Lab # 07 - Informed Search

Best First Searches

Best first search uses the concept of a **priority queue** and **heuristic search**. It is a search algorithm that works on a specific rule. The aim is to reach the **goal** from the **initial** state via the shortest path. If we consider searching as a form of **traversal in a graph**, an *uninformed search* algorithm would blindly traverse to the next node in a given manner without considering the cost associated with that step.

An **informed search**, like **Best first search**, on the other hand would use **an evaluation function** to decide which among the various available nodes is the most promising (or 'BEST') before traversing to that node.

To search the graph space, the BFS method uses two lists for tracking the traversal. An '**Open'** list which keeps track of the current 'immediate' nodes available for traversal and '**CLOSED'** list that keeps track of the nodes already traversed. In the python implementation OPEN list is alternatively named as **QUEUE** and CLOSED list as VISITED.

Greedy Best First Search and **A*** are the two most popular variants of Best First Searches.

There are various flavors of Best First Search algorithm with different **heuristic evaluation functions f(n)**. The only difference between **Greedy** BFS and **A*** BFS is in the evaluation function. Essentially, since A* is more optimal of the two approaches as it also takes into consideration the total distance travelled so far i.e. g(n).

f(n) = Evaluation Function, g(n) = Total cost from Initial to current state
h(n) = An estimate value from current state to the goal state

g(n)	Path Distance
h(n)	Estimate to Goal
f(n)	Combined Hueristics i.e. g(n) + h(n)

For Greedy BFS the evaluation function is: $f(n) = h(n)$
For A* the evaluation function is: $f(n) = g(n) + h(n)$.

Let's have a look at the graph below and try to implement both Greedy Best-First-Search and A* algorithms step by step using the two list, OPEN and CLOSED.

Greedy BFS with evaluation function f(n) = h(n)

Step 1 - Start by adding the start node (S) to the open list with the path distance as 0

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
S	10		

Repeat the next steps until the OPEN List is empty or the Goal node is moved to the CLOSED list

A* BFS with evaluation function f(n) = h(n) + g(n)

Step 1 - Start by adding the start node (S) to the open list with the path distance as 0

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
S	0	10	10		

Repeat the next steps until the OPEN List is empty or the Goal node is moved to the CLOSED list

Step 2 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
A	9	S	
B	7		
C	8		

Step 2 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
A	7	9	16	S	
B	2	7	9		
C	3	8	11		

Step 2 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
B	7	S	
C	8		
A	9		

Step 2 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
B	2	7	9	S	
C	3	8	11		
A	7	9	16		

Step 3 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
C	8	S	
A	9	B	S
D	8		
H	6		

Step 3 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
C	3	8	11	S	
A	7	9	16	B	S
D	6	8	14		
H	3	6	9		

Step 3 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
H	6	S	
C	8	B	S
D	8		
A	9		

Step 3 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
H	3	6	9	S	
C	3	8	11	B	S
D	6	8	14		
A	7	9	16		

Step 4 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
C	8	S	
D	8	B	S
A	9	H	B
F	6		
G	3		

Step 4 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
C	3	8	11	S	
D	6	8	14	B	S
A	7	9	16	H	B
F	6	6	12		
G	5	3	8		

Step 4 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
G	3	S	
F	6	B	S
C	8	H	B
D	8		
A	9		

Step 4 (b) - Re-order the list in ascending order of the combined hueristic value

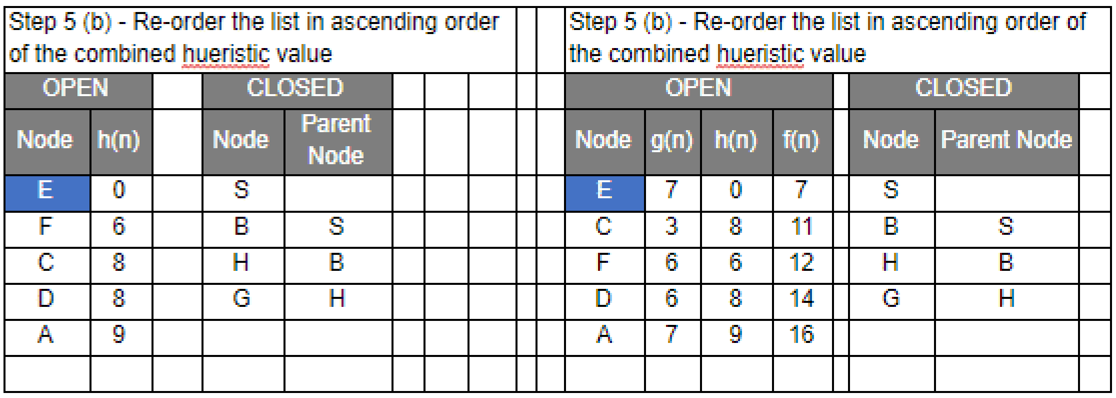
OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
G	5	3	8	S	
C	3	8	11	B	S
F	6	6	12	H	B
D	6	8	14		
A	7	9	16		

