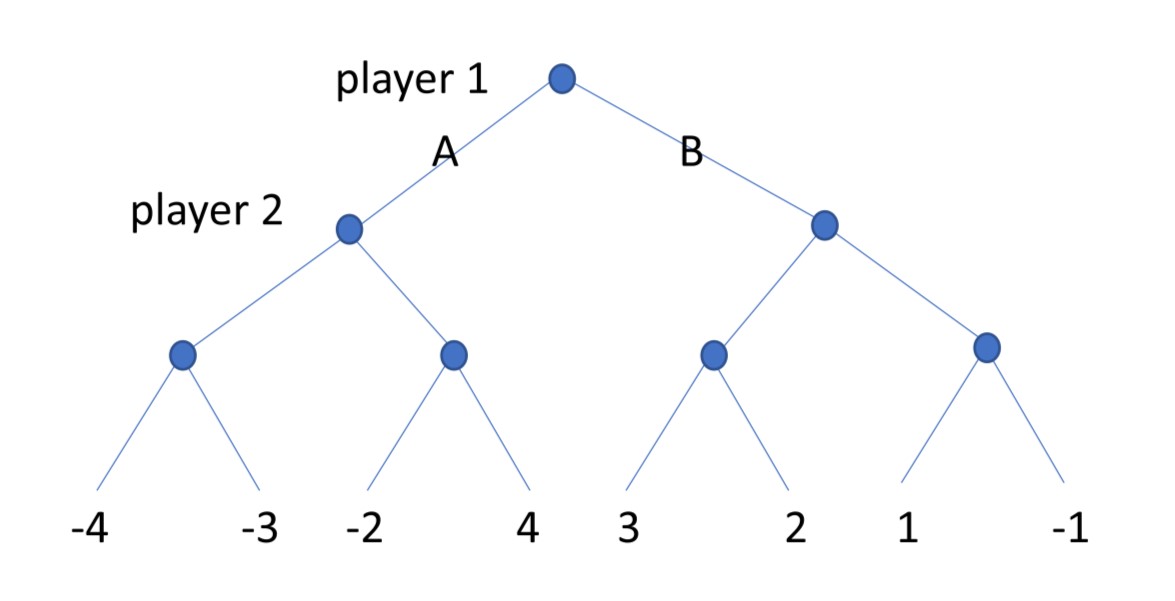
**CSCE 420 - Fall 2023 Homework 1 (HW1)**

**due: Thurs, Oct 5, 11:59pm - Late written homeworks will not receive credit.**

**Turn-in answers as a Word document (HW1.docx or .pdf) and commit/push it to your class github repo.**

|  |  |  |
| --- | --- | --- |
| When pushing the final version of your HW1, also run the command: | |  |
| **git tag "HW1" && git push origin "HW1"** |  |
| This will be used to record time of submission for late penalty when applicable. | | |

1. Given the simple game tree (binary, depth 3) below, label the nodes with up or down arrows, as discussed in the textbook.



Label the leaves with utility values for player 1 (who is at the root).

Compute the *minimax* values at the internal nodes (write the values next each node).

A paper with text and numbers

Description automatically generated

Should the player 1 take action A or B at the root?

B

What is the expected outcome (payoff at the end of the game)?

1

Which branches would be pruned by alpha-beta pruning? (circle them)

How could the leaves be relabeled to maximize the number of nodes pruned? (you can move the utilities around arbitrarily to other leaves, but you still have to use -4,-3,-2,1,+1,+2,+3,+4)

From left to right: 3, -3, 4, -4, 2, -2, 1, -1

How could the leaves be relabeled to eliminate pruning?

From left to right: 2, -2, 1, -1, 4, -4, 3, -3

1. In a simple binary game tree of depth 4 (each player gets 2 moves), suppose all the leaves have utility 0 except one winning state (+1000) and one loosing state (-1000).
   * Could the player at the root force a win?

No, the other player will have the last choice and choose the 0 over the 1000.

* + Does it matter where the 2 non-zero states are located in the tree? (e.g. adjacent or far apart)

No, the opponent will always have a choice to minimize 1000 or 0 or -1000 and will never choose the 1000.

* + If this question was changed to have a different depth, would it change the answers to the two questions above? If yes, how do the answers change? If no, explain why no change would happen.

No. From the bottom up, if it ends on a min node, the opponent will choose the 0 or

-1000 over the 1000. Similarly, if it ends on a max, the player would choose the 1000 and win but the opponent would never let the game reach that point since they would have control over the choice above and choose to go down the other path. The only way the root could force a win was if it was a 1 player game with one choice, which would be pretty boring.

1. Hiking Philosophers. Three philosophers, Alex (A), Bob (B), and Charlie (C), are going on a hike and need to decide the order in which they will hike. Alex and Charlie have PhDs, while Bob has a MS degree. Adjacent hikers in the sequence have to have different degrees. Finally, Charlie does not want to be last.



* 1. Show how to set this up as a Constraint Satisfaction Problem. (what needs to be defined?)

Each person (A, B, C) has a domain of (1, 2, 3) corresponding to their order in the line. C has their domain constricted to (1, 2) since they do not want to be last.

The constraints are:

* people can not be assigned the same number.
* A and C can not be adjacent since they share the same degree.

* 1. Draw the Constraint Graph (label all nodes and edges)

A hand holding a pen

Description automatically generated

* 1. Trace how plain Backtracking (BT) (with no heuristics) would solve this problem, assuming values are processed in *alphanumeric* order. Identify instances where back-tracking happens.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Backtracking Explanation |
| 1 |  |  |  |
| 1 | 2 |  | C does not have any choices left |
| 1 | 3 | 2 | A and C can not be adjacent |
| 2 |  |  |  |
| 2 | 1 |  | C does not have any choices left |
| 2 | 3 | 1 | A and C can not be adjacent |
| 3 |  |  |  |
| 3 | 1 |  |  |
| 3 | 1 | 2 | A and C can not be adjacent |
| 3 | 2 | 1 |  |

* 1. Trace how BT would solve this problem using the MRV heuristic.

|  |  |  |
| --- | --- | --- |
| A | B | C |
|  |  | 1 |
| 3 |  | 1 |
| 3 | 2 | 1 |

2