Q1

a)

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

b)

UCS : This algorithm has the highest space efficiency as it need take lots of memory. So it has the lowest efficiency.

IDS : This algorithm always expand the deepest unexpanded node, which could be failed in infinite-depth spaces ,or spaces with loops, so IDS has the low time efficiency especially between start27 to start43.

A* : This algorithm is similar to UCS, but it will avoid some unnecessary search, but sometime it will also take up big memory from start35 on.

IDA* : This algorithm is a low-memory variant of A*, so it has both the time and space efficiency.

Q2

a) H = 25, H = 43.

b) N = 551168

c)

This algorithm will change the max depth when it doesn't reach the goal state, so I think the reason is that the max depth of start49 is much larger than start51, that will expand more nodes.

Q3

a) c)

	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)

```
% Keep searching until goal is found, or F_limit is exceeded.
depthlim(Path, Node, G, F_limit, Sol, G2)  :-
    nb_getval(counter, N),
    N1 is N + 1,
    nb_setval(counter, N1),
    % write(Node),nl,   % print nodes as they are expanded
    s(Node, Node1, C),
    not(member(Node1, Path)),       % Prevent a cycle
    G1 is G + C,
    W is 1.2,
    h(Node1, H1),
    F1 is (2-W)*G1 + W*H1,
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

d) When w = 2 ,it’s Greedy algorithm which is the fastest algorithm, When w = 0, it's UCS, when w = 1 it's A*. And wen w ranging from 1.2 to 1.4, w and N have a proportional relation, it means that when w increases and N decrease. Hence faster speed with worse quality.