Step 5 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
F	6	S	
C	8	B	S
D	8	H	B
A	9	G	H
E	0		

Step 5 (a) - Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
C	3	8	11	S	
F	6	6	12	B	S
D	6	8	14	H	B
A	7	9	16	G	H
E	7	0	7		

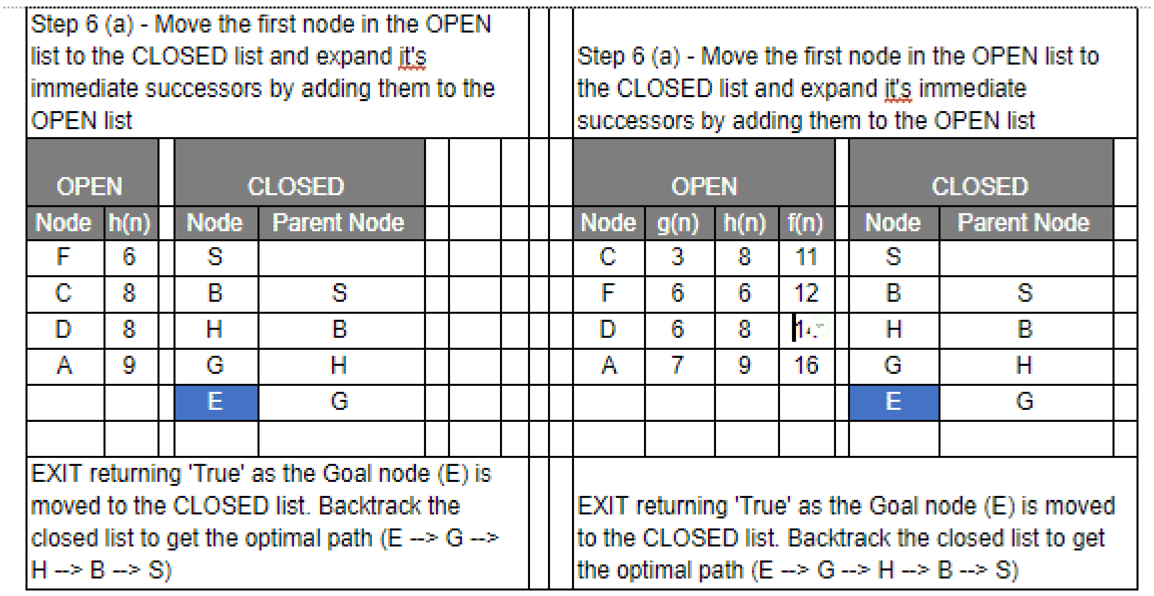


Step 5 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
E	0	S	
F	6	B	S
C	8	H	B
D	8	G	H
A	9		

Step 5 (b) - Re-order the list in ascending order of the combined hueristic value

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
E	7	0	7	S	
C	3	8	11	B	S
F	6	6	12	H	B
D	6	8	14	G	H
A	7	9	16		



Step 6 (a) - Move the first node in the OPEN list to the CLOSED list and expand it's immediate successors by adding them to the OPEN list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
F	6	S	
C	8	B	S
D	8	H	B
A	9	G	H
		E	G

EXIT returning 'True' as the Goal node (E) is moved to the CLOSED list. Backtrack the closed list to get the optimal path (E --> G --> H --> B --> S)

Step 6 (a) - Move the first node in the OPEN list to the CLOSED list and expand it's immediate successors by adding them to the OPEN list

OPEN				CLOSED	
Node	g(n)	h(n)	f(n)	Node	Parent Node
C	3	8	11	S	
F	6	6	12	B	S
D	6	8	14	H	B
A	7	9	16	G	H
				E	G

EXIT returning 'True' as the Goal node (E) is moved to the CLOSED list. Backtrack the closed list to get the optimal path (E --> G --> H --> B --> S)

Even though you would find that both Greedy BFS and A* algorithms find the path equally efficiently, number of steps, you may notice that the A* algorithm is able to come up with is a more optimal path than Greedy BFS. So in summary, both Greedy BFS and A* are Best first searches but Greedy BFS is neither complete, nor optimal whereas A* is both complete and optimal. However, A* uses more memory than Greedy BFS, but it guarantees that the path found is optimal.

Advantages and Disadvantages of Best First Search

Advantages:

1. Can switch between BFS and DFS, thus gaining the advantages of both.

2. More efficient when compared to DFS.

Disadvantages:

1. Chances of getting stuck in a loop are higher.

Try changing the graph and see how the algorithms perform on them.