Subject : Greedy BFS

Date :

b -> Branching
m -> Max depth

Time and space complexity of Greedy BFS => $b^{8,m}$

~~BFS => b^m~~

Approximate

Heuristic (h) => Distance from the Goal State.

1	2	3	Goal State
4	5	6	h = 0
7	8		

B = Blank
Bup = Blank Up

~~Manhattan~~

Manhattan Distance is a distance Measure that is ~~calculated~~ calculated by taking the sum of distances between the x and y coordinates.

Goal ~~Given~~ State

1	2	3
4	5	6
7	8	-

Given ~~Goal~~ State

1	3	2
4	7	8
6	5	

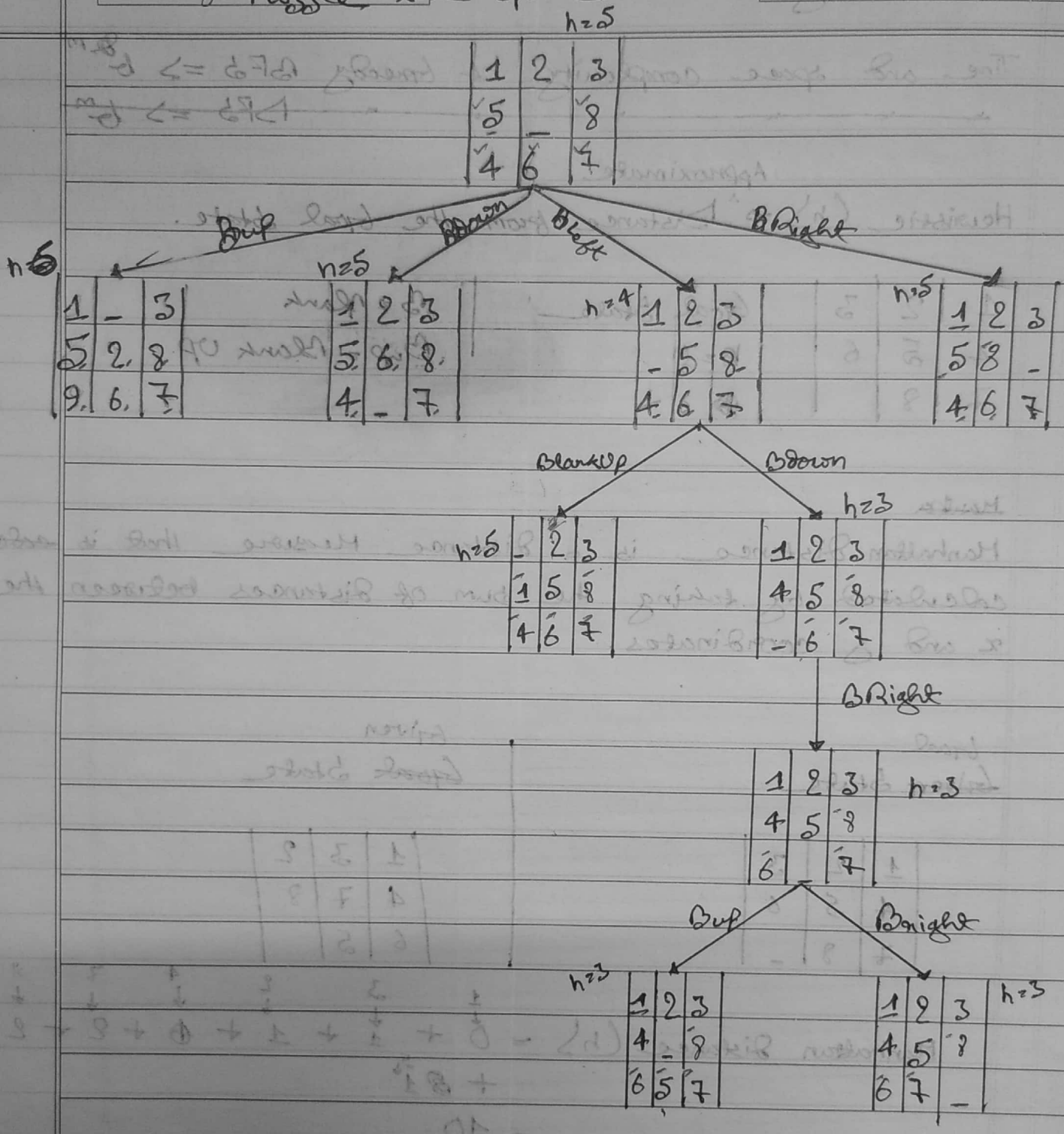
Manhattan Distance (h) = 0 + 1 + 1 + 4 + 2 + 2 + 3 + 1

= 10

A single problem can have multiple heuristic function. In such a case, we should choose the heuristic function that gives longer values.

Subject : 8 Puzzle state space Tree

Date :



Subject :

Date :

Say we have two heuristic functions named h_1 and h_2. For all the nodes, if $h_1(n) \geq h_2(n)$, then we can say that h_1 dominates h_2.

$h_1(A) = 12$, $h_1(B) = 8$, $h_1(C) = 20$, $h_1(D) = 9$

$h_2(A) = 7$, $h_2(B) = 9$, $h_2(C) = 17$, $h_2(D) = 8$

If we have two heuristic functions where for some of the nodes $h_1(n) \geq h_2(n)$ and ~~$h_2(n) \geq h_1(n)$~~ and for other nodes $h_2(n) \geq h_1(n)$ then we choose the heuristic function that happens to be True for most of the nodes. Otherwise, we can also take the best of both worlds i.e. $h_1(A) \geq h_2(A)$. So, for node A we will chose the heuristic function h_1. Similarly for B we will choose h_2 and so on. In short, we will choose the heuristic function that provides a larger value than the other ~~for~~ function for that very node.

Subject :

Date :

* For Tree and Heuristic Table, Look A* search notes.

A^{366}	GBFS → NOT optimal
A^{366} [C^{329} E^{253} B^{374}]	Complete ?
E^{253} [C^{329} B^{374} G^{193} F^{177}]	Tree → Incomplete
F^{178} [C^{329} B^{374} G^{193} I^{0}]	Graph → Finite → Complete
I^{0} [C^{329} B^{374} G^{193}]	Infinite → Incomplete
	Time and Space Complexity : → $O(b^m)$

When the goal node is expanded, the searching algorithm stops.

A → E → F → I

A → E → G → H → I	
Path cost = 140 + 99 + 211 = 450	path_cost = 140 + 80 + 97 + 101 = 418

Thus, optimal result is not achieved.

Optimal / sub-optimal path is defined using True path cost whereas Greedy BFS in its simulation does not even consider True path cost. It is entirely based on approximate path cost. So, the path output given by this algorithm will also be an approximate path.

All the search has two versions :

(i) Tree

(ii) Graph

Subject :

Date :

Tree version has the tendency to get stuck in a Loop.

In a Tree Search version, whether a parent has been visited or not is not kept Track of. But in a graph search version, it is kept track of.

Tree Greedy BFS is not ~~optimal~~ complete in any case. This infinite loop issue can be prevented using Graph Search version.

For finite tree, Graph GBFS is Complete.
" infinite " Incomplete.

Based upon the quality of the heuristic, the actual implementation time will come down lower if it is a good heuristic Function. So, if we use a decent quality heuristic that gives a good approximation, in practical implementation the time complexity decreases significantly compared to uninformed search.

A* search is the better version of GBFS.