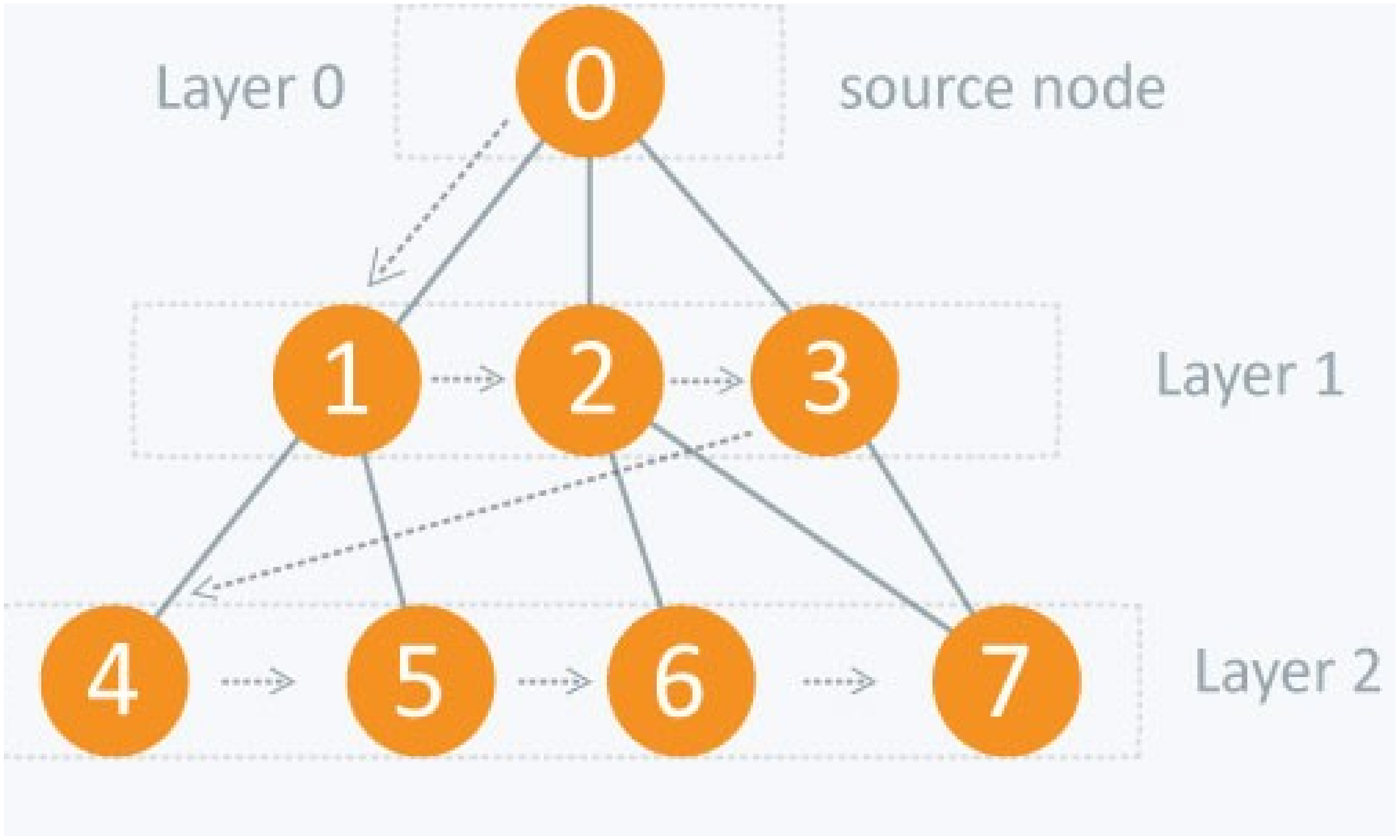
LECTURE 8:Breadth First Search

BFS like DFS is a search algorithm for transversing or searching a tree or graph data structures

In BFS the search starts at the root and explores the neighbour notes first before moving to the next level neighbours.

