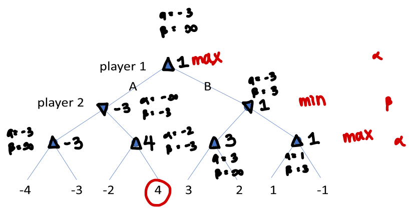
CSCE 420 - Fall 2025
Homework 2 (HW2)

due: Sat, Oct 4, 11:59 pm

Turn-in your answers as a Word document (HW2.docx or .pdf), and commit/push it to your class github repo in the HW2 folder.

1. Consider the simple game tree below (binary, depth 3).



1a. Label the nodes with up or down arrows, as discussed in the textbook .
see above diagram

1b. Compute the *minimax* values at the internal nodes (write the values next each node). see above diagram

1c. Which action, A or B, is optimal for player 1 to take? action B to maximize

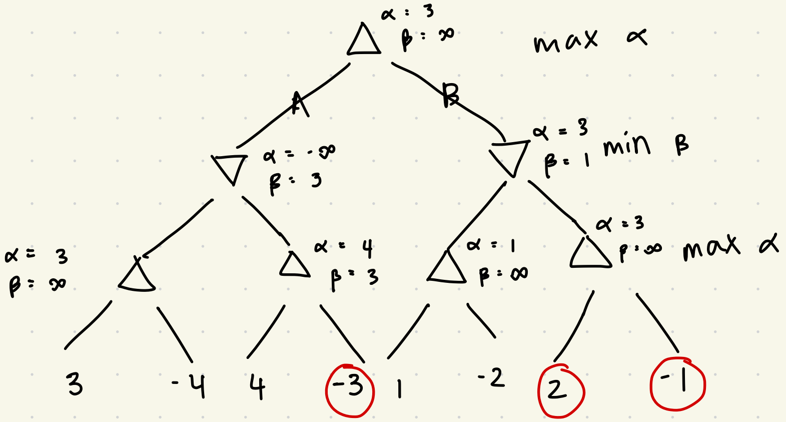
1d. What is the expected outcome (payoff at the end of the game)?
player 1 will end with a score of 1

1e. Which branches would be pruned by alpha-beta pruning? (circle them)
circled in red

1f. 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 the same values: -4,-3,-2,-1,+1,+2,+3,+4)
see below

1g. How could the leaves be relabeled to eliminate pruning?
see below

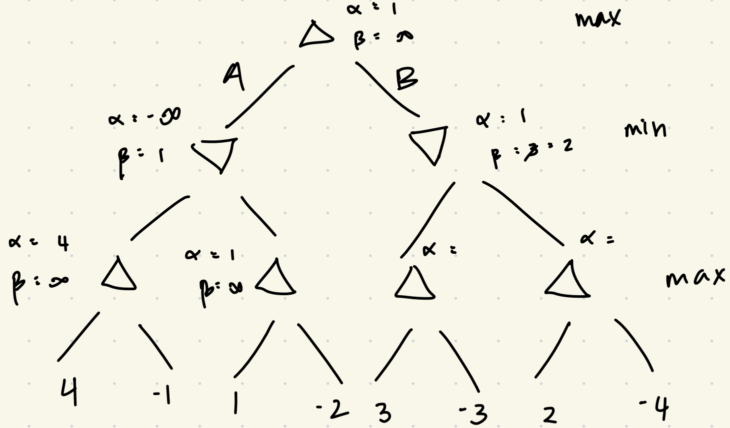
1f.



assuming we do left → right traversal the most nodes we can prune is 3. My example configuration prunes 3 nodes with prunes circled in red. We essentially organize the nodes in each subtree such that the right subtree always has a higher max than the left, therefore ensuring a prune.

In my example we can prune -3 after seeing the leaf w/ value $4 > \beta = 3$. On the B subtree, we have our carried over $\alpha = 3$ root and an updated $\beta = 1$ coming from the max of the B left subtree. Therefore we can prune the rest of B because we would never travel down this path since there is already a better min option of 3 on the A subtree.

1g.



In order to eliminate all pruning, we must never let the α = max of a left subtree be larger than the right subtree. In my example, when we go down the A left branch, we get an α max of 4 which carries into a β score of 4 into the A right branch. In this case 1 and -2 will never be greater than β, so the pruning condition is not met. A similar structure follows for the B subtree as nothing will be pruned since 3 is the first child searched and the largest number in the B subtree, prompting the search to traverse all leaves as no α max will ever be larger than the updated β = 3.

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

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

see answers below

b) Draw the Constraint Graph (label all nodes and edges)

"

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

"

d) <u>Trace</u> how BT would solve this problem using the <u>MRV heuristic</u>.

"

2 a)

Notes : O_1, O_2, O_3 is the order 1st, 2nd, 3rd

degree (X) is the degree that the hiker holds

Definitions :

degree(A) = degree (C) = PhD

degree(B) = MS

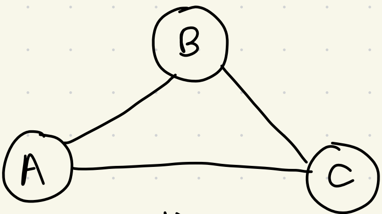
Constraints :

degree(O_1) ≠ degree (O_2) ≠ degree (O_3)

$O_3 \neq C$

2b) Constraint Graph

edge = different degrees



dom (O_1) = {A, B, C}

dom (O_2) = {A, B, C}

dom (O_3) = {A, B}

2c) Trace Backtracking

Assuming we process in alphanumeric order ...

1. O_1 = A
2. O_1 = A, O_2 = B
3. O_1 = A, O_2 = B, O_3 = C backtracks because C cannot be last
4. O_1 = A, O_2 = C backtracks because degree(A) = degree (C)
5. O_1 = B
6. O_1 = B, O_2 = A
7. O_1 = B, O_2 = A, O_3 = C backtracks because $\boxed{\text{degree (C) = degree(A) \& C cannot be last}}$
8. O_1 = B, O_2 = C
9. O_1 = B, O_2 = C, O_3 = A backtracks because degree (C) = degree (A)
10. O_1 = B, O_2 = C backtracks because no options left for O_3

continue on next page

11. $O_1 = B$ backtracks because no more options left for O_2

12. $O_1 = C$

13. $O_1 = C$, $O_2 = A$ backtracks, degree (C) = degree (A)

14. $O_1 = C$, $O_2 = B$

15. $O_1 = C$, $O_2 = B$, $O_3 = A$ solution reached ! constraint satisfied !

2d) Trace backtracking with MRV heuristic

MRV picks the variable w/ the smallest available domain

dom (O_1) = {A, B, C} dom (O_2) = {A, B, C}

dom (O_3) : {A, B}

1. Start w/ O_3

   $O_3 = A$

   dom (O_1) = {B, C} dom (O_2) = {B} removed A bc. degree(C) = degree(A)

2. Assign O_2

   $O_3 = A$, $O_2 = B$

   dom(O_1) = {C}

3. Assign O_1

   $O_3 = A$, $O_2 = B$, $O_1 = C$

   order = C, B, A

★ all constraints satisfied, solution reached