ΠΡΟΑΙΡΕΤΙΚΗ ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ ΔΟΜΗΜΕΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ρούμπας Κωνσταντίνος, ΑΕΜ: 9976
Σοφικίτης Οδυσσέας, ΑΕΜ: 10130

Ο κώδικας που χρησιμοποιήθηκε είναι απλός. Αποτελείται κυρίως από δομές επανάληψης και ελέγχου. Οι περισσότερες μεταβλητές χρησιμοποιούνται σαν counters, ενώ υπάρχουν και λίγες μεταβλητές flag. Γενικά, η στρατηγική του προγράμματος περιλαμβάνει αλλεπάλληλες αναγνώσεις του πίνακα "board" με σκοπό την επιλογή της καλύτερης κίνησης. Βασικό σκεπτικό του προγράμματος είναι ο διαχωρισμός του παιχνιδιού ανάλογα με τις διαστάσεις του πίνακα και ανάλογα με το ποιον παίκτη αντιπροσωπεύει.

1. Παίκτης 1

1.1 Μακρόστενος Πίνακας

Ο παίκτης παίζει πρώτη φορά στην θέση coloumns/2. Με αυτό τον τρόπο υπάρχουν επιλογές ανάπτυξης προς όλες τις κατευθύνσεις (διαγώνια, πλάγια, κατακόρυφα) ανεξάρτητα από την επιλογή του αντιπάλου. Όλες οι επόμενες κινήσεις υπολογίζονται με την βοήθεια ενός πίνακα διάστασης columns- στον οποίο αποθηκεύεται το πόσο "καλό" είναι να παίξει σε αυτήν την στήλη- και της μεταβλητής max. Αυτό γίνεται διαβάζοντας τον πίνακα 4 φορές σε κάθε γύρο, μια φορά για κάθε πιθανή κατεύθυνση (πάνω δεξιά και κάτω αριστερά, πάνω αριστερά και κάτω δεξιά, πλάγια και κατακόρυφα), με την χρήση τον εντολών for και while. (Εικόνα 1) Το σκεπτικό αυτό είναι απλό, αλλά θα μπορούσε να έχει πολλές υλοποιήσεις όσον αφορά τον κώδικα. Η κύρια δυσκολία βρίσκεται στο να προγραμματιστεί σωστά ο παίκτης ώστε να αποφασίζει σε ποιο από τα δύο (ή περισσότερα) κενά θα παίζει όταν αυτά υπάρχουν στην ίδια διαγώνιο (ή στήλη και σειρά αντίστοιχα). Αυτό ξεπεράστηκε με την χρήση κάποιων βοηθητικών μεταβλητών/δεικτών (i, j, tempscore) και συγκεκριμένους ελέγχους με την εντολή if.

```
tempscore=(int *)calloc(columns, sizeof(int));
for(c=0; c<columns; c++){                          //checking diagonally if we have "enough"
  for(r=0; r<rows-1; r++){                         //consecutive chips to play there
    if(!board[r][c] && board[r+1][c]!=0){          //first up-right and down-left
      i=r-1;
      j=c+1;
      while(j>-1 && i>-1){
        if(board[i][j]==1) tempscore[c]++;
        else if(board[i][j]==2 || (!board[i][j] && board[i+1][j]!=0)) break;
        i--;
        j++;
      }
      i=r+1;
      j=c-1;
      while(j<columns && i<rows-1){
        if(board[i][j]==1) tempscore[c]++;
        else if(board[i][j]==2 || (!board[i][j] && board[i+1][j]!=0)) break;
        i++;
        j--;
      }
    }
  }
}
for(c=0; c<columns-1; c++){          //plus one would cause segmentation fault in the previous for
  if(!board[rows-1][c]){             //so we are checking the last row separately
    i=rows-2;
    j=c+1;
    while(j>-1 && i>-1){
      if(board[i][j]==1) tempscore[c]++;
      else if(board[i][j]==2 || !board[i][j]) break;
      i--;
      j++;
    }
  }
}
for(c=0; c<columns; c++){          //choosing the best tempscore
  if(tempscore[c]>max){
    move=c;
    max=tempscore[c];
  }
}
```

Εικ. 1: Υπολογισμός tempscore κάθε στήλης για διαγώνιες κινήσεις

Κατά κύριο λόγο, το πρόγραμμα δεν ασχολείται τόσο με την παρεμπόδιση του αντιπάλου, όσο με τις κατάλληλες κινήσεις για το μέγιστο δυνατό score. Παρόλα αυτά, υπάρχει ένας έλεγχος ώστε να αποκόπτει τον αντίπαλο αν έχει μεγαλύτερο κατακόρυφο score από το εκάστοτε max του δικού μας παίκτη. Τέλος, σε περίπτωση λάθους του κώδικα, υπάρχει η μεταβλήτη-flag “wrong” που ελέγχεται για να παίξει ο παίκτης τυχαία. (Εικόνες 2, 3)

```
tempscore=(int *)calloc(columns, sizeof(int));
for(c=0; c<columns; c++){
    if(board[0][c]!=0) continue;
    for(r=1; r<rows; r++){
    if(board[r][c]==2) tempscore[c]++;
    else if(board[r][c]==1) break;
  }
}
for(c=0; c<columns; c++){
  if(tempscore[c]>max){
    move=c;
    max=tempscore[c];
  }
}
free(tempscore);
```

