2η Εργαστηριακή Άσκηση

ΘΕΜΑ : Κατασκευή παιγνίου

ΟΜΑΔΑ *:*

- Κοτοφώλη Χριστίνα
- Τσαπικούνη Γεωργία

Στην 2η εργαστηριακή άσκηση κατασκευάσαμε ένα πρόγραμμα το οποίο υλοποιεί ένα παίγνιο μεταξύ δύο παιχτών(USER,PC) με τη χρήση του αλγόριθμου MinMax. Παρακάτω θα αναφέρουμε αναλυτικά πως ορίσαμε την κατάσταση παιγνίου, πως ορίσαμε τη αξία των τελικών καταστάσεων μέσα από μία σύντομη περιγραφή του κώδικα .

- ΟΡΙΣΜΟΣ ΚΑΤΑΣΤΑΣΗΣ ΠΑΙΓΝΙΟΥ

Αρχικά, δημιουργήσαμε την κλάση Node η οποία αντιπροσωπεύει κάθε κόμβο του δέντρο που φέρει πληροφορίες για την κατάσταση του παιγνίου κατά την εκτέλεση του αλγορίθμου minMax . Αναλυτικότερα , κάθε κόμβος περιέχει τον πίνακα newNodes που αντιπροσωπεύει όλους τους θυγατρικούς κόμβους που δημιουργούνται από τον τρέχοντα κόμβο , τον posMove που είναι ο καλύτερος θυγατρικός κόμβος για να επιλέξει ο αλγόριθμος ως επόμενη πιθανή

κίνηση και τον πίνακα ακεραίων N που αποθηκεύει των αριθμό των καρτών της κάθε ομάδας την τρέχουσα στιγμή.

```
public class Node{

    private Node[] newNodes ;
    private Node posMove;
    private int[] N;


    public Node(int[] result){
        this.newNodes = new Node[result[0]];
        this.posMove = posMove;
        this.N = new int[result[1]];

    }
```

Εικόνα 1

Επίσης, στη συγκεκριμένη κλάση περιέχονται μέθοδοι που μας επιτρέπουν να παίρνουμε αλλά και να θέτουμε πληροφορίες στα πεδία του κάθε κόμβου .

```
public Node[] getNewNodes(){

    return newNodes;
}

public Node getPosMove(){

    return posMove;
}

public int[] getN(){

    return N;
}

public void setN(int[] A){
    for(int i =0; i< N.length; i++){

        N[i]= A[i];

    }

}
public void setNewNodes(Node[] newNodes) {
    this.newNodes = newNodes;
}
public void setPosMove(Node posMove) {
    this.posMove = posMove;
}
```

Εικόνα 2

Τέλος, η κλάση Node περιέχει και την μέθοδο checkN η οποία ελέγχει εάν ο πίνακας N που περιέχει τις κάρτες της κάθε ομάδας της τρέχουσας κατάστασης είναι μηδέν ή όχι . Αυτή η μέθοδος ανάλογα με το αποτέλεσμα που θα μας επιστρέψει θα καθορίσει και τον νικητή του παιγνίου .

```
public int checkN(){

    for(int i =0; i<N.length; i++){

        if(N[i] != 0){

            return 0;
        }
    }

return 1;
}
```

Εικόνα 3

- ΑΞΙΑ ΤΕΛΙΚΩΝ ΚΑΤΑΣΤΑΣΕΩΝ

Η αξία των τελικών καταστάσεων ορίζεται βάζοντας +1 στην αξία του max φύλλου και -1 για την αξία του min φύλλου. Συγκεκριμένα όταν ο minMax αλγόριθμος φτάσει σε μία τελική κατάσταση, εάν βρίσκεται στο min επίπεδο τότε εμφανίζει +1 στο maxValue υποδηλώνοντας ότι ο νικητής του παιχνιδιού είναι ο PC, ενώ εάν η τελική κατάσταση βρίσκεται στο επίπεδο max τότε σημαίνει ότι κέρδισε ο USER και εμφανίζεται -1 στην αξία του minValue.

```
public static int valueMax = 1;
public static int valueMin = -1;
```

