COMP3411
Artificial Intelligence
Assignment 2

Heuristics and Search

Contents

Question 1

Part A

Start State	BFS		IDS		Greedy		A*	
start1	12	10978	12	25121	12	59182	12	30
start2	17	344890	17	349380	17	19	17	35
start3	18	641252	18	1209934	22	59196	18	133

Part B

The number of nodes expanded increases as path length increases for three of the four search strategies. These three search strategies also had the same path length for each start sequence, which was consistent over multiple executions of the program.

Greedy search was the only exception to this, with the number of nodes expanded being the most inefficient for start sequences 1 and 3. However quite surprisingly, start2 became the most efficient, which shows it's efficiency in the rare case but also its lack of consistency.

The most efficient strategy overall was A* search. It preserves the efficiency of Greedy Search but avoids expanding already-expensive paths, thus being admissible in the sense that it never overestimates the cost to reach the goal. A* search may not be the most efficient strategy for start sequence 2, but was the best overall.

The most inefficient strategy overall was actually IDS, with the highest number of nodes expanded for each start sequence. Unlike BFS, it can be optimal if step costs are identical, but it suffers due to the fact that early states are expanded multiple times. In this case with a path length greater than 10, the runtime quickly escalates and becomes a large problem.

Question 2

Part A

We are required to prove that Heuristic Path Search is optimal $\forall \omega : 0 \leq \omega \leq 1$, assuming $h(n)$ is admissible.

To prove that the Heuristic Path Search is optimal, we are to consider the objective function:

$$f_\omega(n) = (2 - \omega)g(n) + \omega h(n).$$

We aim to show that minimizing $f_\omega(n) = (2 - \omega)g(n) + \omega h(n)$ is the same as minimizing another objective function $f'_\omega(n) = g(n) + h'(n)$ for some function $h'(n)$, with the property that $\forall n : \ h'(n) \leq h(n)$.

We are given that $h(n)$ is admissible. That is, by definition, that the heuristic function $h(n)$ never overestimates the cost to reach the goal. Therefore, since $h'(n) \leq h(n)$, $h'(n)$ is also admissible.

Since $0 \leq \omega \leq 1$, it follows that $1 \leq 2 - \omega \leq 2$. Therefore we define $h(n)$ as follows:

$$h(n) = (2 - \omega)h'(n).$$

Rearranging the above equation yields

$$h'(n) = \frac{1}{2 - \omega} h(n).$$

Now substituting $h(n)$ into $f'_\omega(n)$ and rearranging, we get

$$\begin{aligned}
f'_\omega(n) &= g(n) + h'(n) \\
&= g(n) + \frac{1}{2-\omega} h(n) && \text{(substitution)} \\
&= (2-\omega)g(n) - (1-\omega)g(n) + \omega h(n) + \frac{(\omega-1)^2}{2-\omega} h(n) && (*2) \\
&= (2-\omega)g(n) + (\omega-1)g(n) + \omega h(n) + \frac{(\omega-1)^2}{2-\omega} h(n)) \\
&= (2-\omega)g(n) + \omega h(n) + (\omega-1)\left(g(n) + \frac{\omega-1}{2-\omega} h(n)\right)
\end{aligned}$$

Note that the coefficients of $h(n)$ in the line labelled (*2) are obtained from a rearrangement of ω:

$$\begin{aligned}
\omega &= \frac{\omega(2-\omega)}{2-\omega} \\
&= \frac{-\omega^2 + 2\omega}{2-\omega} \\
&= \frac{-\omega^2 + 2\omega - 1 + 1}{2-\omega} \\
&= \frac{-(\omega^2 - 2\omega + 1) + 1}{2-\omega} \\
&= \frac{-(\omega-1)^2 + 1}{2-\omega} \\
&= -\frac{(\omega-1)^2}{2-\omega} + \frac{1}{2-\omega} \\
\frac{1}{2-\omega} &= \omega + \frac{(\omega-1)^2}{2-\omega}.
\end{aligned}$$