Εικ. 2: Παρεμπόδιση αντιπάλου

```
if(move<0 || move>columns-1) wrong=0;
else if(board[0][move]!=0) wrong=0;

if(!wrong){
  int* moves = (int*)calloc(b->columns, sizeof(int));
  int counter = 0;
  for(i = 0; i < b->columns; i++){
      if(b->board[0][i] == 0){
        moves[counter++] = i;
      }
  }
  move = rand() % counter;
  int ret_move =  moves[move];
  free(moves);
  return ret_move;
}
```

Εικ. 3: Τυχαία κίνηση λόγω λάθους

1.2 Τετραγωνικός Πίνακας

Ο κώδικας του τετραγωνικού πίνακα είναι ακόμη απλούστερος. Εδώ δεν ακολουθείται η ίδια λογική, δηλαδή δεν χρησιμοποιείται ο tempscore για την καλύτερη κίνηση. Αντίθετα, το πρόγραμμα προσπαθεί να "κερδίσει" την δευτερεύουσα διαγώνιο. Έτσι, η πρώτη κίνηση είναι στη θέση 0. Στην συνέχεια, διαβάζεται συνεχώς η διαγώνιος και αν κάποια θέση είναι διαθέσιμη τότε παίζει εκεί (Εικόνα 4). Αντίθετα, το πρόγραμμα επιλέγει τυχαία με συγκεκριμένους ελέγχους ώστε να μην "χαλάσει" την διαγώνιο. Στην περίπτωση που καμία τέτοια θέση δεν είναι διαθέσιμη παίζει στην πιο δεξιά διαθέσιμη στήλη. (Εικόνα 5)

```
if(!turn) return move=0;

else{
  r=0;
  c=columns-1;
  while(r<rows-1 && c>0){
    if(!board[r][c] && board[r+1][c]!=0){
      move=c;
      return move;
    }
    r++;
    c--;
  }
```

Εικ. 4: Έλεγχος διαγωνίου

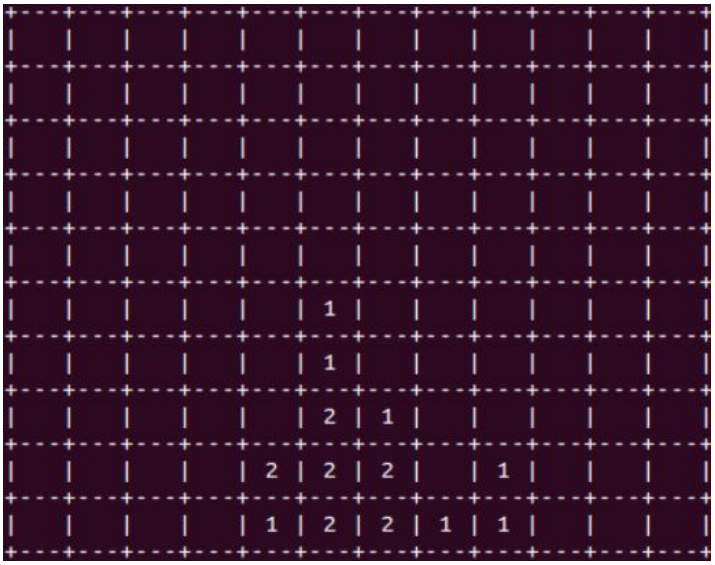
```
int* moves = (int*)calloc(b->columns, sizeof(int));
int counter = 0;
int ret_move;
for(i = 0; i < columns; i++){
  r=rows-1-i;
  c=i;
  if(c==1 && !board[rows-1][1]) continue;
  else if(!board[r][c] && board[r+2][c]!=0) continue;
  else if(board[0][i] == 0){
      moves[counter++] = i;
  }
}
if(!counter){
  for(c = columns-1; c > -1; c--){
    if(!board[0][c]) return move=c;
  }
}
  int move = rand() % counter;
  ret_move =  moves[move];
  free(moves);
  return ret_move;
```

Εικ. 5: Τυχαία επιλογή κίνησης με προσοχή

2. Παίκτης 2

1.1 Μακρόστενος Πίνακας

Αποφασίσαμε να μην πραγματοποιήσουμε αλλαγές στον κώδικα του Παίκτη 1, αλλά να τον κρατήσουμε ίδιο και για τον Παίκτη 2. Στο παρακάτω παράδειγμα, πίνακα 10x12, γίνεται αισθητή η χρήση της tempscore.