Εικόνα 4

```
if(nextPlayer == valueMax && parent.checkN()==1){

    return(-1);
}
else if(nextPlayer == valueMin && parent.checkN()==1){

    return(+1);
}
```

- ΠΕΡΙΓΡΑΦΗ ΤΟΥ minMaxGame

Στο πρόγραμμα μας εφαρμόζεται ο αλγόριθμος minMax για να υλοποιήσουμε το παίγνιο με τις κάρτες . Αρχικά η μέθοδος inputs() ζητάει από το χρήστη να μας δώσει τον αριθμό των ομάδων με ένα άνω όριο τις τέσσερις ομάδες και τον αριθμό των καρτών με ένα άνω όριο τις 14 κάρτες και επιστρέφει έναν πίνακα ακεραίων result που περιέχει τον αριθμό των καρτών και των ομάδων .Στην inputs επίσης δημιουργούνται δύο πίνακες , ο πίνακας A που μας δείχνει τον αριθμό των φύλλων της κάθε ομάδας και τον πίνακα B που μας δείχνει τον μέγιστο αριθμό φύλλων που μπορείς να αφαιρέσεις από κάθε ομάδα . Παρακάτω παραθέτουμε τον κώδικα της inputs().

```
public static int[] Inputs(){

    Scanner input1 = new Scanner(System.in);
    System.out.println("Give the number of cards M : ");
    int M = input1.nextInt();
    while(M <= 0 || M > 13){
        System.out.println("Please give a positive number less than 14");
        M = input1.nextInt();
    }
    System.out.println("Give the number of teams K:");
    int Klength = input1.nextInt();
    while(Klength <= 0 || Klength > 4){
        System.out.println("Please give a number of team less than 5");
        Klength = input1.nextInt();
    }
    int[] result = new int[2];
    result[0] = M;
    result[1] = Klength;
    A = new int[Klength];
    B = new int[Klength];
```

Εικόνα 5

```
do{
    sum = 0;
    for(int i = 0; i < Klength; i++){

        System.out.println("Enter the number of cards for team  A" + i + ":");
        int K = input1.nextInt();
        while(K < 2){

            System.out.println("Please enter more than one card for team " + i);
            K = input1.nextInt();

        }

        A[i] = K;


        System.out.println("Enter the number of total cards that you want to remove from this team : ");
        int numCards_B = input1.nextInt();
        while( numCards_B >= K){

            System.out.println("Please enter B<A ");
            numCards_B = input1.nextInt();

        }
        B[i] = numCards_B;
```

Εικόνα 6

```
            sum += A[i];

        }
        if(sum != M)
        {
            System.out.println("M is not equal to the sum of values in array A");


        }
    }while(sum != M);


return result;
}
```

Εικόνα 7

```
  public static void copy(int[] array1 , int[] array2){

      for(int i =0; i < array1.length; i++){

          array2[i] = array1[i];


      }

  }
```

Εικόνα 8

Στην εικόνα 8 απεικονίζεται η μέθοδος copy η οποία αντιγράφει τα στοιχεία ενός πίνακα σε έναν άλλον πίνακα και χρησιμοποιείται ως μέσο για να αντιγράψουμε τα στοιχεία του πατέρα στο παιδί στην μέθοδο MinMaxAlgorithm.

Η συνάρτηση MinMaxAlgoritm λαμβάνει δύο ορίσματα , ένα αντικείμενο τύπου Node που υποδηλώνει την τρέχουσα κατάσταση του παιχνιδιού και μία ακέραια τιμή που υποδηλώνει ποιος θα είναι ο επόμενος παίχτης , τέλος η μέθοδος επιστρέφει την καλύτερη κίνηση που μπορεί να κάνει ο αλγόριθμος .