See graph below:

BFS graph showing search order

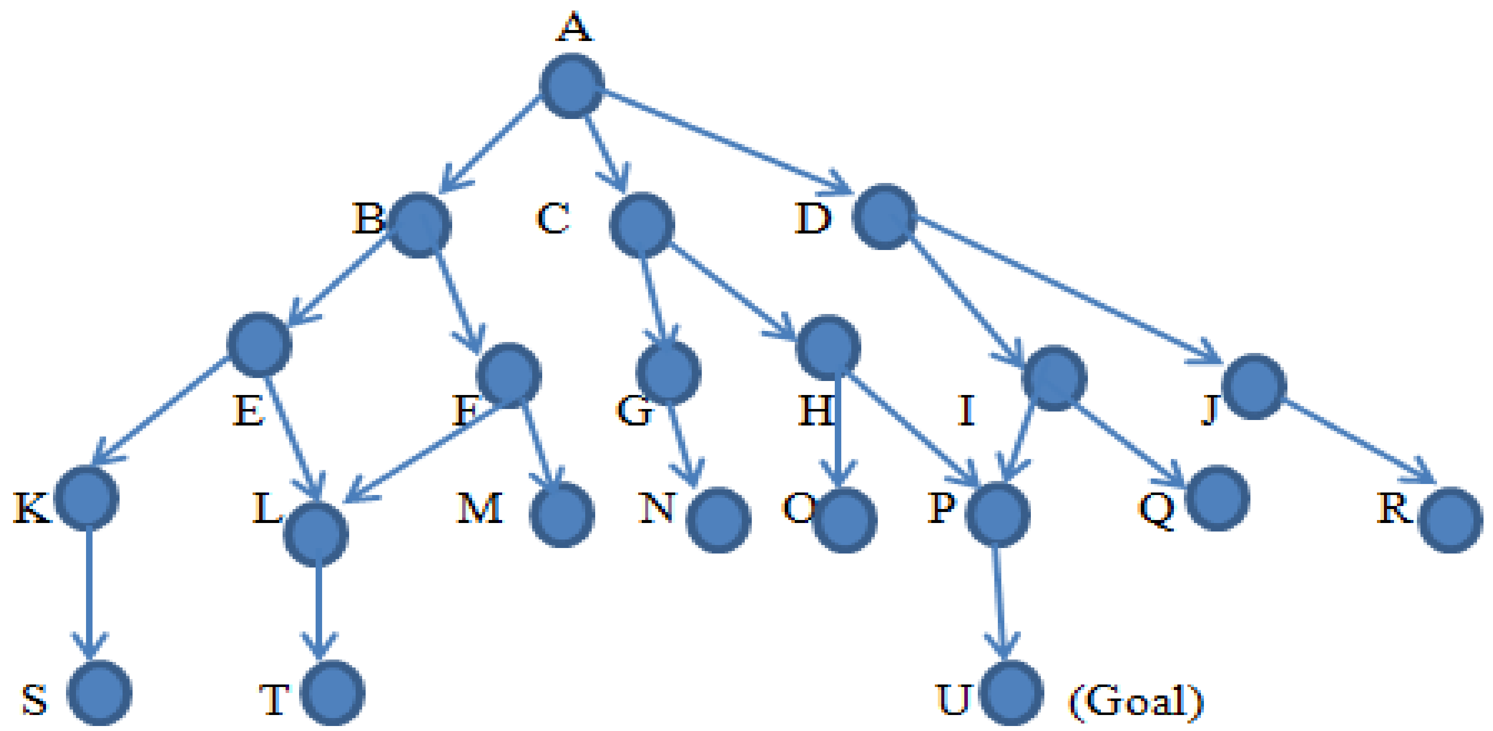


BFS graph showing search order

In BFS you are required to transvers the graph breadthwise as follows:

1) First move horizontally and visit all the nodes in the current layer
2) Move to the next layer and transverse all nodes in layer 2 before moving to layer 3

Consider earlier example of DFS



DFS and BFS searching orders

In DFS the nodes/states are examined in the order: A,B E,K,S,L,T,F,M,C,G,N,H,O,P,U,D,I,Q,J,R

In BFS the nodes/states are examined in the order

:

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U.