Εικ. 6: Πριν την κίνηση του Παίκτη 2

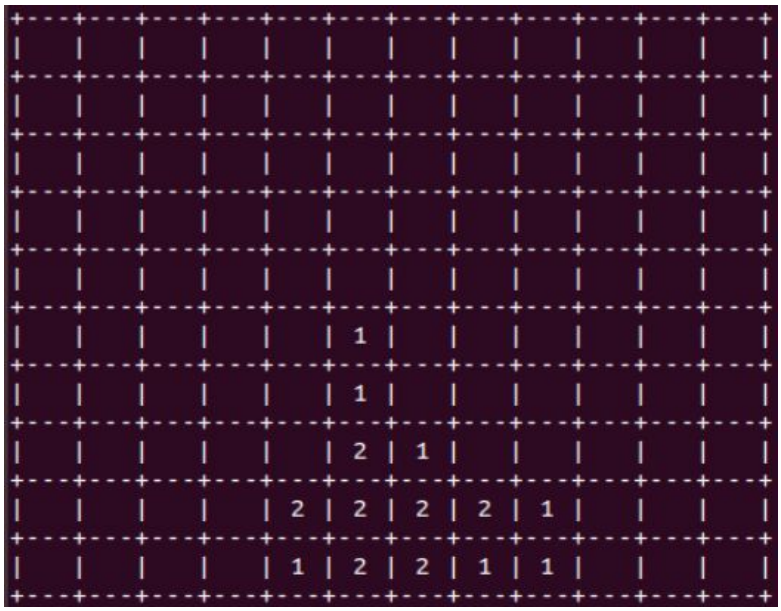
- Ελέγχοντας πρώτα τις κατευθύνσεις πάνω δεξιά και κάτω αριστερά, οι θεσεις: {6,6}, {9,3}, {8,7}, αποκτούν tempscore 2, 2, 1 αντίστοιχα (max: 2, move: 3).

- Έπειτα για τις κατευθύνσεις πάνω αριστερά και κάτω δεξιά, η θέση: {7,4}, αποκτά tempscore 2 (max: 2, move: 3).

- Οι πλάγιες θεσεις: {7,4}, {8,7}, αποκτούν tempscore 1 και 3 αντίστοιχα (max: 3, move: 7).

- Η κατακόρυφη θέση: {7,4} αποκτά tempscore 1 (max: 3, move: 7).

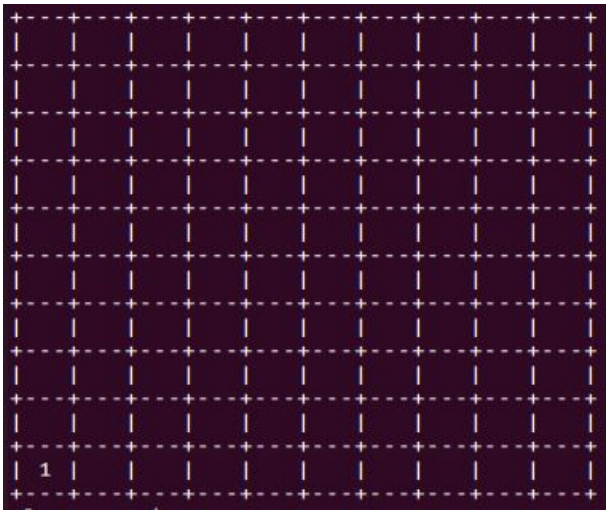
Άρα η επόμενη κίνηση θα πραγματοποιηθεί στη θέση {8,7} του πίνακα. Δεν αποκόπτει τον αντίπαλο αφού δεν έχει μεγαλύτερο κατακόρυφο score απο το 3 (max).



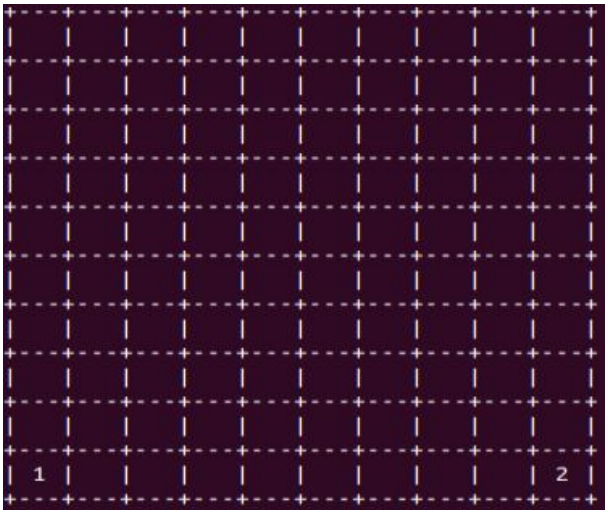
Εικ. 7: Μετά την κίνηση του Παίκτη 2

1.2 Τετραγωνικός Πίνακας

Ο κώδικας, για όταν ο πίνακας είναι τετραγωνικός, του Παίκτη 1, είναι ίδιος και για τον Παίκτη 2 με την διαφορά ότι σαν πρώτη κίνηση, αν έχει παιχτεί η θέση για μια από τις διαγωνίους, επιλέγει την άλλη που είναι διαθέσιμη.



Εικ. 8: Πριν την κίνηση του Παίκτη 2



Εικ. 9:Μετά την κίνηση του Παίκτη 2

```
if(!turn){

        if(!board[rows-1][0]  && !board[rows-1][1]){
                move=0;
                return move;
        }
        else{
                move=columns-1;
                return move;
        }
}
```

Εικ. 10: Έλεγχος διαγωνίου