# Entry number: 2021CS50592 Name: KUSHAGRA GUPTA

# COL 333/671 Autumn 2023 Midterm

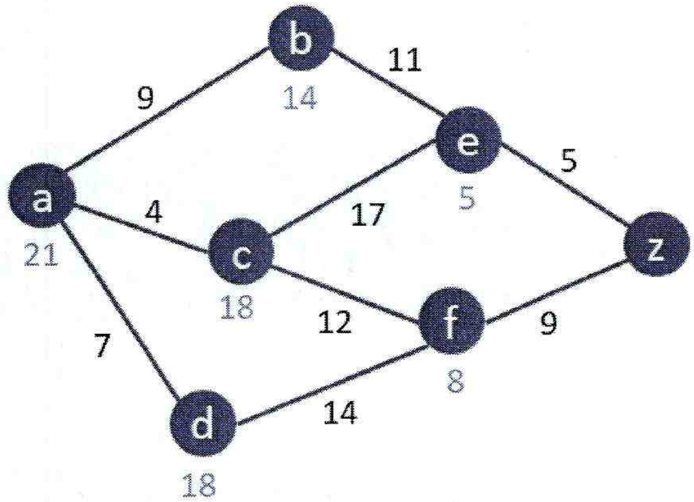
**Welcome to midterm exam. The exam is for 2 hours and for 125 pts. Please use only pens while answering questions. Do not use a pencil. Please do not write outside the margins – that part may not get graded.**

**If we find any form of collaboration with any other person during the course of exam, it will result in a straight zero on the exam – no exceptions.**

**Before starting the exam, close your eyes and take three deep breaths. Your performance in the exam is not an accurate reflection of your understanding of the material. Nevertheless, if you are relaxed, you will likely perform better.**

| Question Number | Maximum Marks | Marks Obtained |
|---|---|---|
| 1 | 18 | |
| 2 | 14 | |
| 3 | 13 | |
| 4 | 18 | |
| 5 | 17 | |
| 6 | 11 | |
| 7 | 18 | |
| 8 | 16 | |

1. [18 points] Execute search algorithms through this undirected graph. Step costs are given next to each arc, and heuristic values next to each node. The successors of each node are indicated by the arrows out of that node. a is the start node and z is the goal. Assume successors are returned in reverse lexicographic order. In case of any ties, use lexicographic ordering for tie breaking.




For each search strategy below, indicate the order in which nodes are visited. Also, mention the path returned by each strategy.

(a) Greedy best-first search (no duplicate detection) a b e z

Path returned : a → b → c → z .

(b) Iterative-deepening DFS search (no duplicate detection)

a b e z

a → b → e → z

(c) A* (full duplicate detection)

a b c z   a → b → c → z .

(d) Depth-first branch and bound (partial duplicate detection: only parent not re-expanded). Let branch policy be $g(n)+h(n)$. Let lower bound of cost through a node $n$ be $g(n)+h(n)$.

a then c

first part. a → c → f → z ~~a b c z~~ ~~a → b → e → z.~~   a c f z   25

then :

(e) Uniform cost search (no dup. detection). For this part assume that edge a-c is directed a→c.

a b c z

a → b → c → z

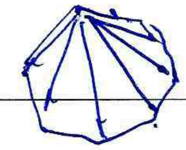
for bidirectional a → c. we will have path returned a → c → f → z

2. [14 points] We first list a few advantages and disadvantages of search algorithms that compare any two algorithms X and Y. For each pair of algorithms in a row, suggest *all* comparisons that are valid. Assume edge costs are positive but can be different for different edges. Also, generally assume that d (depth at which a goal is found) << m (max depth of search tree).