Since $0 \leq \omega \leq 1$, it follows that $-1 \leq \omega - 1 \leq 0$, that is, $\omega - 1$ is either negative or zero. Therefore it suffices that

$$\begin{aligned} f'_\omega(n) &= (2-\omega)g(n) + \omega h(n) + (\omega - 1)\left(g(n) + \frac{\omega - 1}{2-\omega}h(n)\right) \\ &\leq (2-\omega)g(n) - \omega h(n) \\ &= f_\omega(n). \end{aligned}$$

Therefore given that $h(n)$ is admissible and $h'(n) \leq h(n)$, the function $h'(n)$ also does not overestimate the true cost to reach the goal from n, which preserves admissibility. Therefore $h(n)$ minimises the cost $g(n)$ added to an admissible heuristic, ensuring the shortest path is found without overestimating the true cost.

Thus the Heuristic Path Search is optimal, and we are finished.

<u>Part B</u>

	start4		start5		start6	
IDA* Search	45	545120	50	4178819	56	169367641
HPS, $\omega = 1.1$	47	523052	54	857155	58	13770561
HPS, $\omega = 1.2$	47	29761	56	64522	60	265672
HPS, $\omega = 1.3$	55	968	62	5781	68	9066
HPS, $\omega = 1.4$	65	9876	70	561430	80	37869

<u>Part C</u>

As the value of ω increases, the path length increases for all start sequences, with start 4 being the shortest and start 6 being the longest. This suggests that increasing values of ω become less and less optimal. Values of ω between 0 and 1 were proven to be optimal in part A.

However, the reverse is observed for the number of nodes expanded between values of ω from 1.0 to 1.3; a strong decline. This suggests that the time complexity improves as the value of ω increases. Then, for $\omega = 1.4$, the number of nodes expanded starts to increase again for all start sequences, which suggests time complexity could represent a parabolic relationship; at $\omega = 1.3$ the time complexity is cheapest, and either side of this it increases.

What was interesting about number of nodes expanded was that for $\omega = 1.4$, start5 had a larger number than start6. This was not true for all the other ω values. Also interesting for start5 was that $\omega = 1.4$ had a larger number than $\omega = 1.2$. Again this was not true for all the other ω values. This suggests something strange about the behaviour of the heuristic for some rare cases.

Question 3

Part A

Starting with the special case k = 0, compute $M(n,k)$ where 1 ≤ n ≤ 21.

$$M(1,0) = [+,-] = 2$$

$$M(2,0) = [+,\circ,-] = 3$$

$$M(3,0) = [+,\circ,\circ,-] = 4$$

$$M(4,0) = [+,+,-,-] = 4$$

$$M(5,0) = [+,+,-,\circ,-] = 5$$

$$M(6,0) = [+,+,\circ,-,-] = 5$$

$$M(7,0) = [+,+,\circ,-,\circ,-] = 6$$

$$M(8,0) = [+,+,\circ,\circ,-,-] = 6$$

$$M(9,0) = [+,+,+,-,-,-] = 6$$

$$M(10,0) = [+,+,+,-,-,\circ,-] = 7$$

$$M(11,0) = [+,+,+,-,\circ,-,-] = 7$$

$$M(12,0) = [+,+,+,\circ -,-,-] = 7$$

$$M(13,0) = [+,+,+,\circ,-,-,\circ,-] = 8$$

$$M(14,0) = [+,+,+,\circ,-,\circ,-,-] = 8$$

$$M(15,0) = [+,+,+,\circ,\circ,-,-,-] = 8$$

$$M(16,0) = [+,+,+,+,-,-,-,-] = 8$$

$$M(17,0) = [+,+,+,+,-,-,-,\circ,-] = 9$$

$$M(18,0) = [+,+,+,+,-,-,\circ,-,-] = 9$$

$$M(19,0) = [+,+,+,+,-,\circ,-,-,-] = 9$$

$$M(20,0) = [+,+,+,+,\circ,-,-,-,-] = 9$$

$$M(21,0) = [+,+,+,+,\circ,-,-,-,\circ,-] = 10$$

Part B

We are required to explain why the above pattern in $M(n,k)$ generalises to

$$M(n,0) = \lceil 2\sqrt{n} \rceil.$$

Instead of using recurrence relations, we are required to use the following identity,

$$\lceil 2\sqrt{n} \rceil = \begin{cases} 2s+1, if\ n = s^2 + k, 1 \le k \le s, \\ 2s+2, if\ n = s^2 + s + k, 1 \le k \le s, \\ 2s+2, if\ n = (s+1)^2. \end{cases}$$