BFS Search algorithm(pseudo-code)

```
Function breadth-first-search
Open: = [Start]
Closed: = [ ];
While open ≠ [ ] do
Begin
Remove leftmost state from open, call
it X
If X is a goal then return SUCCESS
```

Function breadth-first-search

```
Else begin
Generate children of X;
Put X on closed;
Discard children of X if already on open or
closed;
Put remaining children on right end of open;
  end
 end
  return FAIL
End.
```

Function breadth-first-search(FIFO)

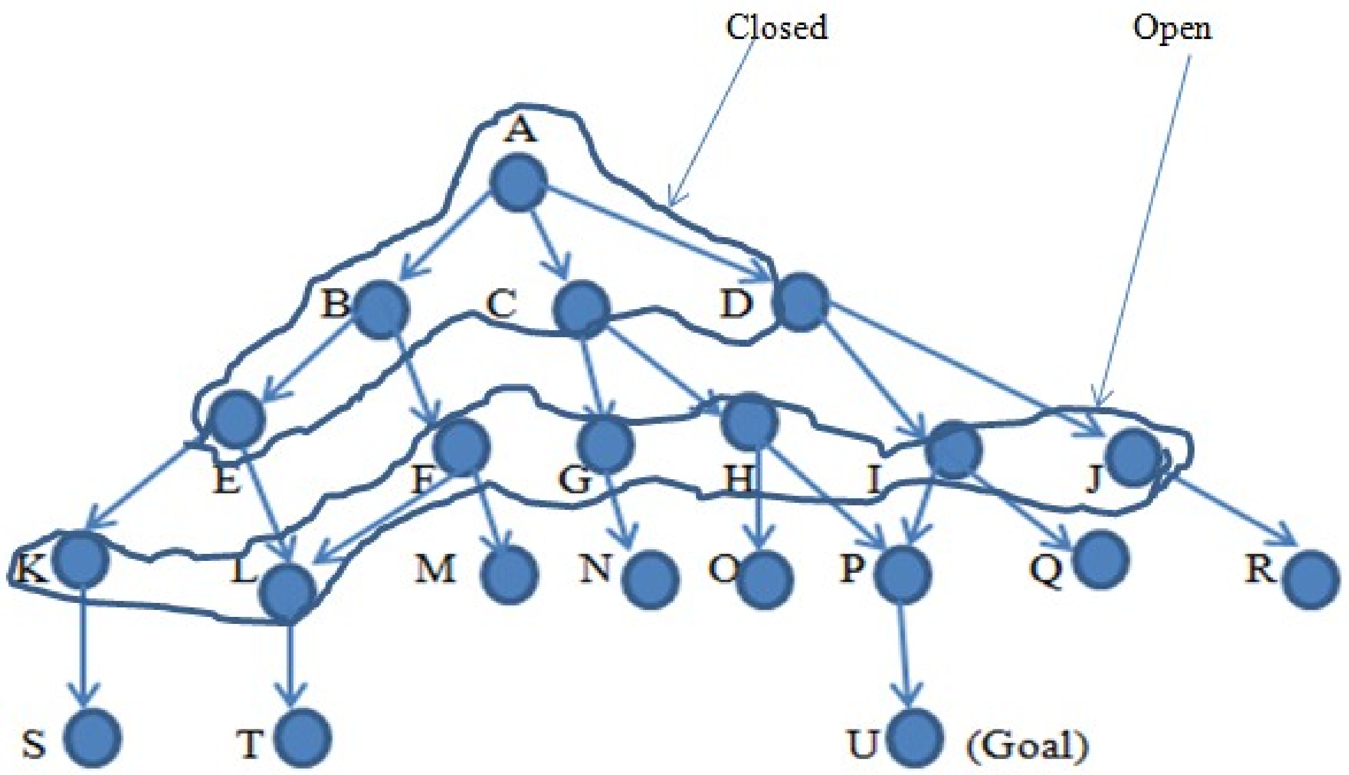
Child states are generated by inference rules, legal moves of the game, or other state transition operators.