I. Theoretically, X takes less asymptotic space than Y.
II. Theoretically, X takes more asymptotic space than Y.
III. For large problems, X is generally computationally more efficient (in finding a good solution) than Y, in practice.
IV. For large problems, X is generally computationally less efficient (in finding a good solution) than Y, in practice.
V. X is (asymptotically) complete but Y is not.
VI. Y is (asymptotically) complete but X is not.
VII. X is (asymptotically) optimal but Y is not.
VIII. Y is (asymptotically) optimal but X is not.
IX. None of the above.

| Setting | X | Y | Comparisons |
|---|---|---|---|
| No duplicate detection | Uniform cost search | A* with zero heuristic | IX (both are same) |
| Finite search space | Depth first search with ancestor-based (only parent) duplicate detection | Breadth first search with no duplicate detection | I, IV, |
| Infinite search space with very few goal states | Iterative deepening depth first search | Depth first search | II, III, V, VII |
| Admissible heuristic *h*. No duplicate detection | Weighted A* with *w=5* | Best first search with *f(n) = 2(g(n)+h(n))* (A*) | I, III, VIII |
| Exactly one goal state. Large fan in, Small fan out. | Uniform cost search in forward direction | Uniform cost search in backward direction | I, III (I & III) |
| Exactly one goal state. Fan in and fan out are similar | Breadth first search with full duplicate detection | Bidirectional search with full duplicate detection | I, IV, |
| Ignore choices III, IV, V and VI | Enforced hill climbing | Greedy hill climbing with random restarts | II, |
| Ignore choices III, IV, V and VI | Random Sampling | Stochastic Local Beam Search | I, |

~~(DFS with parent duplicate)~~



3. [13 points] You are interested in solving the Hamiltonian cycle problem using genetic algorithms. I.e., given a graph G=(V,E), you wish to find a minimum cost cycle visiting all nodes in the graph exactly once. Let the nodes be numbered *1..n*. Let each pair of nodes be connected to each other, and the cost of following the edge from node $i$ to node $j$ ($e_{ij}$) be $c_{ij}$. All costs are positive.

(a) [3 points] Define a state (string) representation for any individual within a population. Describe the correspondence between the string and the equivalent cycle in the graph.

State representation :-
The string containing each node number exactly once starting from 1 : eg: 1 2 n n-1 5 4 3 ... is our string $s[1] = 1$.
The equivalent cycle is represented by the path from $s[i]$ to $s[i+1]$ ($\forall i \in (\cancel{0}1, n-1)$) & from $s[n]$ to $s[1]$. (where $s[1] = 1$
Whereas, the directed cycle in the graph can be used to form a string representation by breaking 1 & its predecessor
eg: 5 1 2 3 4 is 1 2 3 4 5 (not 5 1 2 3 4) $\therefore$ $\exists$ a bijection

(b) [3 points] Define a fitness function for a given individual. Note that it should be possible to take your fitness function and directly use it in the natural selection procedure discussed in class.

Fitness function :- The inverse cost corresponding to the cycle mapped by a string.
$\therefore$ if the string is 1 3 4 2 : Cost $= \dfrac{1}{C_{13} + C_{21} + C_{34} + C_{42}}$

This ensures that the fitness is more for lower cost individuals, & it works for natural selection as well. (% of mating is also more for min cost).

(c) [2 points] Suggest a mutation operator that makes a local change in a given state.

A mutation operator could be swapping any 2 elements of the string to create a newer individual, except swapping the first character $s[1] = 1$

Note: We cannot only replace one character as our string represents a permutation and ~~not~~ there are no self loop.

(d) [5 points] Suggest a crossing over operator that takes exactly two strings in your representation and creates a new state that has solution components from both parent strings. Explain, how your operator leads to a valid new solution, and how it has constituents of both parents.

operator for crossing over will take 2 permutations $\sigma_1, \sigma_2$ beginning with $s_1 = 1$ and construct a new permutation by:

Eg:

1 | 2 3 4
1 | 4 2 3

it will ~~be~~ be a consecutive application (composition of one permutation over the other).

$\therefore \sigma_1 \sigma_2 (s) = \sigma_1(\sigma_2(s))$

so. we will reorder 2 3 4 according to 4 2 3

like 2 at 4th posn → 3 4 2.
3 at 2nd posn
4 at 3rd posn

it will definitely be a permutation & has characteristics of both parents (allow us to look for other parts of ~~for~~ search space)

4. [18 points] The exam of AI is coming up, and there will be a total of six questions in the exam, Q1 on uninformed search, Q2 on informed search, Q3 on local search, Q4 on games, Q5 on CSPs and Q6 on logic. There are seven TAs: Aayush (A), Dhruvil (D), Kausik (K), Prachi (P), Raajita (R), Saptarshi (S) and Vipul (V). Each TA will work on exactly one question, though each question may have multiple TAs. However, TAs are students and have many constraints.