We start with the third case in the above relation, namely,

$$\lceil 2\sqrt{n} \rceil = 2s + 2, if\ n = (s+1)^2.$$

In other words, since n $= (s+1)^2$, we have that n is a square number. In the domain 1 ≤ n ≤ 21, the numbers in the set {1, 4, 9, 16} satisfy this condition:

$$\lceil 2\sqrt{1} \rceil = 2 = 2s + 2$$
$$1 = (s+1)^2$$
$$\therefore s = 0$$

$$\lceil 2\sqrt{4} \rceil = 4 = 2s + 2$$
$$4 = (s+1)^2$$
$$\therefore s = 1$$

$$\lceil 2\sqrt{9} \rceil = 6 = 2s + 2$$
$$9 = (s+1)^2$$
$$\therefore s = 2$$

$$\lceil 2\sqrt{16} \rceil = 8 = 2s + 2$$
$$16 = (s+1)^2$$
$$\therefore s = 3$$

We then move to the first case in the above relation, namely,

$$\lceil 2\sqrt{n} \rceil = 2s + 1, if\ n = s^2 + k, 1 \leq k \leq s.$$

In other words, the result of $\lceil 2\sqrt{n} \rceil$ is an odd number. In the domain 1 ≤ n ≤ 21, the numbers in the set {2, 5, 6, 10, 11, 12, 17, 18, 19, 20} that have not been already mentioned satisfy this condition:

$$\lceil 2\sqrt{2} \rceil = 3 = 2s + 1$$
$$2 = s^2 + k$$
$$\therefore s = 1, k = 1$$

$$\lceil 2\sqrt{5} \rceil = 5 = 2s + 1$$
$$5 = s^2 + k$$
$$\therefore s = 2, k = 1$$

$$\lceil 2\sqrt{6} \rceil = 5 = 2s + 1$$
$$6 = s^2 + k$$
$$\therefore s = 2, k = 2$$

…

We finally move to the second case in the above relation, namely,

$$\lceil 2\sqrt{n} \rceil = 2s + 2, if\ n = s^2 + s + k, 1 \leq k \leq s.$$

In other words, the result of $\lceil 2\sqrt{n} \rceil$ is an even number. In the domain 1 ≤ n ≤ 21, the numbers in the set {3, 7, 8, 13, 14, 15, 21} that have not been already mentioned satisfy this condition:

$$\lceil 2\sqrt{3} \rceil = 4 = 2s + 2$$
$$3 = s^2 + s + k$$
$$\therefore s = 1, k = 1$$

$$\lceil 2\sqrt{7} \rceil = 6 = 2s + 2$$
$$7 = s^2 + s + k$$
$$\therefore s = 2, k = 1$$

$$\lceil 2\sqrt{8} \rceil = 6 = 2s + 2$$
$$8 = s^2 + s + k$$
$$\therefore s = 2, k = 2$$

…

Thus extrapolating patterns in the combination of all three cases satisfies the identity, and thus satisfies the general formula

$$M(n,0) = \lceil 2\sqrt{n} \rceil.$$

Part C

We are required to prove that, assuming the result from part B, if k ≥ 0 and

$$n \geq \frac{1}{2}k(k-1),$$

then

$$M(n,k) = \left\lceil 2\sqrt{n + \frac{1}{2}k(k+1)} \right\rceil - k.$$

Proof: we consider the path of the agent as $M(n,k)$ as part of a larger path. The expression $M(n,k)$ is the path from location 0 to goal n starting with velocity k.