Each iteration produces all children of state X and adds them to open.

BFS search example

1) Open = [A]; closed = [ ]
2) Open =[B,C,D]; closed [A]
3) Open =[C,D,E,F]; closed [B,A]
4) Open = [D,E,F,G,H]; closed [C,B,A]
5) Open = [E,F,G,H,I,J]; closed [D,C,B,A]
6) Open = [F,G,H,I,J,K,L]; closed [E,D,C,B,A]
7) Open = [G,H,I,J,K,L,M]; as L is already open closed[F,E,D,C,B,A]
8) Open = [H,I,J,K,L,M,N]; closed =[G,F,E,D,C,B,A]
9) And so on until either U is found or open = [ ]

Graph of previous example showing closed and open states after 6 iterations



Graph of previous example showing closed and open states after 6 iterations

BFS is guaranteed to find the shortest path from start to the goal.

Class activity

Consider the 8-puzzle below:

Use BFS search to find the goal

Initial State

2	8	3
1	6	4
7		5

Goal State

1	2	3
8		4
7	6	5

Informed Breadth First Search

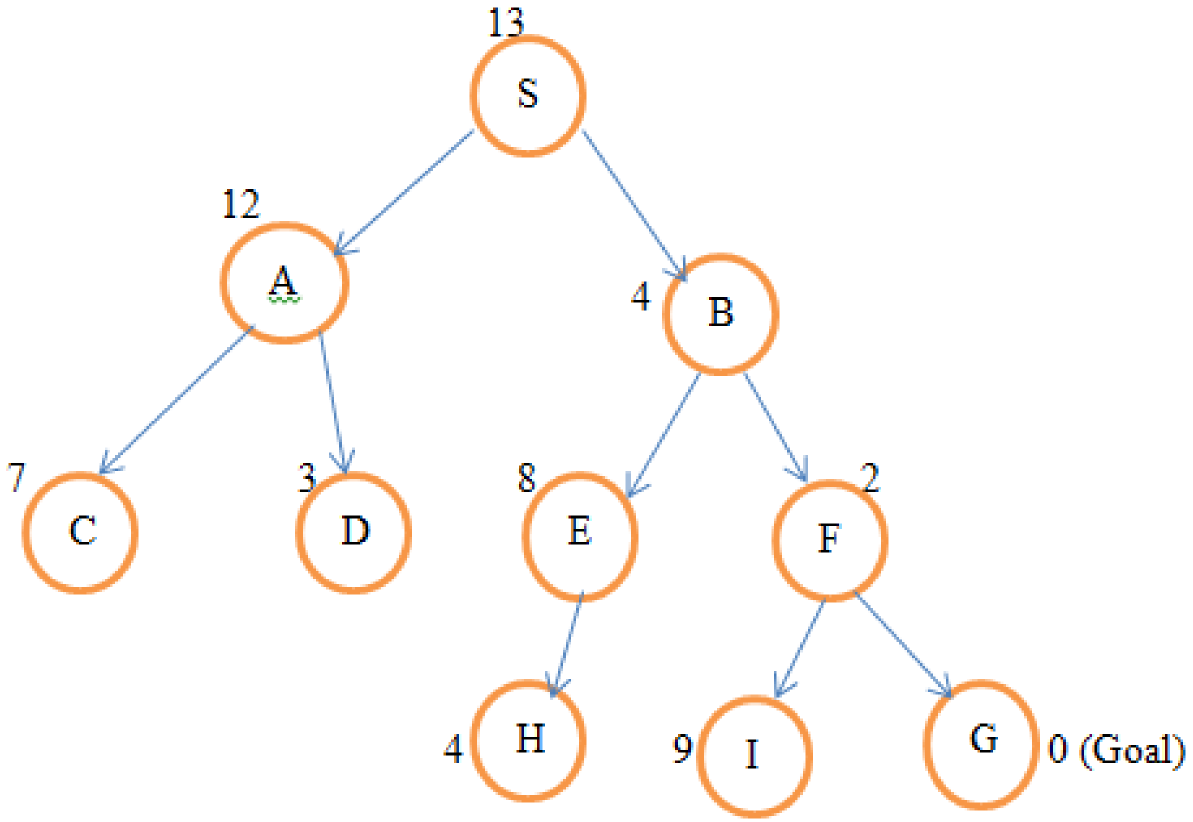
Also called informed BFS

This search algorithm explores the graph by expanding the most promising node chosen according to a specific value(heuristic value)