Dhruvil will not work with Prachi.
Raajita will only work on three search questions.
Saptarshi is really odd, and will only work on odd-numbered questions.
Vipul must work on a question that is numbered lower than Saptarshi's question.
Raajita must work on a question that is numbered lower than Dhruvil's question.
Aayush made the quiz for CSPs so he will only do the CSP question.
Prachi must work on a question that is numbered greater than Vipul's question.
Aayush and Vipul compete as PhD students, and are not willing to work together on any question.
Vipul will not work on logic.
Kausik will not work on games, CSP or logic.
Dhruvil will not work on CSPs.
Dhruvil must work on a question that is numbered lower than Kausik's question.

We will model this as a CSP, with TAs as variables and question numbers as values.

(a) [2 points] After applying unary constraints, what values remain for all variables?

Raajita $\in \{1, 2, 3\}$. Aayush $\in \{5\}$ Prachi $\in \{1,2,3,4,5,6\}$
Vipul $\in \{1, 2, 3, 4, 5\}$ Kausik $\in \{1, 2, 3\}$.
Dhruvil $\in \{1, 2, 3, 4, 6\}$ Saptarshi $\in \{1, 3, 5\}$

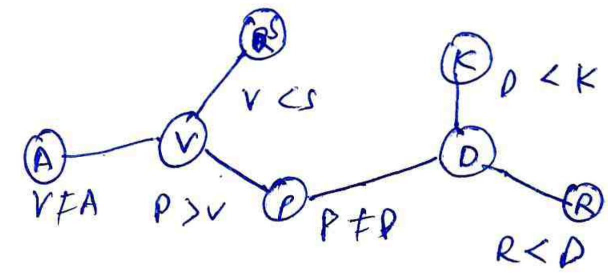
(b) [2 points] Which variable will be assigned a value first based on heuristics discussed in class?

MRV heuristic $\rightarrow$ as aayush has only one remaining value he will be assigned first.
(Degree Heuristic for tie)

(c) [4 points] Lets say, instead of assigning the variable you found in part (b), we assign Saptarshi first. Using the value ordering heuristic discussed in class, find the ordering of values we will try for Saptarshi? Assume all unary constraints have been applied first. Show your work.

Aayush = CSP. (unary constraint)
So Saptrishi has now $\{1, 3\}$ remaining in domain.
Binary constraints: $D \neq P$, $V < S$, $R < D$, $P > V$,
$V \neq A$, $D < K$.
Now we assign saptrishi the value which constrains the least neighbours :- {Our value heuristic}.
Since only V is a neighbour to S :- we assign S as 3
(as S = 1 causes no value for V)

(d) [2 points] Draw the constraint graph for this CSP




Realizing that it is tree-structured CSP, we decide not to run backtracking search anymore, and instead run the specialized algorithm for tree-structured CSPs. We will run this after applying all unary constraints. Let the topological sort gives us the following ordering of variables: Raajita, Dhruvil, Kausik, Prachi, Vipul, Aayush, Saptarshi.

(d) [6 points] Compute the values remaining for each variable after doing the domain pruning step.

Topological sort :- R D K P V A S

6 arcs :

We will prune the domain for ly tree arc consistency.

(n down 1 parent to child) First check which value of S causes no value for V.

↳ ~~V = 1~~ . S = 1 × ⇒ S = {5, 3}.

Now for A = 5 , ∃ V so not pruned.

Now for V = {1, 2, 3, 4, 5} ∃ 1, 2, 3, 4, 5, 6 ∈ P s.t P > V so not pruned

for P = {1, 2, 3, 4, 5, 6} ∃ {2, 3, 4, 5} so P not pruned.

for K = 1 ∃ no D. So 1 is pruned K ∈ {2, 3}.

for D , D > R ⇒ for D = 1 , no value of R so pruned.

Finally : R ∈ {1, 2, 3} , V ∈ {1, 2, 3, 4, 5, 6} , D ∈ {2, 3, 4, 6} , S ∈ {3, 5} , K ∈ {2, 3} , A ∈ {5} , P ∈ {1, 2, 3, 4, 5, 6}

(e) [2 points] Find the solution to the CSP. If multiple values are possible for any TA, choose the highest numbered.

