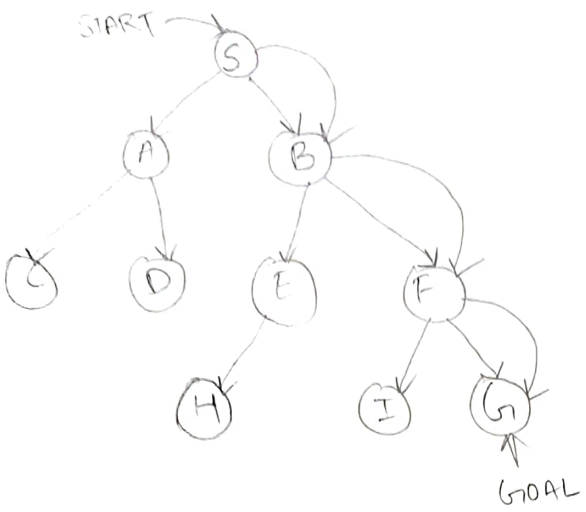
BEST FIRST SEARCH



NODE	h(n)
S	13
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
G	0

STEPS

Initialization → open = [S], close = [ ]

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]

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

→ reached goal state

PATH ⇒ S → B → F → G

EXPERIMENT - 5

AIM

To develop best first search and A* algorithm for real world problems

PART-1 BEST FIRST SEARCH

ALGORITHM

1. Start
2. Place the starting node into the open list.
3. If the open list is empty, stop and return failure
4. Remove the node n from the open list which has the lowest value of h(n) and place it in closed list
5. Expand the node n and generate the successors of node n
6. Check each successor of node n and find whether any node is a goal node or not. If any successor node is a goal node then return success and terminate the search else proceed to step 4.
7. For each successor node, algorithm checks for evaluation function f(n) and then checks if node is been in both list then add it to open list.
8. Return to step 3.
9. Stop.

A* SEARCH

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Final State

$g(n)$ = depth of node , $h(n)$ = number of misplaced tiles

Initial state

2	8	3
1	6	4
7		5

$g=0$
$h=4$
$f=0+4=4$

2	8	3
1	6	4
7		5

$g=1$
$h=5$
$f=1+5=6$

2	8	3
1		4
7	6	5

$g=1$
$h=3$
$f=1+3=4$

2	8	3
1	6	4
7	5	

$g=1$
$h=5$
$f=1+5=6$

2	8	3
	1	4
7	6	5

$g=2$
$h=3$
$f=2+3=5$

2		3
1	8	4
7	6	5

$g=2$
$h=3$
$f=2+3=5$

2	8	3
1	4	
7	6	5

$g=2$
$h=4$
$f=2+4=6$

	8	3
2	1	4
7	6	5

$g=3$
$h=3$
$f=3+3=6$

2	8	3
1	1	4
	6	5

$g=3$
$h=4$
$f=3+4=7$

	2	3
1	8	4
7	6	5

$g=3$
$h=2$
$f=3+2=5$

2	3	
1	8	4
7	6	5

$g=3$
$h=4$
$f=3+4=7$

1	2	3
	8	4
7	6	5

$g=4$
$h=1$
$f=4+1=5$

Final state

1	2	3
8		4
7	6	5

$g=5$
$h=0$
$f=5+0=5$

1	2	3
7	8	4
	6	5

$g=5$
$h=2$
$f=5+2=7$

PART - 2 : A* SEARCH ALGORITHM

ALGORITHM

1. Start
2. Place the starting node in open list
3. Check if the open list is empty or not, if the list is empty then return failure and exit the program.
4. Select the node from open list which has the smallest value of evaluation function $f(n) = g(n) + h(n)$, if node n is goal node then return success and exit the program
5. Expand node n and generate all of its successors and push it into the closed list. For each successor 'n', check whether n is already in the open or closed list, if not then compute evaluation function for n and place in open list.
6. Else if node n is already in open list and closed list then it should be attached to the back pointers which reflects lowest $g(n)$ value.
7. Retums to step 3.
8. Stop

RESULT

Successfully implemented Best First Search And A* search algorithm.