Consider a path $M(n+b,0)$ consisting entirely of "+"s at the beginning up to location 0. In other words, the path starts at a location n + b which is less than 0, consisting of only "+" moves across b spaces up to location 0, and then performs $M(n,0)$ to arrive at goal n. From part B, we know that

$$M(n,0) = \lceil 2\sqrt{n} \rceil,$$

and by substitution we get

$$M(n+b,0) = \lceil 2\sqrt{n+b} \rceil.$$

Thus to prove the above formula, we must find a value of b such that the velocity at n is k. Given that $n \geq \frac{1}{2}k(k-1)$, we let

$$b = \frac{1}{2}k(k+1).$$

Then we have

$$M\left(n + \frac{1}{2}k(k+1), 0\right) = \left\lceil 2\sqrt{n + \frac{1}{2}k(k+1)} \right\rceil.$$

Then the least number of moves required to get to position n with velocity k is k. The least number of moves required here is the sequence of actions consisting only of "+"s.

Therefore we subtract k from the above formula to get

$$\begin{aligned} M\left(n+\frac{1}{2}k(k+1),0\right)-k &= \left\lceil 2\sqrt{n+\frac{1}{2}k(k+1)}\right\rceil - k \\ &= M(n,k), \end{aligned}$$

and we are finished.

Part D

We are tasked to derive a formula for $M(n,k)$ which satisfies k ≥ 0 and

$$n < \frac{1}{2}k(k-1).$$

In this case, it is likely that the goal n cannot be obtained in just one direction. In other words, a sequence of actions consisting only of "-"s cannot arrive at the goal n with velocity 0. Instead it will land at a location past the goal n with velocity 0, and we can continue adding another sequence of actions to move in the opposite direction back to the goal n.

Because of this, we split $M(n,k)$ into a sum of two parts. Let p be a point past n. Then we have

$$M(n,k) = M(p,k) + M(p-n,0),$$

where $M(p,k)$ represents the steps required to leave the current location starting at velocity n and stop at point p with velocity 0 which is past n, and $M(n-p,0)$ represents the steps required to leave point p at velocity 0 and travel in the opposite direction in the 1-dimensional plane to arrive at point n with velocity 0. Note that $M(p-n,k) = M(n-p,k)$.

By substituting appropriate values derived from parts B and C, we have that

$$\begin{aligned} M(n,k) &= M(p,k) + M(p-n,0) \\ &= \left\lceil 2\sqrt{p+\frac{1}{2}k(k+1)}\right\rceil - k + \left\lceil 2\sqrt{p-n}\right\rceil. \end{aligned}$$

Let $p = \frac{1}{2}k(k-1)$. Substituting this expression into $M(p,k)$ gives

$$\begin{aligned}
M\left(\frac{1}{2}k(k-1),k\right) &= \left\lceil 2\sqrt{\frac{1}{2}k(k-1)+\frac{1}{2}k(k+1)}\right\rceil - k \\
&= \left\lceil 2\sqrt{\frac{1}{2}k(k-1+k+1)}\right\rceil - k \\
&= \left\lceil 2\sqrt{k^2}\right\rceil - k \\
&= 2k-k \\
&= k
\end{aligned}$$

Then we have

$$\begin{aligned}
M(n,k) &= M\left(\frac{1}{2}k(k-1),k\right) + M\left(\frac{1}{2}k(k-1)-n,0\right) \\
&= k + \left\lceil 2\sqrt{\frac{1}{2}k(k-1)-n}\right\rceil \\
&= \left\lceil 2\sqrt{\frac{1}{2}k(k-1)-n}\right\rceil + k.
\end{aligned}$$

This is the formula to derive, and we are done.

<u>Part E</u>

We are required to provide an admissible heuristic in the form

$$h(r,c,u,v,r_G,c_G) = \max(M(..,..),M(..,..)),$$

where r = current row, c = current column, u = horizontal velocity, v = vertical velocity, r_G = goal row and r_G = goal column.

From parts C and D we derived formulas for $M(n,k)$. It would thus be possible to substitute these variations into the admissible heuristic. However, values of $n \geq \frac{1}{2}k(k-1)$ cannot be directly substituted into the formula in part D for values of $n < \frac{1}{2}k(k-1)$ without evaluating the square root of a negative number (since complex numbers do not exist in this 2-dimentional Graph Paper Grand Prix).

Thus we make the assumption that the max function used in the heuristic will favour the other expression if one expression cannot be evaluated.

