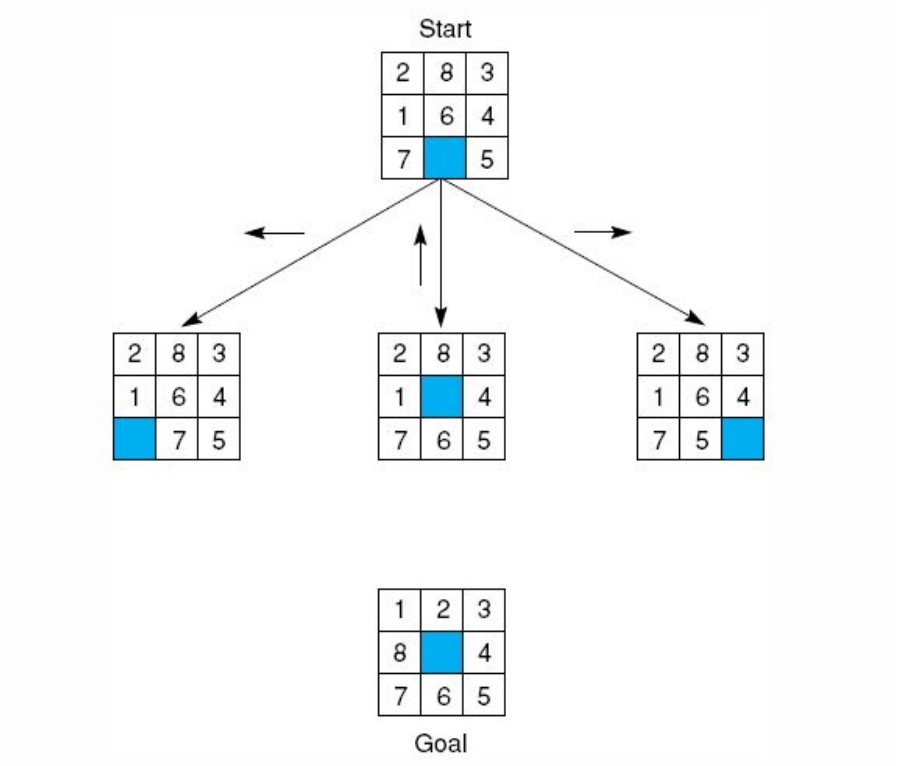
- A heuristic *h(n)* is admissible if for every node *n*, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from *n*.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)
- Theorem: If *h(n)* is admissible, A* using `TREE-SEARCH` is optimal

