National University of Computer and Emerging Sciences

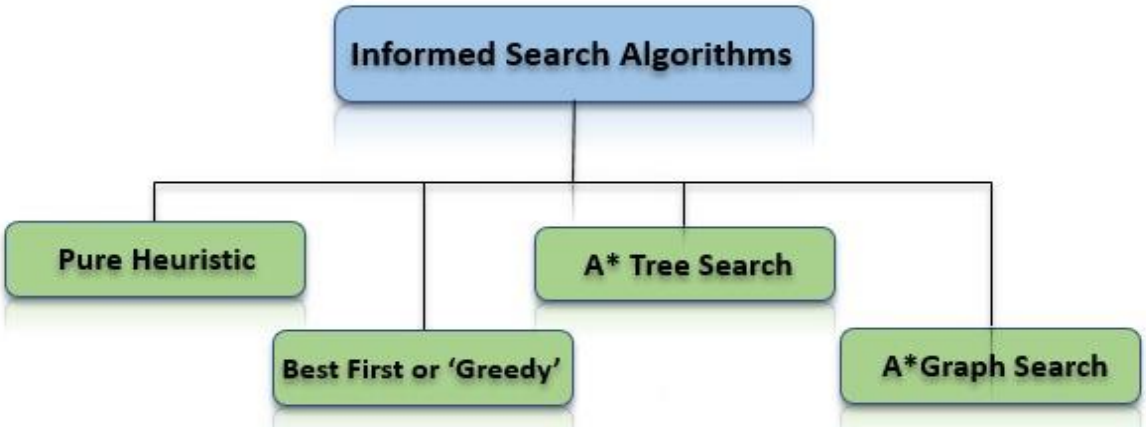
CL 461 Artificial Intelligence

Lab Manual 06

Problem Solving by Searching – Informed/Heuristic Based Search

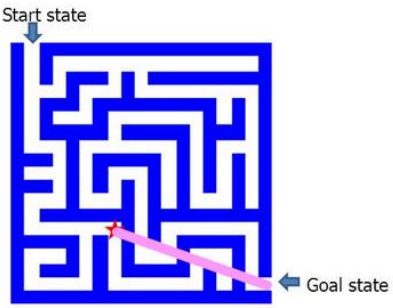
Heuristic (or informed) search algorithms:
– A solution cost estimation is used to guide the search.
– The optimal solution, or even a solution, are not guaranteed.

Some information about problem space (heuristic) is used to compute preference among the children for exploration and expansion.
To solve large problems with large number of possible states, problem-specific knowledge needs to be added to increase the efficiency of search algorithms.



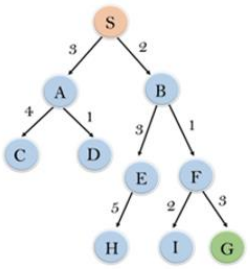
Heuristic function:
Heuristic function h(n) estimates the cost of reaching goal from node n.
They calculate the cost of optimal path between two states.

- Examples: Manhattan distance, Euclidean distance for path finding.



Pure Heuristic Search

It expands nodes in the order of their heuristic values. It creates two lists, a closed list for the already expanded nodes and an open list for the created but unexpanded nodes.



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

In each iteration, a node with a minimum heuristic value is expanded, all its child nodes are created and placed in the closed list. Then, the heuristic function is applied to the child nodes and they are placed in the open list according to their heuristic value. The shorter paths are saved and the longer ones are disposed.

Best-First Search

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.

The Best first search uses the concept of a Priority queue and heuristic search. 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.