We need to substitute the values given as arguments in the heuristic function into n and k. The parameter n does not represent the location of the goal, but rather the distance from the current position to the location of the goal. This is because $M(n, k)$ starts at location 0.

Therefore by using the distance formula, derived from Pythagoras' Theorem, we have that

$$\begin{aligned} n &= \sqrt{x^2 + y^2} \\ &= \sqrt{(r_G - r)^2 + (c_G - c)^2}, \end{aligned}$$

where x represents the horizontal distance between the current location and the goal, and y represents the vertical distance between the current location and the goal.

Then the parameter n represents the velocity at the current location, which, also using the distance formula above, is

$$k = \sqrt{u^2 + v^2}.$$

Thus we can substitute values of n and k with their associated expressions. In order to distinguish between the two variations of $M(n, k)$ we write the heuristic as

$$\begin{aligned} h(r, c, u, v, r_G, c_G) &= \max\left(M(n,k), k + M\left(\frac{1}{2}k(k-1) - n, 0\right)\right) \\ &= \max(M\left(\sqrt{(r_G - r)^2 + (c_G - c)^2}, \sqrt{u^2 + v^2}\right), \sqrt{u^2 + v^2} \\ &\quad + M\left(\frac{1}{2}\sqrt{u^2 + v^2}\left(\sqrt{u^2 + v^2} - 1\right) - \sqrt{(r_G - r)^2 + (c_G - c)^2}, 0\right), \end{aligned}$$

where

$$\begin{aligned} M(n,k) &= M\left(\sqrt{(r_G - r)^2 + (c_G - c)^2}, \sqrt{u^2 + v^2}\right) \\ &= \left\lceil 2\sqrt{\sqrt{(r_G - r)^2 + (c_G - c)^2} + \frac{1}{2}\sqrt{u^2 + v^2}\left(\sqrt{u^2 + v^2} + 1\right)} \right\rceil - \sqrt{u^2 + v^2}, \end{aligned}$$

which was specifically derived from part C, and

$$\begin{aligned} M\left(\frac{1}{2}k(k-1) - n, 0\right) &= M\left(\frac{1}{2}\sqrt{u^2 + v^2}\left(\sqrt{u^2 + v^2} - 1\right) - \sqrt{(r_G - r)^2 + (c_G - c)^2}, 0\right) \\ &= \left\lceil 2\sqrt{\frac{1}{2}\sqrt{u^2 + v^2}\left(\sqrt{u^2 + v^2} - 1\right) - \sqrt{(r_G - r)^2 + (c_G - c)^2}} \right\rceil, \end{aligned}$$

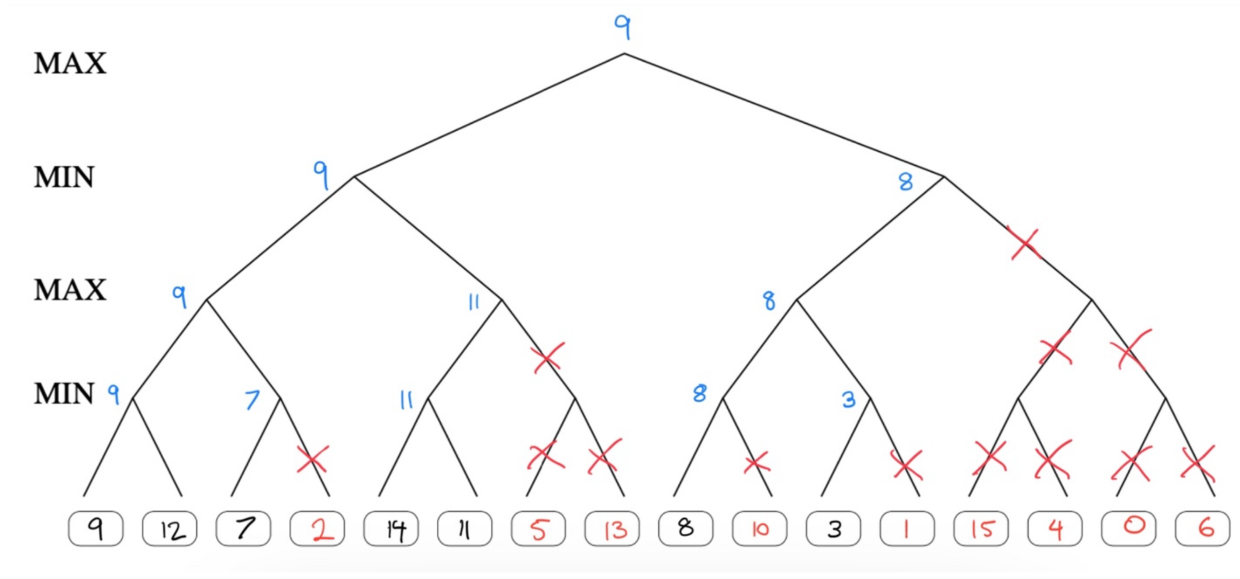
which was specifically derived from part D.

