**Report findings for COMP3411**

**Question 1**

1. Draw up a table in this format: (length and number of paths expanded)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start State | BFS | | IDS | | Greedy | | A\* | |
| start1 | 12 | 10,978 | 12 | 25,121 | 12 | 59,182 | 12 | 30 |
| start2 | 17 | 344,890 | 17 | 349,380 | 17 | 19 | 17 | 35 |
| start3 | 18 | 641,252 | 18 | 1,209,934 | 22 | 59,196 | 18 | 133 |

1. **Briefly discuss the efficiency of these four search strategies, with regard to the number of nodes Expanded, and the length of the resulting path.**

**Question 2**

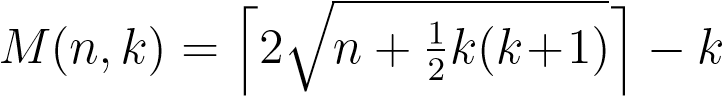
1. **Prove that Heuristic Path Search is optimal when 0 ≤ w ≤ 1, assuming h() is admissible. [Hint: show that minimizing fw(n) = (2 − w) g(n) + wh(n) is the same as minimizing fw′ (n) = g(n) + h′(n) for some function h′(n) with the property that h′(n) ≤ h(n) for all n.]**
2. **Draw up a table in this format (the top row has been filled in for you):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | start4 | | start5 | | start6 | |
| IDA\*Search | 45 | 545,120 | 50 | 4,178,819 | 56 | 169,367,641 |
| HPS, *w* = 1*.*1 | 47 | 523,052 | 54 | 857,155 | 58 | 13,770,561 |
| HPS, *w* = 1*.*2 | 47 | 29,761 | 56 | 64,522 | 60 | 265,672 |
| HPS, *w* = 1*.*3 | 55 | 968 | 62 | 5781 | 68 | 9066 |
| HPS, *w* = 1*.*4 | 65 | 9876 | 70 | 561,430 | 80 | 37,869 |

1. **Briefly discuss how the path Length and the number of Expanded nodes change as the value of w varies between 1.0 (equivalent to IDA\*) and 1.4.**

**Question 3**

1. **Starting with the special case k = 0, compute M(n, 0) for 1 ≤ n ≤ 21 by writing down the optimal sequence of actions for all n between 1 and 21. For example, if n = 7 then the optimal sequence is [+ + ◦ − ◦ −] so M(7,0) = 6. (When multiple solutions exist, you should pick the one which goes “fast early” i.e. with all the +’s at the beginning.)**
2. **Assume n ≥ 0. By extrapolating patterns in the sequences from part (a), explain why the general formula for M(n,0) is**
3. **Assuming the result from part (b), show that if *k* ≥ 0 and then**

****

**Hint: Consider the path of the agent as part of a larger path.**

1. **Derive a formula for *M*(*n,k*) in the case where *k* ≥ 0 and**
2. **Write down an admissible heuristic (that approximates the true numberof moves as closely as possible) for the original 2-dimensional GPGP game in terms of the function *M*() derived above. Hint: Consider the horizontal and vertical motion separately, keeping in mind that, unlike the 15-puzzle, the agent can move in the horizontal and vertical direction simultaneously. Your heuristic should be of this form: *h*(*r,c,u,v,rG,cG*) = max(*M*(*..,..*)*,M*(*..,..*))**

**Question 4**

1. **Consider a game tree of depth 4, where each internal node has exactly two children (shown below). Fill in the leaves of this game tree with all of the values from 0 to 15, in such a way that the alpha-beta algorithm prunes as many nodes as possible. Hint: make sure that, at each branch of the tree, all the leaves in the left subtree are preferable to all the leaves in the right subtree (for the player whose turn it is to move).**
2. **Trace through the alpha-beta search algorithm on your tree, clearly showing which of the original 16 leaves are evaluated.**
3. **Now consider another game tree of depth 4, but where each internal nodehas exactly three children. Assume that the leaves have been assigned in such a way that the alpha-beta algorithm prunes as many nodes as possible. Draw the shape of the pruned tree. How many of the original 81 leaves will be evaluated?**

**Hint: If you look closely at the pruned tree from part (b) you will see a pattern. Some nodes explore all of their children; other nodes explore only their leftmost child and prune the other children. The path down the extreme left side of the tree is called the line of best play or Principal Variation (PV). Nodes along this path are called PV-nodes. PV-nodes explore all of their children. If we follow a path starting from a PV-node but proceeding through non-PV nodes, we see an alternation between nodes which explore all of their children, and those which explore only one child. By reproducing this pattern for the tree in part (c), you should be able to draw the shape of the pruned tree (without actually assigning values to the leaves or tracing through the alpha-beta algorithm).**

1. **What is the time complexity of alpha-beta search, if the best move is always examined first (at every branch of the tree)? Explain why.**