**Question 1: Search Algorithms for the 15-Puzzle**

**(a)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **start10** | **start20** | **start27** | **start35** | **start43** |
| **UCS** | 2565 | Mem | Mem | Mem | Mem |
| **IDS** | 2407 | 5297410 | Time | Time | Time |
| **A\*** | 33 | 915 | 1873 | Mem | Mem |
| **IDA∗** | 29 | 952 | 2237 | 215612 | 2884650 |

**(b)**

As we can see from this table, UCS cost too much memory when the search depth rises, thus, its efficiency is the lowest and it has the highest space complexity.

As for IDS,it has the highest time complexity because when IDS search 27,35,43 depth, it cannot finish within 5 minutes. Although IDS can search 20 depth within 5297410 nodes,it still cost too much.Hence,IDS is the second lowest efficiency algorithm in these four algorithms.

In addition,A\* performs really good. However, it also needs too much memory when depth is getting huge.

IDA\* is the best efficiency algorithm among these algorithms. That’s because it is the only algorithm which can finish the start43 within 2 minutes and 2884650 nodes, which prove that IDA\* has the lowest time and space complexity

**Question 2: Deceptive Starting States**

**(a)**

Start49:

Heuristic value = 25

Start51:

Heuristic value = 43

**(b)**

Use idastar to search from start51, the number of nodes expanded is 551168

**(c)**

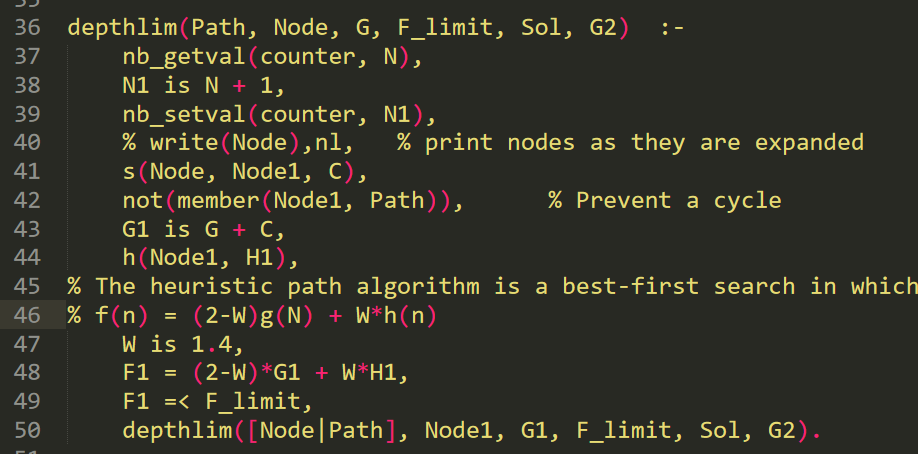
Based on the heuristic function,Although the true path is very similar,the Start(49) still cost too much than Start51.Hence,it expands so many more nodes than search from Start51.

**Question 3: Heuristic Path Search**

**(a)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Start49** | | **Start60** | | **Start64** | |
| **IDA\*** | 49 | 178880187 | 60 | 321252368 | 64 | 1209086782 |
| **1.2** | 51 | 988332 | 62 | 230861 | 66 | 431033 |
| **1.4** | 57 | 311704 | 82 | 3673 | 94 | 188917 |
| **Greedy** | 133 | 5237 | 166 | 1617 | 184 | 2174 |

**(b)**



I have changed the code for calculating F1,which according to formula: f(n) = (2-W)g(n) + W\*h(n)

**(d)**

When the W is increasing from 1.2 to 1.4,the length of path is growing and the number of nodes is decreasing.When w = 0,this algorithm is same as UCS algorithm,when w=1, this algorithm is same as A\* algorithm,when w=2, this algorithm is same as Greedy Search algorithm.Hence,the speed of solution is getting faster but the quality is getting worse with the value of W increases from 0 to 2.

**Question 4: Maze Search Heuristics**

Another admissible heuristic dominates the Straight-Line-Distance Heuristic-- Manhattan Distance Heuristic:

h(x,y,,)=|x-|+|y-|

**(b)**

(i):

No. The Straight-Line-Distance heuristic is not admissible.When the agent move diagonally from a gird center to another gird center,it will cost .Thus the heuristic value is ,but the actual cost to goal is 1,which violates the definition of heuristic admissible.Hence, The Straight-Line-Distance heuristic is not admissible.

(ii):No,it is not admissible too because it is over estimation too,the Manhattan Distance Heuristic only satisfy the move horizontally and vertically.When agent is moving diagonally,the Manhattan Distance Heuristic will consider it has the same cost as moving horizontally and vertically.