Για κάθε κάρτα ελέγχουμε όλους τους περιορισμούς που μας έχει θέσει η άσκηση δηλαδή αν υπάρχουν διαθέσιμες κάρτες για να αφαιρέσεις και αν ο αριθμός των καρτών που θες να αφαιρέσεις είναι μικρότερος από τις αρχικές κάρτες . Εάν είναι μικρότερος τότε μπορείς να αφαιρέσεις τόσες όσες και το B διαφορετικά ο πίνακας N δεν επηρεάζεται και παραμένει ως είχε. Στη συνέχεια , για κάθε μία κάρτα που αφαιρώ δημιουργείται ένας νέος κόμβος δηλαδή δημιουργείται ένα νέο παιδί , αντιγράφονται οι κάρτες του πατέρα στο παιδί μέσω της copy και το παιδί αφαιρεί j κάρτες από την τρέχουσα ομάδα . Έπειτα αναδρομικά και με τον αλγόριθμο DFS ο τρέχων κόμβος που αποτελεί το παιδί φτιάχνει τα δικά του παιδιά και αποθηκεύονται στον πίνακα nodes . Τέλος , ο πίνακας layers δείχνει προς όλους τους θυγατρικούς κόμβους του parent.

```
for(int i = 0; i < A.length; i++){

    if( parent.getN()[i] > 0 ){

        if(B[i]<(parent.getN()[i])){

            removedCards = B[i];

        }else{

            removedCards = parent.getN()[i];
        }

        for(int j =1; j <= removedCards; j++){

            Node childNode = new Node(result);
            childNode.setN(new int[result[0]]);
            copy(parent.getN(), childNode.getN());
            childNode.getN()[i] = childNode.getN()[i] - j;
            parent.getNewNodes()[k] = childNode;
            nodes[k] = MinMaxAlgorithm(childNode, nextPl);
            layers[k] = childNode;
            k++;

        }
    }
}
```

Εικόνα 9

```
    if(nextPlayer == valueMax){

        bestChild = nodes[0];
        choosenChild = layers[0];
        for(int i = 1; i< k; i++){
            if(nodes[i] > bestChild){
                bestChild = nodes[i];
                choosenChild= layers[i];
            }
        }
    }else{
        bestChild = nodes[0];
        choosenChild = layers[0];
        for(int i = 1; i< k; i++){
            if(nodes[i] < bestChild){

                bestChild = nodes[i];
                choosenChild= layers[i];
            }
        }
    }


parent.setPosMove(choosenChild);
return bestChild ;

}
```

Εικόνα 10

Στην εικόνα 10 απεικονίζεται η διαδικασία που υλοποιείται ο DFS από τα φύλλα προς τον πατέρα κρατώντας την τιμή από τα παιδιά. Αναλυτικότερα , αν ο nextPLayer βρίσκεται στο επίπεδο max τότε για κάθε ένα θυγατρικό κόμβο βρίσκει την καλύτερη τιμή του πίνακα nodes , την εκχωρεί στο bestChild και το chosenChild είναι ένας δείκτης που δείχνει προς το καλύτερο παιδί . Με αντίστοιχο τρόπο υλοποιείται η DFS και για το επίπεδο min με την διαφορά ότι στη συγκεκριμένη περίπτωση καλύτερη τιμή θεωρείται η min τιμή του πίνακα nodes. Τέλος, η καλύτερη επόμενη κίνηση του parent είναι το choosenChild.

Κλείνοντας θα κάνουμε μία αναφορά στη main συνάρτηση και στο πως έχει δομηθεί το παιχνίδι μέσα σε αυτή. Αρχικά , καλούμε την inputs στην οποία ο χρήστης δίνει τα ορίσματα για το πόσες κάρτες θα υπάρχουν στο ταμπλό καθώς και το πόσες είναι οι ομάδες . Με τις εισόδους αυτές του χρήστη δημιουργείται και η ρίζα του παιχνιδιού η οποία αντιπροσωπεύει την αρχική κατάσταση του παιχνιδιού . Εκείνη τη στιγμή εμφανίζεται στην οθόνη μας το μήνυμα «---- START THE GAME ».

```
public static void main(String[] args){

    result = Inputs();
    Scanner input1 = new Scanner(System.in);
    Node root = new Node(result);
    root.setN(A);
    int[] N = root.getN();
    Node pos = root.getPosMove();
    int nextPlayer = valueMax;

    for(int i =0; i< A.length; i++){

        root.getN()[i]= A[i]; // fill the root of the game

    }

    System.out.println("-----------START THE GAME --------------------");
    System.out.printf("Initial cards and teams of the Game \n");
```