Q4

a) Manhattan Distance heuristic. $h_{MD}(x,y,x_G,y_G) = | x - x_G | + | y - y_G |$

b)

i) No, it will not be admissible, because according to this question, when it moves diagonally, a diagonal step is still considered to have the same 'cost' as one move, but actually the cost is sqrt(2), a little bit larger than 1. So it is not admissible.

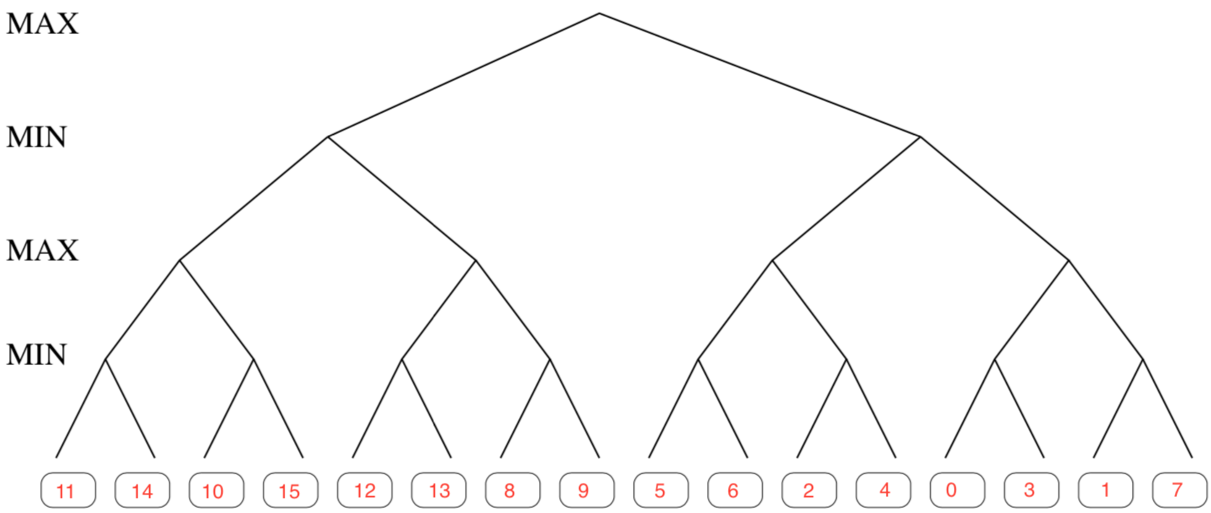
ii) No, heuristic from part(a) is the sum of all vertical and horizontal move, but as I mentioned above, this heuristic have same cost in vertical, horizontal and diagonal. So heuristic from part(a) is not admissible.

iii)

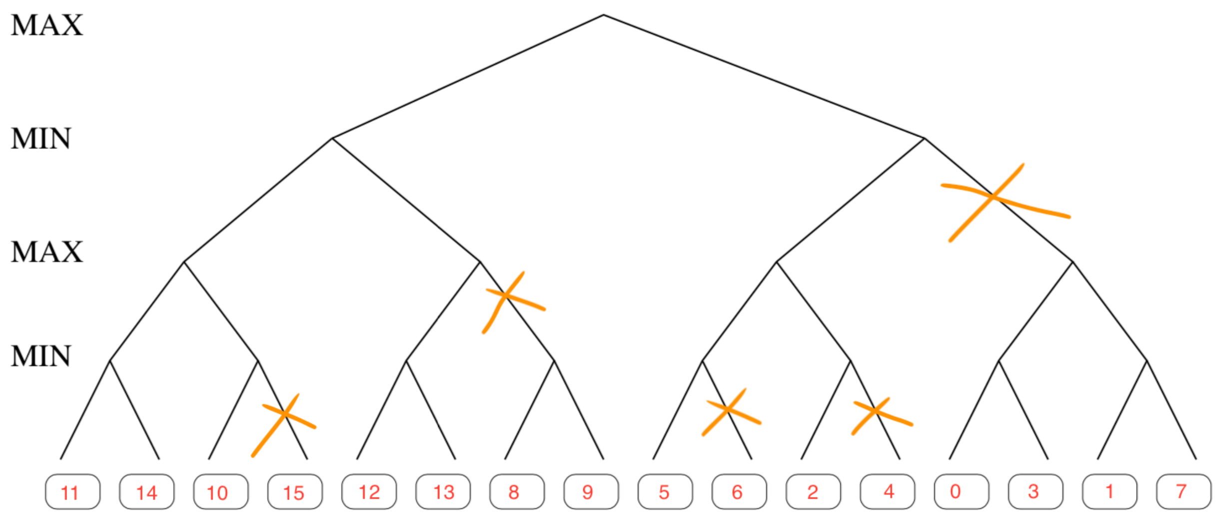
$$h(x,y,x_G,y_G) = \begin{cases} | x - x_G | & if\ xG - x \geq y - yG \\ | y - y_G | & if\ xG - x < y - yG \end{cases}$$

Q5

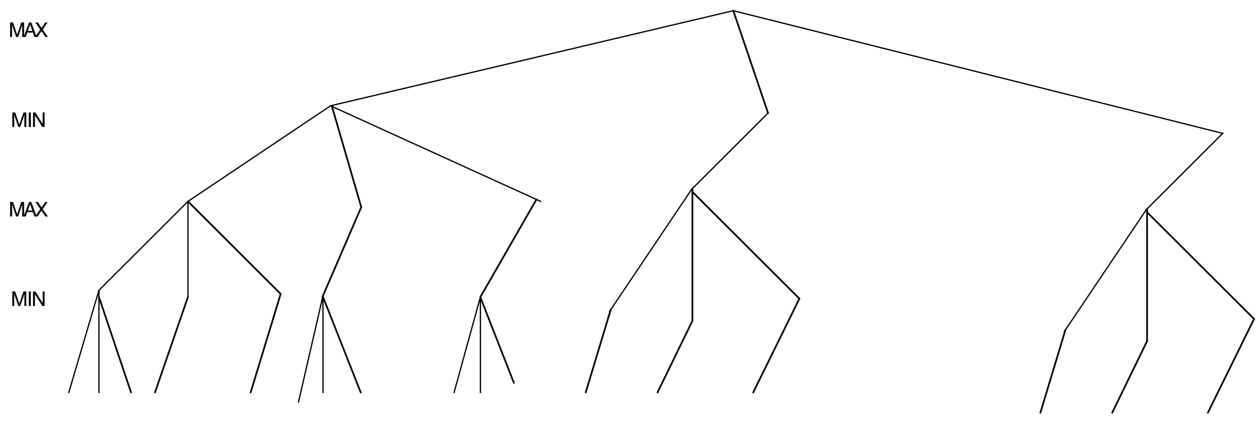
a) My game tree shows below:



b) 7 original leaves are evaluated and the others are pruned, which shows in graph below:



c) As we can see, there will have 17 of original leaves are evaluated, after trace through the alpha-beta search algorithm on this tree. The graph below shows the shape of the pruned tree.



d) The time complexity of alpha-beta search is O(b^(d/2)).

Where b = branching factor, d = depth of the tree.

If the best move is always examined first(at every branch of the tree), the number of leaf node positions evaluated is about *O*(*b**1**b**1*…**b*) for odd depth and *O*(*b**1**b**1*…*1) for even depth, or O(b^(d/2)).