Question 4

Part A

The question asks us to fill in a depth 4 game tree, where each node has 2 children, with unique values 0 to 15 such that the algorithm prunes as many nodes as possible.

Many combinations of numbers exist to give the maximum number of nodes pruned. In the diagram below, the numbers in red do not matter, since they are pruned.



Part B

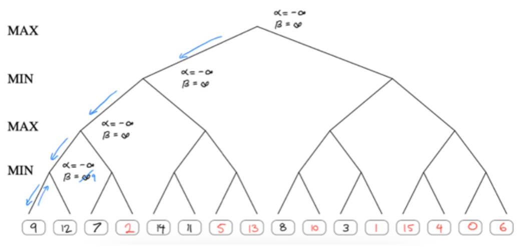
We are required to trace through the algorithm, showing the path and determination of alpha and beta values to explain which of the 16 leaf nodes are evaluated in the above tree.

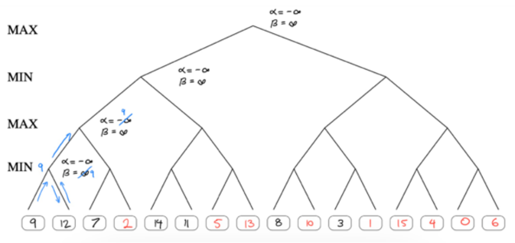
In total, **7 of the 16 leaves were evaluated.**

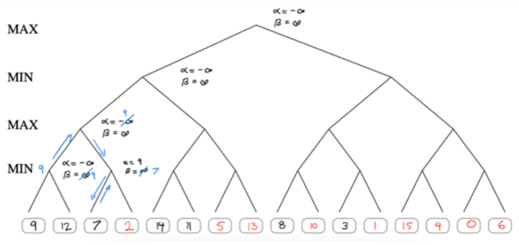
By default, the algorithm sets the alpha value to negative infinity and beta to positive infinity, since any value is larger than negative infinity for the maximum value alpha, and any value is smaller than infinity for the minimum value beta.

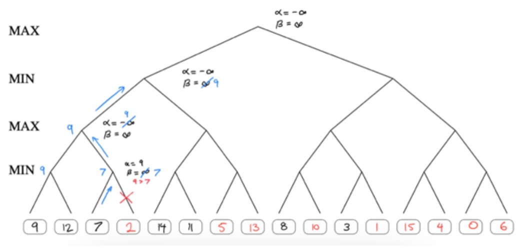
1. Starting with the leftmost value of 9, the algorithm stores 9 as its beta value in the nearest min node, since 9 is smaller than infinity.
2. Then it considers whether it needs to search the other branch. Because we do not have enough information, we search that node and compare the new value of 12 with 9 to see which is smaller. Since 9 is smaller, we backtrack up the tree and store 9 as the alpha value in this max node, since 9 is larger than negative infinity.
3. Going further down these levels, the algorithm encounters the number 7 and stores it as its beta value. Here, alpha = 9 and beta = 7.
4. Since 9 > 7, we can prune the right branch. It doesn't matter what this number is, because the min node before it will ensure a 7 or lower and the max node before that will always guarantee 9.
5. Now because the maximum of 7 and 9 is 9, the algorithm backtracks and stores 9 as the beta value in this min node, and then explores down the right hand side all the way to 14.