Called informed search because there is more information given.

The algorithm works with two lists just like the blind search: open list and closed list

Example of informed BFS



Node	h(n)
	13
	12
	4
	7
	3
	8
	2
	4
	9
	0

Example of informed BFS

S is the Root

h(n) is the estimated value from current node to the Goal

From S, the neighbouring nodes are A and B

Compare their values(distance to the goal) and choose the one with the smaller value. B is chosen and placed on closed, A remains on open. Repeat the same procedure at node B

Example of informed BFS algorithm

Open = [A,B]; close [S]

Open = [A]; close [S,B]

Open = [E,F,A]; close [S,B]

Open = [E,A]; close [S,B,F]

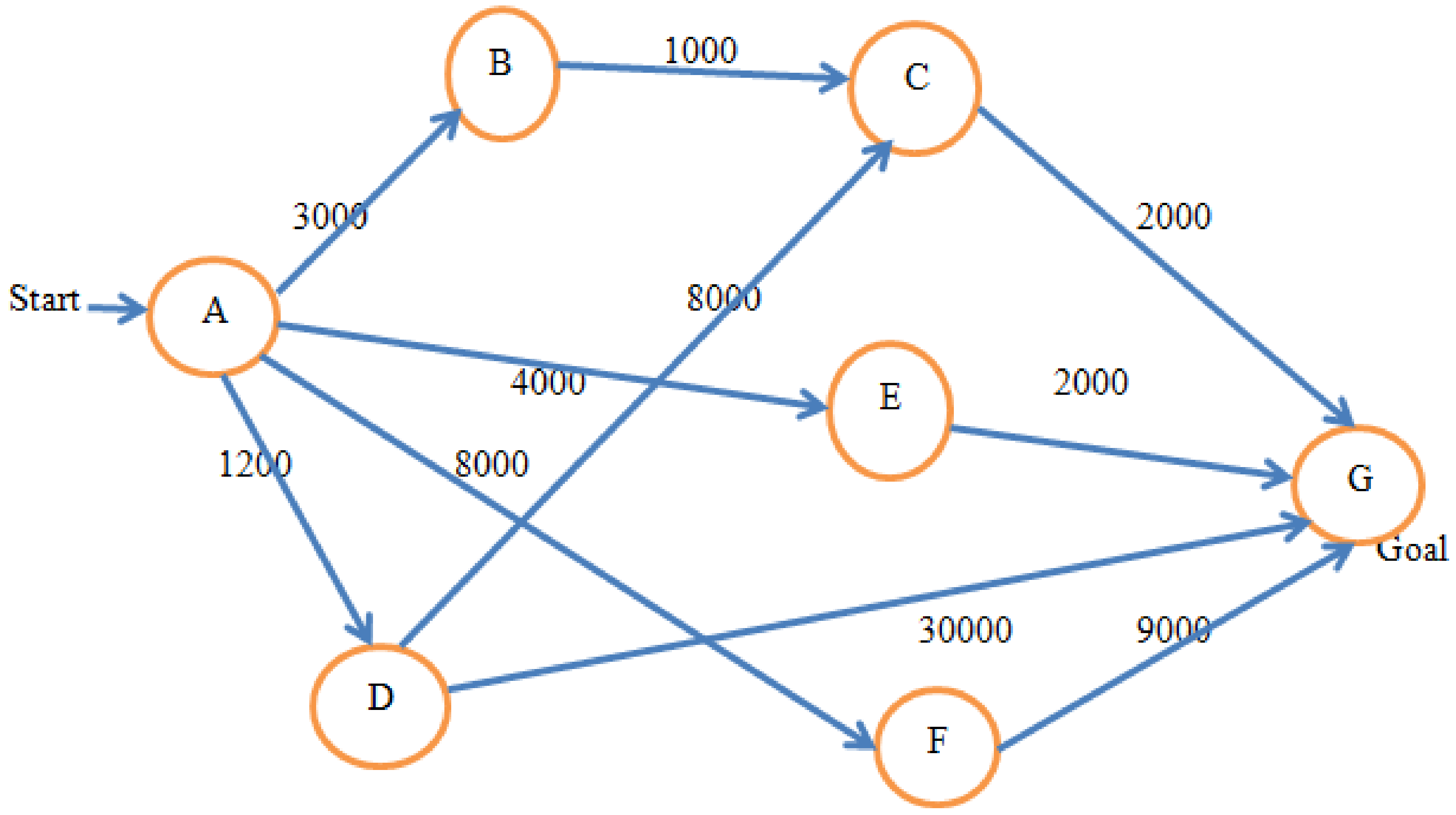
Open = [E,A,I,G]; close [S,B,F]