By search we have solⁿ to CSP

A ∈ {5} D = {2}

S ∈ {5} R ∈ {1}

V ∈ {4} K = {3}

P ∈ {6}

R

5. [17 points] Consider the game of Two-Player Uno: For a set of colors C and a set of numbers N, there is a deck of cards D = C×N. This deck is mixed and then partitioned into two hands $H_1$, $H_2$ with an equal number of cards, and the remaining cards are kept in a pile P. Players alternately take turns to construct a stack S of cards. The first player to finish their hand of cards wins.

Let the card at the top of S is $(c_s, n_s)$. When it is a player i's turn to play, they do the following:

- check if there is a card $(c, n) \in H_i$ such that the color or number matches the top of the stack ($c = c_s$ OR $n = n_s$). If there is, play that card (add it to the top of the stack). If there are multiple cards in $H_i$ satisfying this condition, the player may choose any one.
- if not, draw a card from the pile P. If the drawn card's color or number matches the top of the stack, play the drawn card. Otherwise, continue to draw cards until the drawn card can be played, or P becomes empty. If P is empty, then they skip their turn.

If the stack is empty (i.e. the game has not yet started), the first player may play any card from their hand.

(a) [2 points] Classify the game based on whether it is a deterministic game or a game of chance, and perfect or imperfect information game.

The game is a game of chance because there ~~are~~ is some randomness/probability P with which the player gets the next card from Pile P.
However it is a perfect information game as we can mark "P" for all nodes (chance nodes) in expectiminimax tree

For our game, let C = {Red, Green} and N = {1, 2, 3}. We are player one and have been dealt the hand $H_1$ = {R1, G2}. We have no information about $H_2$. And player two has no information about $H_1$. We make the first move.

(b) [2 points] Which is a better starting move, R1 or G2? Justify your answer.

By ~~[illegible]~~ symmetry we can claim that both moves are equally good. R1 or G2.