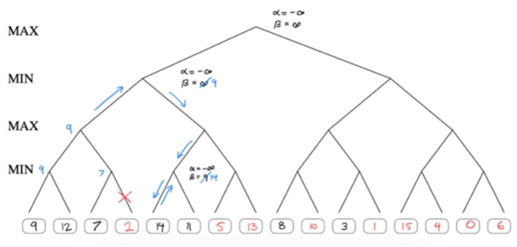
6. Since no alpha value is stored, we need to explore the right branch, which is 11. Backtracking once again to the max node, we know for sure that the min of 14 and 11 is 11, so alpha becomes 11 and beta becomes 9.
7. Since 11 > 9, we can prune the entire right branch, pruning 3 more branches. This is because this max node will guarantee 11 or higher, but the min node before it will always guarantee 9.
8. Now backtracking to the root with a value of 9 as the alpha, the algorithm explores the right branch and then all left branches until the leaf is found, which is 8. 8 becomes the beta, and 9 is the alpha still.
9. Since 9 > 8, we can prune this right branch and backtrack to the max node before exploring the right branch at this max node.
10. The next number encountered is 3, and whether or not the alpha value is 9 or 8, it doesn't matter because both are > 3, so we can prune the right branch.
11. Backtracking even further, we still keep our alpha value of 9, and the 8 we found earlier becomes the beta value of this min node. Since 9 > 8, we can actually prune the entire right side, therefore pruning 7 branches. Then the algorithm terminates at the root.
12. By ensuring this configuration of numbers, we have now successfully maximised the number of branches pruned, which is 13.

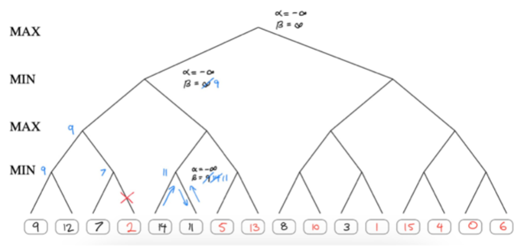
1.


2.


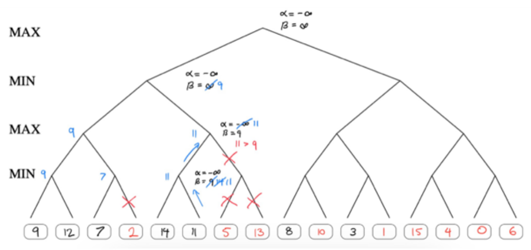
3.


4.


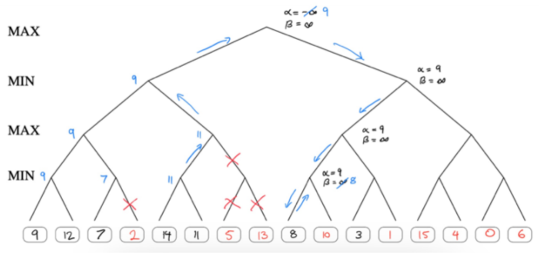
5.


6.


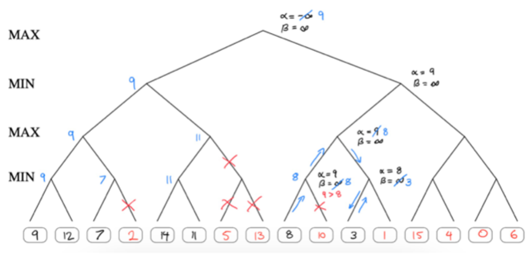
7.



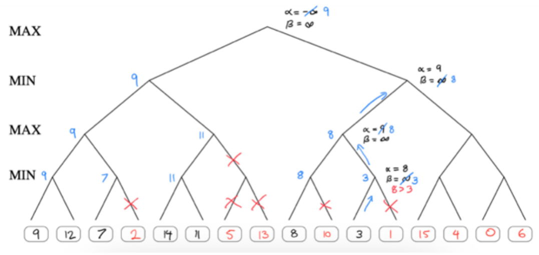
8.