G is removed from open since its value is 0 meaning it’s the goal

Open = [E,A,I]; close [S,B,F,G]

So the path to the Goal is: S→B→F→G

Class activity: Calculate shortest path to the goal using informed BFS



State space generated in a heuristic search of the 8-puzzle graph

The heuristic function is made up of two parts:

F(n) = g(n) + h(n): g(n) is the actual distance from n to the start state. In the BFS g(n) represents the level(s) and h(n) represents the number of tiles out of place. For example if given the initial state and the goal state, one can always calculate the number of tiles out of position at any state.

State space generated in a heuristic search of the 8-puzzle graph

Consider the Initial state and the Goal state of the 8-puzzle graph below:

Initial State

2	8	3
1	6	4
7		5

Goal State

1	2	3
8		4
7	6	5

State space generated in a heuristic search of the 8-puzzle graph

Initial state call it state a and the function is f(a).

The value of f(a) is given by the number of tiles out of place e.g. f(a) = 4 indicating the number of tiles out of place i.e. tile numbers: 1, 2, 6 and 8. The g(n) = 0 since the initial state is level 0.

Activity

Complete the 8-puzzle space diagram.

The A* Algorithm

The A* is an extended Breadth First Search(BFS) algorithm that prioritizes the shortest routes that it can reach first and then the other routes.

Is an informed search algorithm

It is an optical and complete

- Optimal means- A* algorithm is sure to give the best solution that is available for the problem
- Complete means- A* algorithm is sure to find all possible solutions that exist for the problem

Why the A* Algorithm

A* is slow when compared to other algorithms as it searches all the paths that are available to us

Comparison between A* and Dijkstra algorithm:

Dijkstra algorithm finds all the paths that can be taken

Why the A* Algorithm

Dijkistra does not know when to stop as it does not know which is the best path and tends to compute inefficiently

A* compute the best possible way/route/path and stops when the best route is found

This makes the computations effective and efficient

The A* formula

The A* optimizes the path by calculating the least cost from one node to the next

The formula used by A* is F = G + H

This formula is considered heart and soul of the A* algorithm

The G and H help in optimizing the efficient path

The A* formula

F – parameter of the A* is used to find the least cost from the one node to the other