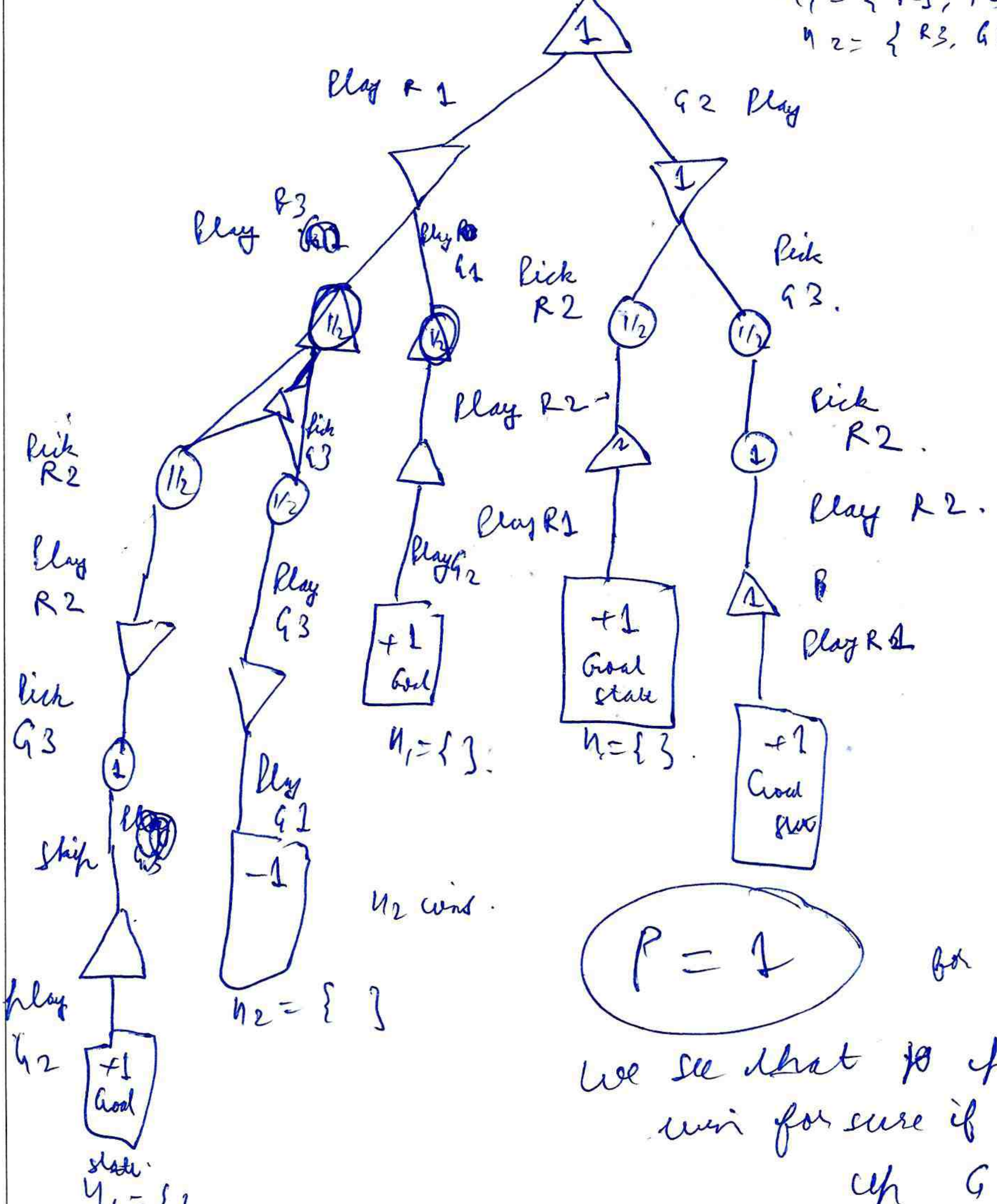
(c) [5 points] For this part, we have the starting hand $H_1$ and also know the opponent's hand $H_2 = \{R3, G1\}$. Similarly, opponent also knows our hand. Draw a game tree. Use the notation discussed in class. I.e., draw upward triangle for our move, downward triangle for opponent's move, a circular node for a chance node, and at the end a square node to represent a leaf. On each edge indicate the player action: card name if it is played, or "pick" to indicate a card is picked, and "pass" to indicate that pile is empty and no move is feasible. Moreover, on edges coming out of chance nodes, indicate probability of occurrence. After completing the game tree, compute the probability of win. (Note: if a player picks multiple cards from the pile, it may result in multiple player moves before opponent gets their chance)

$H_1 = \{R1, G2\}$
$H_2 = \{R3, G1\}$

max

min

max




$P = 1$ for win

We see that player 1 will win for sure if she play up G2 at first

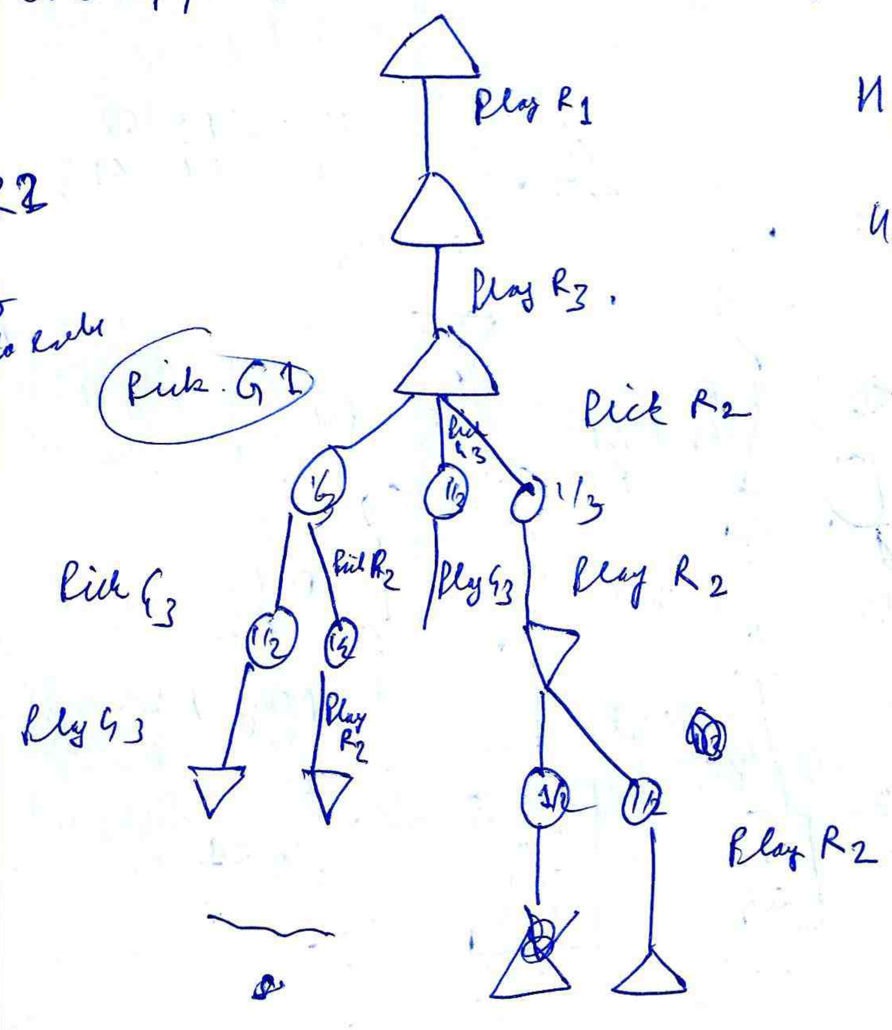
(d) [8 points] Now suppose we don't know opponent's hand, and opponent does not know ours. We initially play R1, and opponent initially plays R3. Note that R3 may or may not be optimal action for the opponent. Draw the game tree now. Using this compute the probability of winning the game after opponent has already played R3.