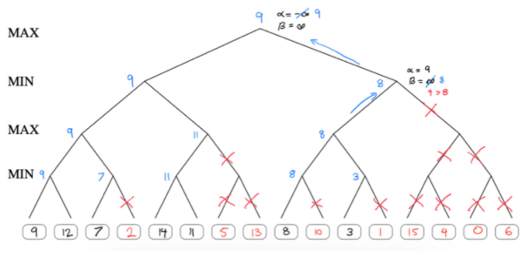
9.



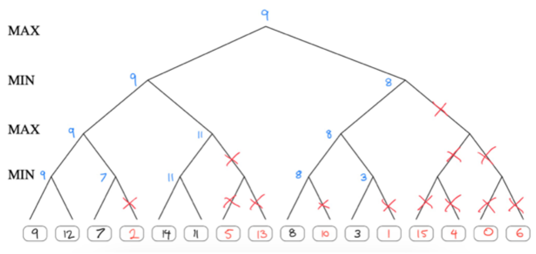
10.



11.



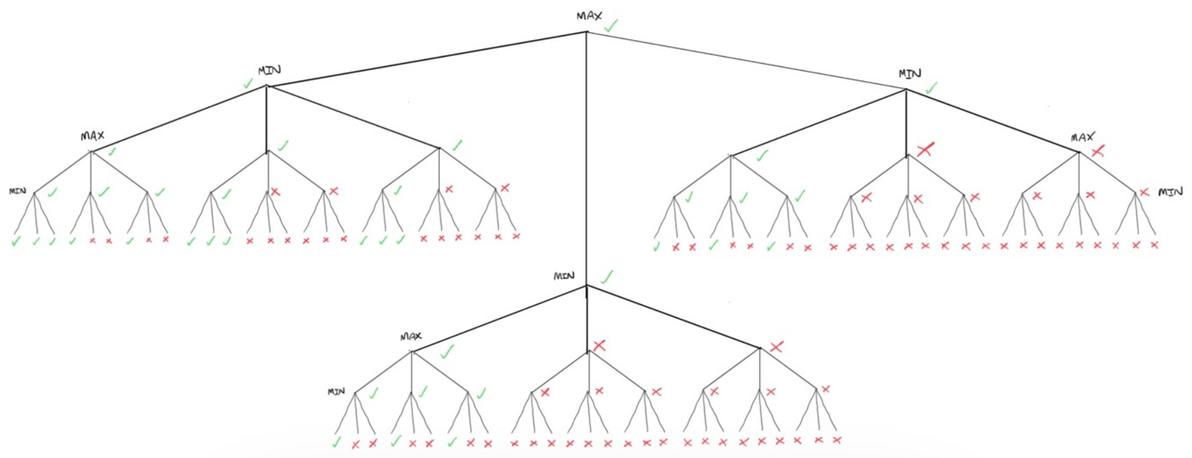
12.



Part C

We are to consider another depth 4 game tree, with 3 children at each node. We are required to draw the shape of the pruned tree such that the algorithm prunes as many modes as possible, and evaluate how many of the original 81 leaves will be evaluated.

In total, **17 of the 81 leaves will be evaluated**.



Part D

We are required to evaluate the time complexity of the alpha-beta pruning algorithm, provided the best move is always examined first at every branch of the tree.

If the best move is always examined first, the time complexity is $O\left(b^{\frac{m}{2}}\right)$.

For the explanation, we first consider the regular minimax algorithm. The minimax value is the best achievable payoff against optimal opponent. It assumes all nodes are evaluated with respect to a fixed player. Because it evaluates all nodes, its time complexity is $O(b^m)$.

Alpha-beta pruning is guaranteed to give the same result as minimax, but speeds up the computation substantially by storing alpha and beta values. Alpha is the best already explored option along path to the root for the maximiser, and beta is the same but for the minimiser.

In a two-player game, to prove that a bad move is bad we need only consider one good reply. But to prove that a good move is good, we need to consider all replies. Alpha-beta pruning prunes away bad moves to strengthen the player, thus being able to search twice as deep as plain minimax. Thus it's time complexity becomes $O\left(b^{\frac{m}{2}}\right)$, and we are done.