Εικόνα 11

Ο βρόχος while είναι ο κύριος βρόχος του παιχνιδιού ο οποίος εκτελείται μέχρι να κερδίσει είτε ο USER είτε ο PC. Εάν είναι η σειρά του του pc , καλείται η μέθοδος του minMaxAlgorithm με το όρισμα valueMax . Μέσω αυτού του αλγορίθμου ο pc εκτελεί την καλύτερη κίνηση γι' αυτόν , ενημερώνει τον πίνακα N με τη νέα κατάσταση του παιχνιδιού. Εάν η σειρά είναι του χρήστη , ζητείται από το χρήστη να επιλέξει μία ομάδα και να αφαιρέσει ένα συγκεκριμένο αριθμό καρτών από αυτή την ομάδα. Ο πίνακας N αντίστοιχα ενημερώνεται υποδηλώνοντας τη νέα κατάσταση του παιχνιδιού. Αυτή η διαδικασία συνεχίζεται να υλοποιείται μέσα στο βρόγχο έως ότου εμφανιστεί κάποιο μήνυμα στην οθόνη που υποδηλώνει τον νικητή του παιχνιδιού.

```
while(true){


    //check if the nextplayer is the pc and if there is not any card to play, then user is the winner
    if(nextPlayer == valueMax  && root.checkN()== 1){

        System.out.printf("\n\n");
        System.out.printf("USER IS THE WINNER!!\n");
        System.out.printf("--END OF THE GAME--");
        break;
    }//check if the nextplayer is the user and if there is not any card to play, then pc is the winner
    else if(nextPlayer == valueMin  && root.checkN()==1){

        System.out.printf("\n\n");
        System.out.printf("PC IS THE WINNER\n");
        System.out.printf("--END OF THE GAME--");
        break;
    }
```

Εικόνα 12

```
else if(nextPlayer == valueMax ){//if the pc has cards to play call the MINMAXALGORITHM

    System.out.printf("---------------------------------------- \n");
    System.out.printf("--TURN OF PC--\n");
    MinMaxAlgorithm(root,valueMax);
    root = root.getPosMove();

    for (int i = 0;i < A.length;i++)
    {
        System.out.println("Team " + i + " has "  +  " " + root.getN()[i] + " cards");
    }
nextPlayer = valueMin;

}
else{//if user has cards to play then do all the steps bellow like choosing team , ask for the nu

    System.out.printf("\n CHOOSE TEAM FOR REMOVING CARDS\n");
    for (int i = 0;i <A.length;i++){

        if (root.getN()[i] > 0){

            System.out.printf("Removing from team " + i + " press " + i + "\n");
```

Εικόνα 13

```
        }
        int new_team = input1.nextInt();
        while(new_team >= A.length ){
            System.out.println("Please give a valid team ");
            new_team = input1.nextInt();
        }

        if (B[new_team] < (root.getN()[new_team])){

            System.out.printf("Remove 1 to " + B[new_team] + " cards \n");

            int cards_removed = input1.nextInt();
            while(cards_removed< 1 || cards_removed > B[new_team]){

                System.out.println("Please give a positive number less than or equal " +B[new_team]);
                cards_removed = input1.nextInt();
            }
            root.getN()[new_team] = root.getN()[new_team] - cards_removed;
        }
```

Εικόνα 14

```
        else{

            System.out.printf("Remove 1 to " +  root.getN()[new_team] + " cards \n");
            int cards_removed = input1.nextInt();
            while(cards_removed< 1 || cards_removed > root.getN()[new_team]){
                System.out.println("Please give a positive number less than or equal " + root.getN()[new
                cards_removed = input1.nextInt();
            }
            root.getN()[new_team] = root.getN()[new_team] - cards_removed;
        }

        System.out.printf("---------------------------------------- \n");
        System.out.printf("--TURN OF USER--\n");
        for (int i = 0;i < A.length;i++){

            System.out.println("Team " + i + " has "  +  " " + root.getN()[i] + " cards");
        }
        nextPlayer = valueMax;
```

Εικόνα 15