Admissible heuristics – 8 Puzzle Problem

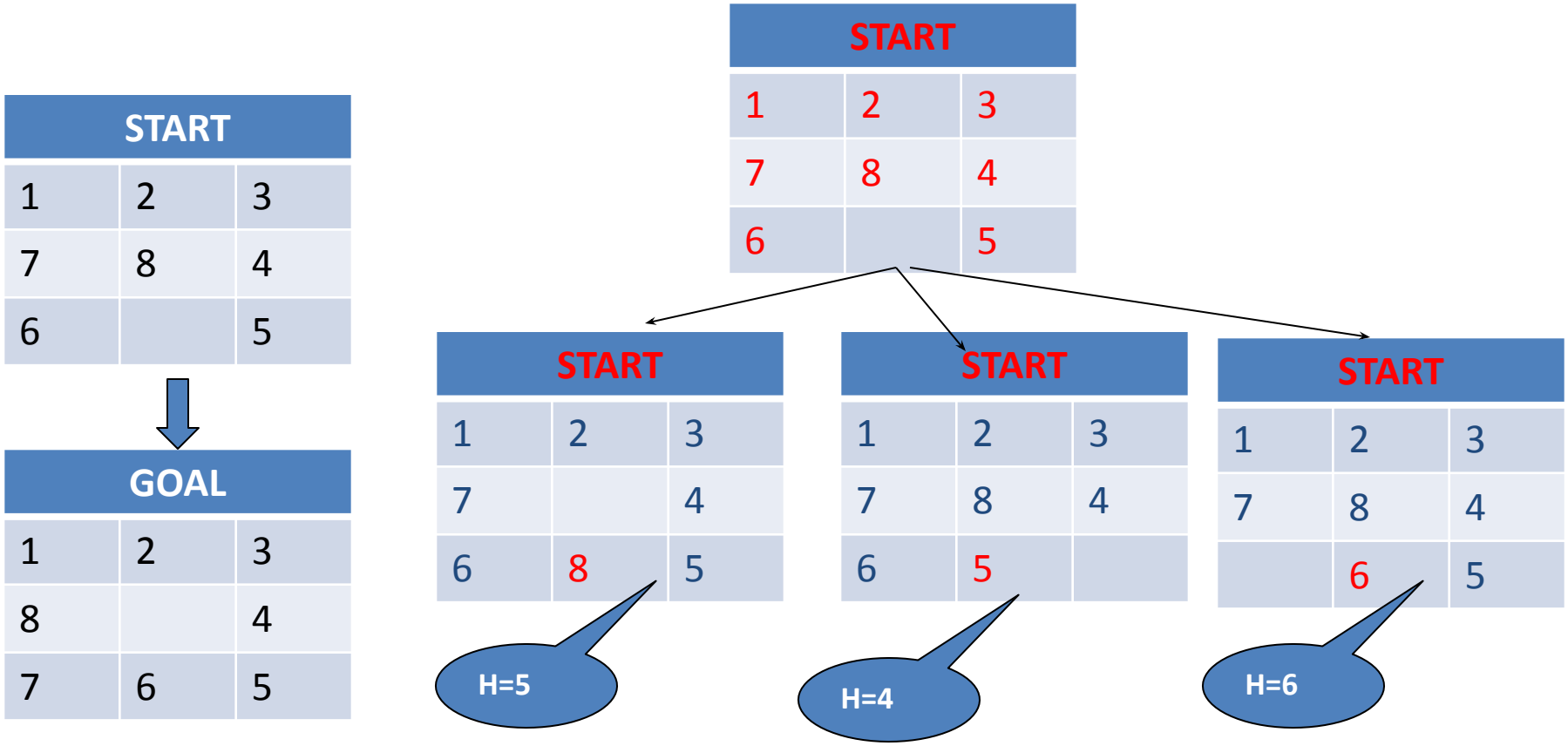
Try it out!

START		
1	2	3
7	8	4
6		5

GOAL		
1	2	3
8		4
7	6	5



Admissible heuristics – 8 Puzzle Problem



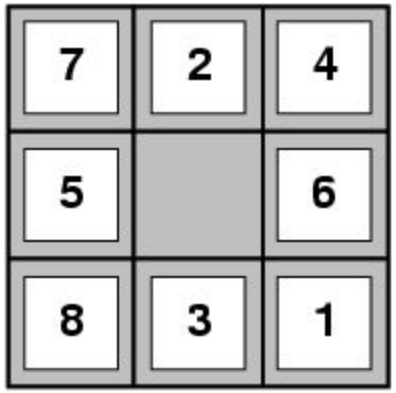
Admissible heuristics

E.g., for the 8-puzzle:

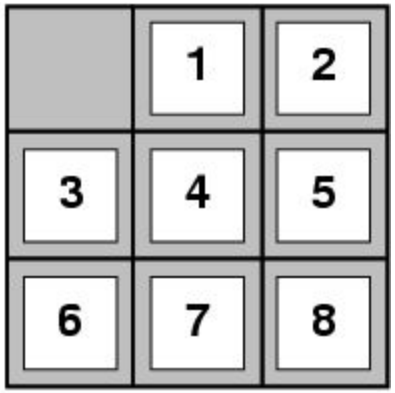
- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

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

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



Start State



Goal State

Evaluation of Search Algorithms

Completeness

Is the algorithm guaranteed to nd a solution if one exists?

Optimality

When the algorithm nds a solution, is this the optimal one?

Time complexity

How long does it take to nd a solution?[a]

[a] Often measured as number of nodes generated during search.

Space complexity

How much memory is needed to perform the search?[a]

[a] Often measured as maximum number of nodes stored in memory.

Evaluation of Search Algorithms

Parameters	BFS	DFS	UCS	IDS
T(n)	O(b^d)	O(b^d)	O(b^ ceil(c*/e))	O(b^d)
S(n)	O(b^d)	O(d)	O(b^d)	O(d)
Completeness	Complete	It is not always complete. It is not complete if the depth is very high.	Complete	
Optimality	Guaranteed	Not Guaranteed	Guaranteed	

END
UNIT-2