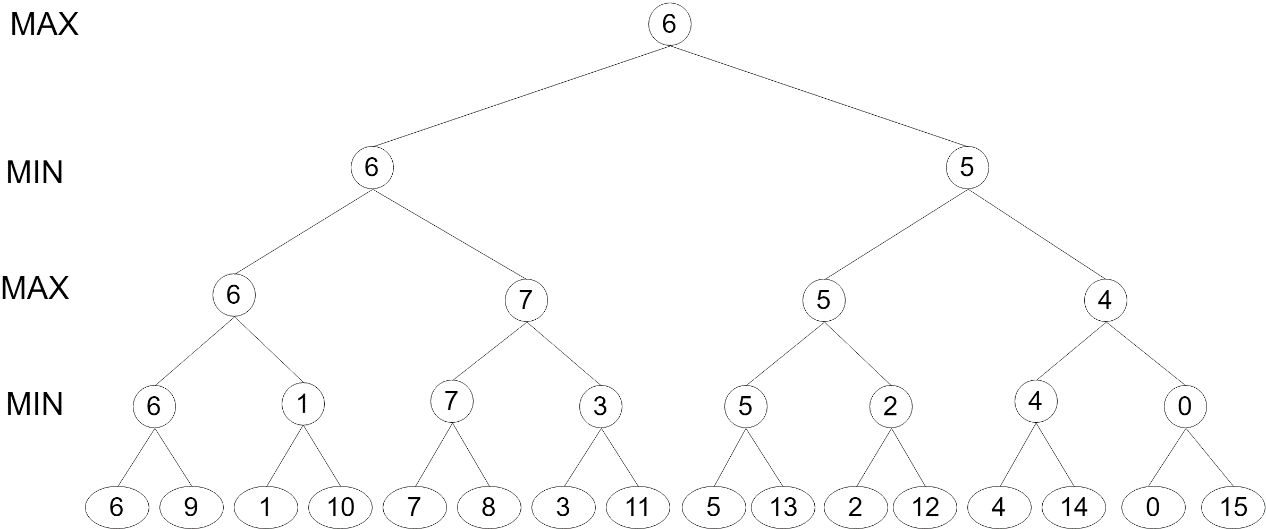
(iii): Chebyshev distance:

h(x,y,,)=max(|x-|,|y-|)

**Question 5: Game Trees and Pruning**

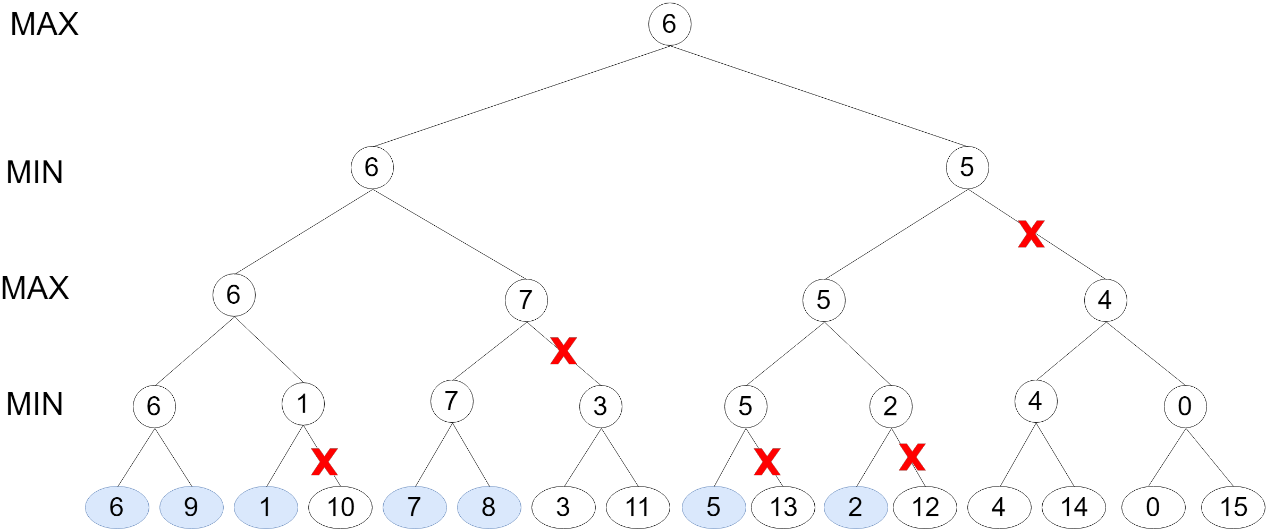
**(a)**

The game tree of depth 4 below is the one whose each node has exactly two children, with values from 0 to 15 for the leaves, in such a way that the alpha-beta algorithm prunes as many nodes as possible:



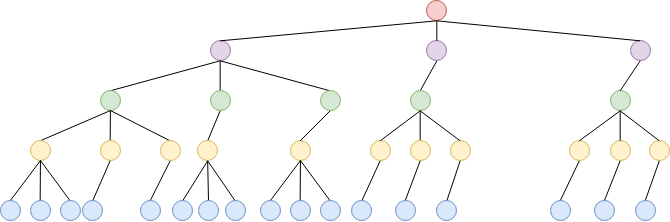
**(b)**

I trace through the alpha-beta search algorithm on my tree.The nodes of the blue background are evaluated.The red cross on the branch is the pruned



**(c)**

17 of 81 leaves(blue background) will be evaluated.



**(d)**

If the best move is always examined first (at every branch of the tree),the time complexity O(). b = branching factor, d = depth of the tree,To explain why time complexity is O().First,when the branching factor = b,then the maximum number of leaf node is O(b\*b\*b\*b\*…\*b) = O().Thus,if the best move is always examined first,Then,the time complexity is O(b\*1\*b\*1\*b…\*1\*b) for both odd and even depth.Hence,the time complexity for the best move first is O().