Variants of Best First Search

- Greedy best-first search
- A* best-first search

Best first search algorithm:

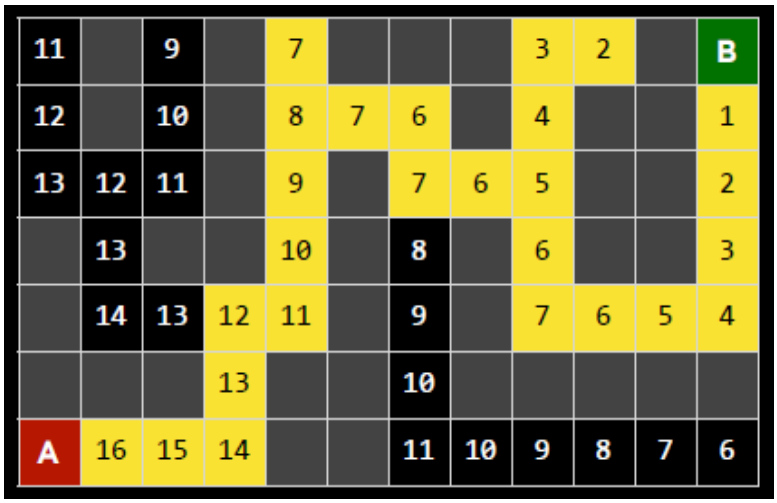
1. Create 2 empty lists: OPEN and CLOSED
2. Start from the initial node (say N) and put it in the 'ordered' OPEN list
3. Repeat the next steps until GOAL node is reached
    1. If OPEN list is empty, then EXIT the loop returning 'False'
    2. Select the first/top node (say N) in the OPEN list and move it to the CLOSED list. Also capture the information of the parent node
    3. If N is a GOAL node, then move the node to the Closed list and exit the loop returning 'True'. The solution can be found by backtracking the path
    4. If N is not the GOAL node, expand node N to generate the 'immediate' next nodes linked to node N and add all those to the OPEN list
    5. Reorder the nodes in the OPEN list in ascending order according to an evaluation function f(n)

Greedy Best first search algorithm:

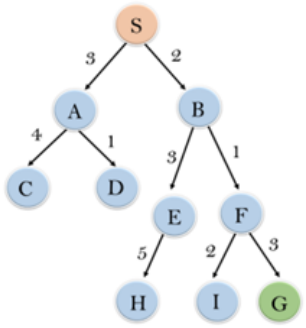
A search method of selecting the best local choice at each step in hopes of finding an optimal solution.

It is the combination of depth-first search and breadth-first search algorithms.

At each step, we choose the most promising node. In the greedy search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e. f(n)= h(n).



- Evaluation function *f(n) = h(n)*
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state
- Greedy best-first search expands the node that appears to be closest to goal
- It is implemented using priority queue.



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

Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

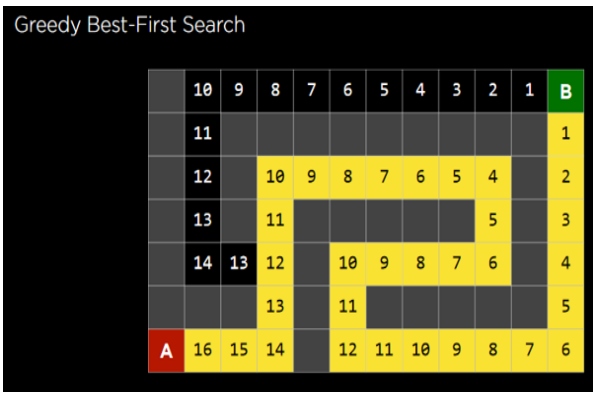
Iteration 1: Open [A], Closed [S, B]

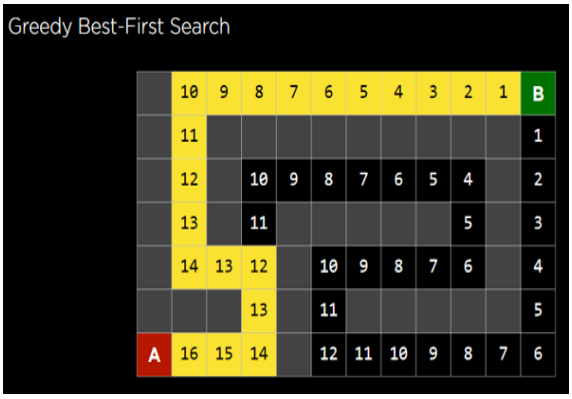
Iteration 2: Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: S----> B----->F----> G

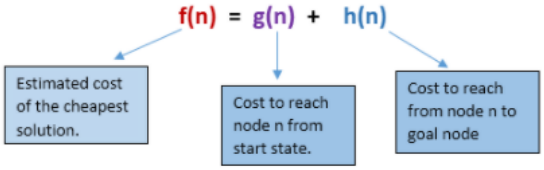
Disadvantage − It can get stuck in loops. It is not optimal.