The opponent hand can be anything, not same as $H_2$.

$H_1 \in \{R_1, G_2\}$.

$H_2 \in \{\text{random}\}$

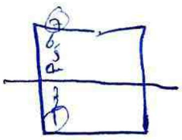
Pick R2 & Pick R3 similar to Pick G1



6. [11 points] Several kinds of symmetries are exploited in constraint satisfaction problems to reduce the size of the search space. This happens in two steps. First we recognize that two or more states represent the same structure in the problem, and hence will behave similarly (i.e., will either both have a solution or both won't have a solution). As an example, consider a 7-queens representation of the problem where Xi represents the location of queen in the i^th column. The board configuration B=(X1,...,X7) is symmetric to configuration Bv=(X7,...,X1) because Bv is a mirror image of B1 along the vertical axis.

One way to break this symmetry we need to add constraint(s) so that only one of the symmetric configurations will be explored by the algorithm. For example, in the previous example we can add an additional "global" constraint X1<=X7. You may check that this constraint allows it to break configurations symmetric along the vertical axis. In this question consider a similar symmetry along the horizontal axis of the board.

(a) Give an algebraic representation of the configuration (Bh) that is symmetric along horizontal axis if B is (X1,...,X7).



$Bh = \{8-X_1, 8-X_2 \ldots 8-X_7\}$ is symm to $B = \{X_1, \ldots X_7\}$.

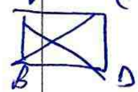
(b) Add constraint(s) so that only one of B and Bh are explored by the CSP algorithm.

to only allow B or Bh we can constraint only one variable s.t $\leq 4$, & if it is $4 \Rightarrow X_2 \leq 3$.

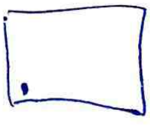
$\Rightarrow X_1 \leq 4 \wedge$ if $X_1 = 4$ $X_2 \leq 3$ eg: $X_1 \leq 4 \wedge (X_1 = 4 \Rightarrow X_2 \leq 3)$ (if $X_1 = 4$, do it for $X_2$).

(c) What other symmetries are present in 7-queens (list as many as you can find)?

If the position of the board Bd is symmetric along the diagonal for B $\Rightarrow$ B & Bd represent the same state / configuration, for both diagonals BC & AD.

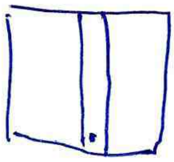


(d) Symmetries can also be dynamic. After a value assignment new symmetries may become valid. Suppose the first queen we place is X1=1. Are there new symmetries present in the problem? If so, which one(s)?



