CSC265 Fall 2020 Homework Assignment 8

Student 1: Jiawei Chen
Student 2:

1. Consider the following simplified version of the Mastermind game.
   The Chooser chooses a sequence of two (not necessarily different) colours from among {Blue, Green, Red}. The Guesser must discover this sequence.
   Each turn, the Guesser gives the Chooser a sequence of two (not necessarily different) colours from among {Blue, Green, Red}. Then the Chooser responds with the number of positions in which the two sequences agree.
   The game ends when the Chooser answers 2, indicating that the Guesser has discovered the sequence.

   (a) Prove that any decision tree for this problem has height at least 4.
   If you can't solve this problem, you will get a small number of marks for proving that any decision tree for this problem has height at least 3.

   Solution:
   If there exists a decision tree with height less than 4, then it implies it can get the right answer in at most three guesses.

   I will show that this is impossible using the adversary argument.

   Denote the three colors with R, G, B.

   Let's say the first guess is (A_1, B_1). The adversary answers 1 regardless of what the guess is.

   Without loss of generality, assume the chooser never use the same guess twice, because that is essentially wasting a guess. So for the second guess (A_2, B_2), there are two possible cases: the second guess agrees with the first guess in either 0, or 1 position.

   Case 1. The two guesses agree in 0 positions. The adversary replies 1 again to the second guess.
   The possible true answers are (A_1, B_2) and (A_2, B_1). No matter what the third guess is, the adversary can always pick one of them that has not been guessed, and reply correspondingly.

   Case 2. The two guesses agree in 1 position. Without loss of generality, assume that they agree in the first position (i.e. the second guess is (A_1, B_2)). The adversary replies 0. This implies that the true answer has B_1 in position 2. However, the only thing we know about the first position is that it is not A_1.
   Let the third guess be (A_3, B_3). Since there are three possible colours, the adversary can always choose the first position of the true answer so that it is neither A_1 nor A_3.

   This shows that three guesses is not always sufficient. Therefore a correct decision tree must have a path with length at least 4. In other words, the height is at least 4.

   (b) Give a decision tree for this problem that has height 4. Justify why your algorithm is correct.

   Solution:
   See figure 1. R, G, B stands for the three colours.

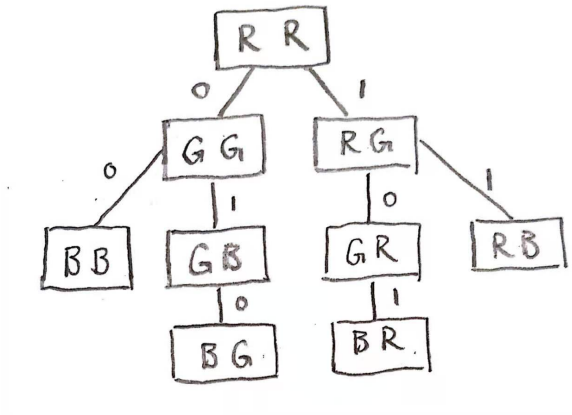

Figure 1: The decision tree

At each node, there are three possible replies: 0, 1, 2. Whenever the guesser receives 2, the game terminates right away. For convenience, I will omit the 2's branch and draw the 0 and 1 branches only.

In fact, at node (G, B) and (G, R). There is only one branch because only possibility left if the reply is not 2. I will explain this in detail in the following passage.

First observe that there are $3 \times 3 = 9$ possible sequences of 2 colours. The root (i.e. the first guess) is (R, R). Based on the reply, we can divide the 9 sequences into 2 subsets. $\{(G, G), (B, B), (G, B), (B, G)\}$ agrees with (R, R) in 0 positions, so they are under the 0-branch. $\{(R, G), (G, R), (B, R), (R, B)\}$ which agree with (R, R) in one position are therefore under the 1-branch.

Suppose we go into the 0-branch of (R, R), we know that neither positions can be R. The second guess is (G, G). If the reply is 0 again, we know that neither position is G. There are only three colours, so clearly the only possibility is (B, B).

If the reply to (G, G) is 1, then we know that one of the positions is G and the other one is not G. As we learnt from the first guess, it can't be R either, so it must be B. So the answer is either (G, B) or (B, G). We try (G, B) first. If it's wrong, we conclude that it is (B, G).

Suppose we go into the 1-branch of (R, R), we know that one of the positions is R, and the other may be G or B. We try (R, G) first. If the reply is 1, then it implies that R in first position is correct and that the second position is neither R nor G. So we conclude (R, B) is correct.

If the reply to (R, G) is 0, then it indicates we have changed the wrong position in our guess. R in the second position is correct. Then we can test (G, R) and (B, R) and one of them must be the true answer.

2. Consider the problem of searching for a value x in an $n \times n$ matrix A of integers in which

the entires of each row are sorted in nondecreasing order from left to right and the entries of each column are sorted in nondecreasing order from top to bottom.
Prove that every algorithm solving this problem which only performs comparisons between x and elements of A must perform at least $2n-1$ comparisons in the worst case.

Solution:
Use an adversary argument.
Strategy: If algorithm compares A[i][j]:x, where i,j$\leq n$, then:
if i+j $> n$ return x<A[i][j]
else return x>A[i][j]

Classify all positions in the matrix as either "eliminated", meaning that x can't appear in that position, or "possible",meaning that x might appear there. Initially all n^2 positions are possible.

Notice that this adversary argument never admits any A[i][j]=x, so all possible positions in the matrix must be eliminated to make sure that the matrix does not include x.

There is one special case for this adversary argument, A[n][1], comparing x with this eliminates row n entirely, but otherwise we have this claim : at least 2 comparisons are required to eliminate any row besides row n, and that comparisons at anywhere else will not reduce the 2 times needed for this row.

Proof: Consider an arbitrary row A[k][1]...A[k][n]. We divide this row into two parts, A[k][1]...A[k][n-k] (part 1) and A[k][n-k+1]...A[n] (part 2). Since every row has a unique row number, this k is never the same for two different rows. In order to eliminate this row, we can compare A[k][n-k]:x and A[k][n-k+1]:x. But this is not enough. We need to prove that the two special positions, A[k][n-k] and A[k][n-k+1] will not be eliminated by comparisons at ANYWHERE ELSE.

Let A[i][j]:x be an arbitrary comparison:
If i+j $> n$, then we can eliminate all positions with row number $\geq$ i and column number $\geq$ j. A[k][n-k+1] will not be eliminated in this process because any position with row number $\leq$ k and column number $\leq$ n-k+1 will have a row-column sum $\leq$ n (contradicting the assumption), not to mention A[k][n-k] which has a even smaller column number.

If i+j $\leq n$, then we can eliminate all positions with row number $\leq$ i and column number $\leq$ j. A[k][n-k] will not be eliminated in this process because any position with row number $\geq$ k and column number $\geq$ n-k+1 will have a row-column sum more than n (contradicting the assumption), not to mention A[k][n-k+1] which has a even bigger column number.

Therefore, we need at least 2(n-1) comparisons to eliminate the first n-1 rows, and 1 comparison to eliminate row n. This is at least 2n-1 comparisons.