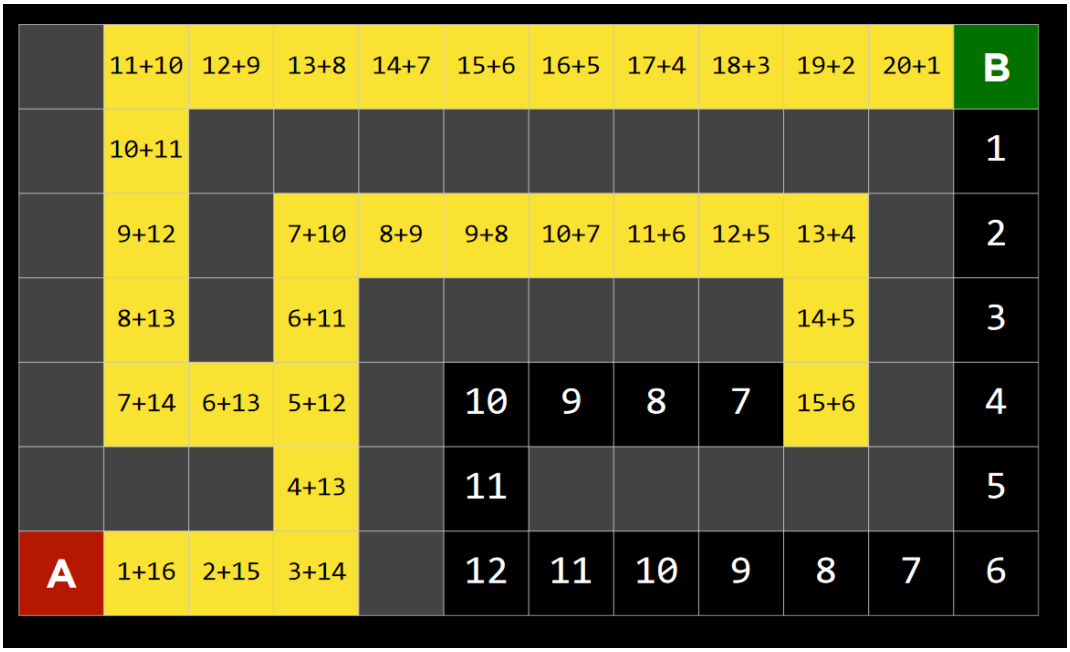


A* search

It is best-known form of Best First search. It avoids expanding paths that are already expensive, but expands most promising paths first.



•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

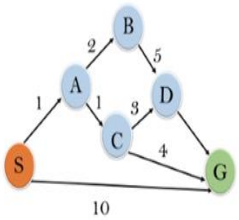


Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

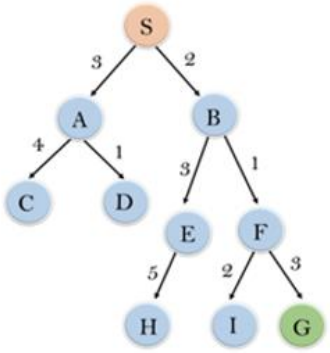


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

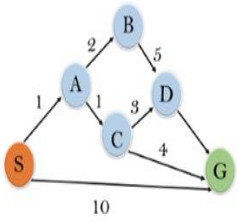
LAB EXERCISE

1. Implement the following tree using greedy algorithm having a destination node H.



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

2. Implement the following graph using A* search algorithm starting from Node S to Node D.



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

3. Solve the below maze using Greedy Best First and A* Best First search algorithm.

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

4. Write a program to solve the 8-puzzle problem using **Heuristics (h(n)) for A***