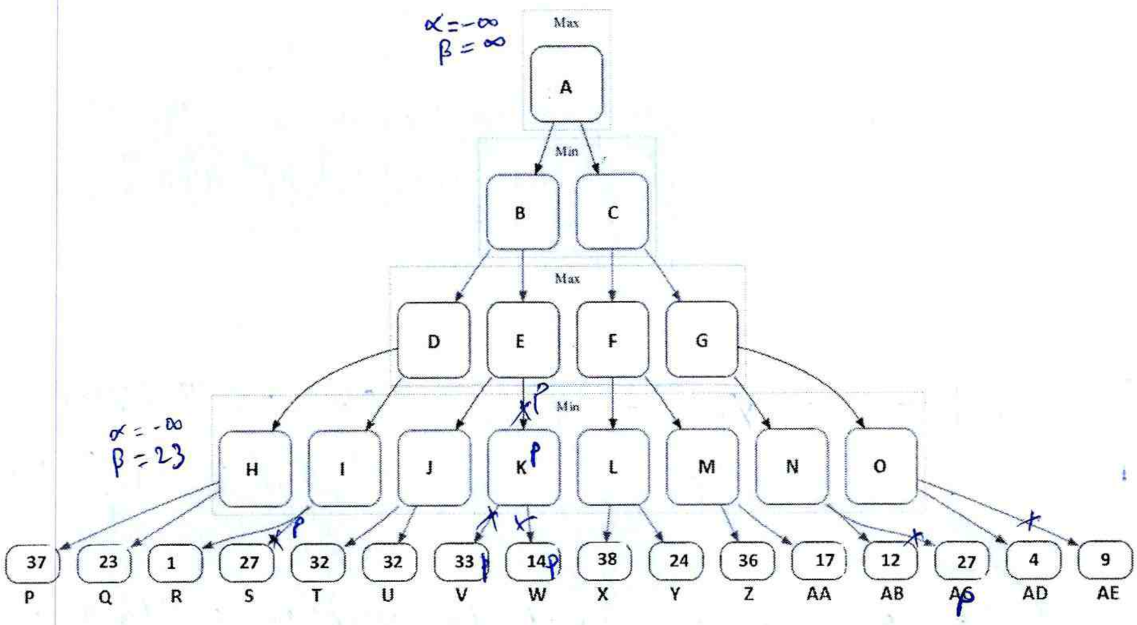
any board position with one queen at the corner is symmetric to this board position i.e $X_1 = 7$, $X_7 = 1$, $X_7 = 7$, $X_1 = 1$

(e) Are there new symmetries present if the first queen is X4=1? If so, which one(s)?



any board position with one queen at midpoint of an edge is symmetric to the i.e $X_1 = 4$ or $X_7 = 4$ or $X_4 = 1$ or $X_4 = 7$.

7. [18 points] The following figure shows an adversarial search tree. If the minimax search always chooses the children from left to right, find the output of the search procedure, i.e., the value for each node in the tree. Assume that the search procedure is using alpha-beta pruning. If a subtree is pruned then label all nodes (including the leaves) of the subtree as pruned.




(a) [2 points] What is the final value of A? What is the optimal path found by the search algorithm?

A = 23, Path found by algorithm :- A → B → D → H → Q

(b) [5 points] Mention all nodes that are pruned by the search procedure.

(marked with P)

S will be pruned as $\alpha = 23$, $\beta = 1$

Similarly K, V, W will be pruned as ($\alpha = 32$, $\beta = 23$)

AC will be pruned (as $\alpha = 23$, $\beta = 12$)

AE will be pruned (as $\alpha = 23$, $\beta = 4$)

(c) [5 points] What is the last node explicitly pruned by the algorithm (not considering other nodes in its subtree, if any)? At the time of this pruning, what are alpha and beta values of the pruned node? What are the alpha and beta values of A?

The Last Node explicitly pruned by the algorith is AE.

values of $\alpha = 23$, $\beta = 4$

and at this state $\alpha = 23$, $\beta = \infty$.

(d) [6 points] Suppose that this evaluation function has a special property: it is known to give the correct minimax value of any internal node to within 2, and the correct minimax values of the leaf nodes exactly. That is, if v is the true minimax value of a particular node, and e is the value of the evaluation function applied to that node, e − 2 ≤ v ≤ e + 2, and v = e if the node is leaf node (e.g., P to AE above). Using this special property, you can modify the alpha-beta pruning algorithm to prune more nodes.