F – responsible for finding the optimal path between source and destination

G- is the cost of moving from one node to other

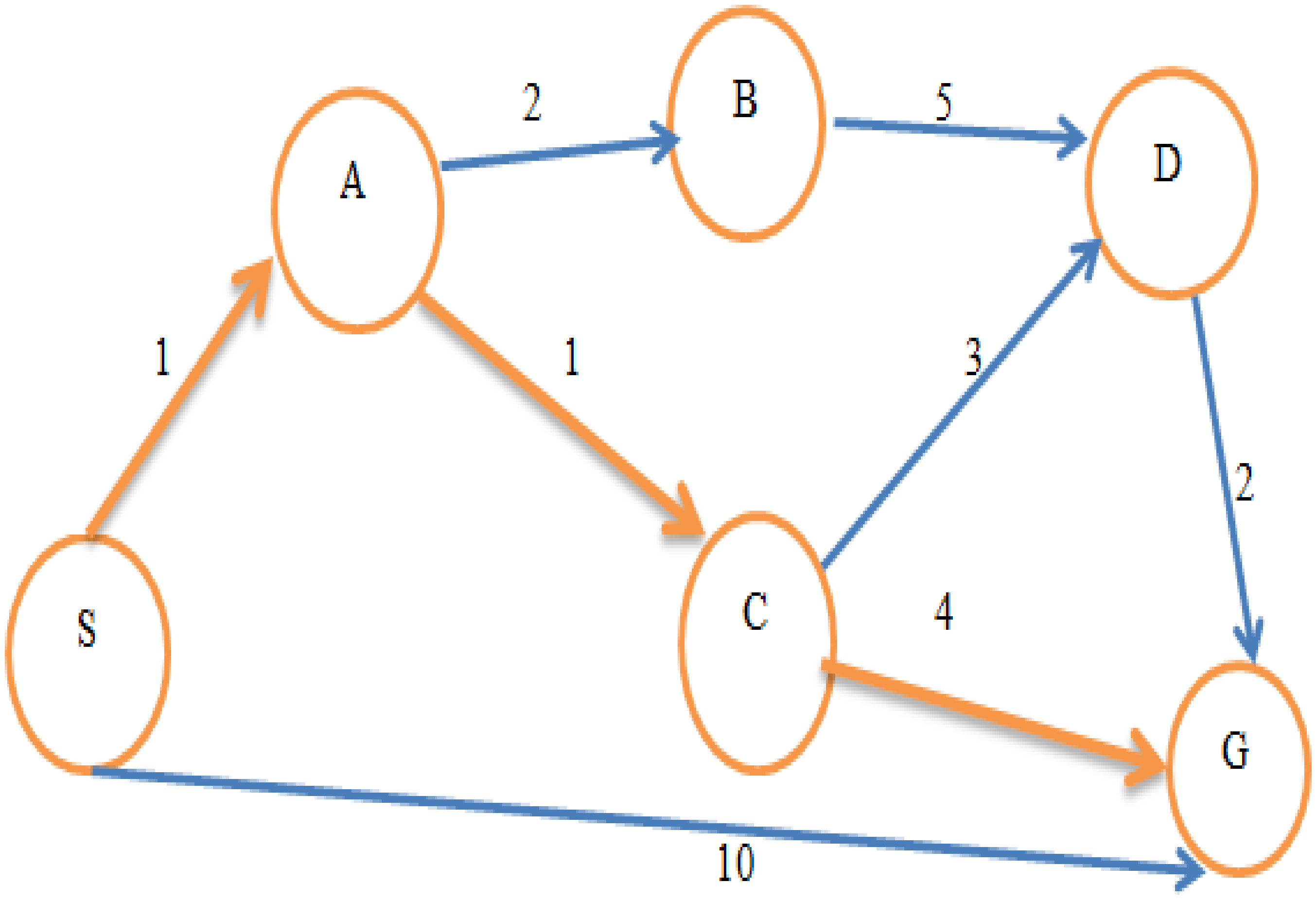
The A* formula

H- is the heuristic path cost between the current node to the destination

H- is not the actual cost but is the assumption cost from the node to the destination.

A good heuristic system must never over estimate the cost but can be allowed to underestimate the cost.

A* algorithm example



A* algorithm example

State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

1) S→A→F(n) = g(n) + h(n) = 1+3 = 4

S→G =F(n)= 10 + 0 = 10 (hold)

2) S→A→B = F(n) = 3+4 = 7 (hold)

S→A→B = F(n) = 2+2 = 4

3) S→A→C→D = F(n) = 5+6 = 11

S→A → C→G = F(n) = 1+1+4 = 6