Standard alpha-beta pseudocode is given below (only the max-value recursion). Fill in the box (1) and replace the box (2) so that the pseudocode prunes as many nodes as possible, taking account of this special property of the evaluation function.

```
function MAX-VALUE(node, α, β)
    e ← EVALUATIONFUNCTION(node)
    if node is leaf then
        return e
    end if
    [                    ] (1)
    v ← −∞
    for child ← CHILDREN(node) do
        [v ← MAX(v, MIN-VALUE(child, α, β))] (2)
        if v ≥ β then
            return v
        end if
        α ← MAX(α, v)
    end for
    return v
end function
```

if ~~α − 2 ≥ β~~

$|\alpha - 2| \leq \beta$

replace this by

8. [16 points] In this question we look at Qwordle. This is a variation on the standard Wordle game, where, instead of one, there are two correct words. Since, in our question, the solution words may be meaningless we refer to them as strings. The objective is to find either of the two correct strings (every string is of 5 letters) by guessing one string at a time. For each try you get a response using which you impose further constraints on your search.

1. If a letter doesn't belong to any correct string, it is left as is.
2. If a letter belongs to a correct string but is at wrong position, it is marked by triangles.
3. If a letter belongs to a correct string and is at the correct position, it is marked by squares.
4. If letters are doubly marked by squares or triangles, *all* the marked letters in that guess belong to same correct string.
5. If letters are marked with single squares or triangles, then ~~the~~ all the marked letters belong to ~~both~~ cannot the ~~wor~~ds. The information about which letter corresponds to which correct string is not provided. same correct string. Atleast 2 belong to different strings.
6. No letter is common in the two strings.
7. Correct strings do not have duplicate letters.

| | | | | |
|---|---|---|---|---|
| T△ | O△ | U | C | H |
| G□ | R△ | A | S△ | H |
| C | H | A | O△ | S△ |
| S□ | K | I△ | P△ | A |
| | | | | |
| | | | | |

Correct string attempt row wise.

We define this problem as a CSP. Let $\{G_i \mid i \in \{1, 2, 3, 4, 5\}\}$ and $\{S_i \mid i \in \{1, 2, 3, 4, 5\}\}$ be two sets of variables, denoting the letter assigned to each position, 1 to 5 from left to right respectively. Here $G_i$ denotes variables for the $w_G$ - string that begins with 'G'. Similarly, $S_i$ denotes variables for $w_S$ - the string that begins with 'S'.

(a) [5 points] Compute $|G_i|$ for i=1,…,5 after applying all constraints in Qwordle above. Similarly, compute $|S_i|$ for i=1,…,5.

Sol $|S_1| = 1$ for S  $|G_1| = 1$
$|S_2| = \{I, P\}$ or 2  $G_2 = \{O, R, T\}$
$|S_3| = \{I, P\}$ or 2  $|G_2| = 3$
$|G_3| = 3$
$|G_4| = 3$.

(b) [2 points] Is there an *alldiff* constraint in this problem? If so, list all *alldiff* constraints for the Qwordle above.

Yes there is an all diff constraint.

(c) [2 points] Let V denote the subset of English alphabet where we know that a letter belongs to either of the correct strings. What is V?

$V = \{ S, \cancel{S} \}$

(d) [2 points] We make another try below. We get slightly lucky. What letters can now be proven to belong to $w_G$? What about $w_S$?

| | | | | |
|---|---|---|---|---|
| T△ | O△ | U | C | H |
| G□ | R△ | A | S△ | H |
| C | H | A | O△ | S△ |
| □S□ | K | △I△ | △P△ | A |
| △B△ | △E△ | △L△ | △O△ | W |
| | | | | |

$W_G = \{ E, B, L, O \}$.

(e) [3 points] Suppose our next attempt is to guess $w_G$? Based on the constraints so far, how many possible values can $w_G$ take? Show your work.

5 values

(f) [2 points] We use backtracking search to find $w_G$. We start by assigning first variable $G_1$ = 'G'. Based on heuristics discussed in class, which variable should be assigned next, and what value should be tried first?

$G_1 = G$ then I the f—

[EXTRA PAGE]