МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Игра Уголки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование темы проекта или работы)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Алгоритмы и структуры данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

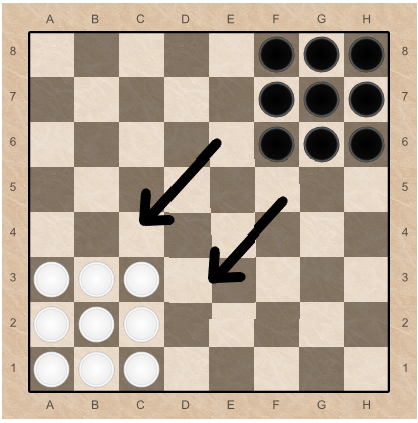
С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

**Текст задачи:**

Цель игры переставить шашки из своего угла (дома) в угол соперника раньше, чем он построит свои в Вашем углу. Ходы делаются по очереди. Движение возможно по горизонталям и вертикалям только в направлениях к углу противника. За один ход передвигается не более одной шашки. Шашка может либо передвигаться на одно поле, либо перепрыгивать через одну или несколько шашек.

Игра заканчивается, когда один из игроков построил все свои шашки в доме противника. Все шашки необходимо вывести из своего дома не позднее чем за 3\*'количество шашек' ходов - иначе будет засчитан проигрыш.

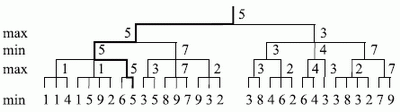


**Описание принципов минимакса**

Возвращается оценка состояния, которая получается при ходе из ветки и сравнивается с другими поддеревьями. С помощью алгоритма «Минимакс» значения листьев поднимаются к корню. В результате работы алгоритма мы получим позицию с наибольшим количеством очков для компьютера на данном ходе, при этом для каждого отдельного случая мы сделаем просчет следующих возможных ходов в зависимости от максимального уровня рекурсии.

Функция scorebot-оценочная функция, используемая алгоритмом минмакса. Структура задана с помощью бинарного дерева из-за особенностей задачи (т.е. робот может пойти или вниз, или влево). Сначала, с помощью функции treewrite записываются значения листьев дерева игры. А уже с помощью функции minmax. Ищется и записывается по алгоритмы нужное нам значение.

**Пример расчета минимакс**



Первым шагом поднимаемся от листьев к корню. Минимальное значение для родителей: 1,1,5,3,7,2,3,2,3,2,4,3,2,7. Для следующего уровня, аналогично находим, но уже из принципа максимума: 5,7,3,4,7. Аналогично, минимумы: 5,3. И уже из минимума выбираем максимальное

И именно эта ветка, с этим значением нам будет лучше всего подходить.

**Программный код:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
// Оценочная функция scorebot //  
int scorebot(int row, int col,int\*\* mass, int n, int ki, int kg, int\*\* OFbot){  
 bool flag1=true,flag2=true;  
 int f1=n;  
 if(row>0 and mass[row-1][col]==0){ // Ход на одну влево  
 f1=f1+1; // Прибавление к числу оценки +1  
 ki=ki+1;  
 if(ki<4) {  
 f1=scorebot(row-1, col,mass,f1,ki,kg,OFbot); // Вызов данной функции для позиции слева  
 }  
 }  
 else{  
 if(row>1 and mass[row-2][col]==0){ // Ход через фишку влево  
 f1=f1+3; // Прибавление к числу оценки +3  
 if(ki<4) {  
 f1=scorebot(row-2,col,mass,f1,ki,kg,OFbot); // Вызов данной функции рекурсивно для позиции через фишку слева  
 }  
 }  
 }  
 int f2=n;  
 if(col>0 and mass[row][col-1]==0){ // Ход на одну вниз  
 f2=f2+1; // Прибавление к числу оценки +1  
 kg=kg+1;  
 if(kg<4) {  
 f2=scorebot(row,col-1,mass,f2,ki,kg,OFbot); // Вызов данной фунции рекурсивно для позции вниз  
 }  
 }  
 else{  
 if(col>1 and mass[row][col-2]==0){ // Ход на через фишку вниз  
 f2=f2+3; // Прибавление к числу оценки +3  
 if(kg<4) {  
 f2=scorebot(row,col-2,mass,f2,ki,kg,OFbot); // Вызов данной функции для позиции через фишку вниз  
 }  
 }  
 }  
 if(f2>=f1){ // Блок выбора наибольшего числа оценки  
 if(ki==0 and kg==0){  
 OFbot[row][col]=f2;  
 }  
 return f2;  
 }  
 if(f2<f1){  
 if(ki==0 and kg==0){  
 OFbot[row][col]=f1;  
 }  
 return f1;  
 }  
}  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
struct tnode {  
 int num; // поле данных  
 struct tnode \*left; // левый потомок  
 struct tnode \*right; // правый потомок  
};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
struct tnode \* addnode(int x, tnode \*tree,int k) {  
 if (tree == NULL) { // Если дерева нет, то формируем корень  
 tree =new tnode; // память под узел  
 tree->num = x; // поле данных  
 tree->left = NULL;  
 tree->right = NULL; // ветви инициализируем пустотой  
 }  
 if (k>0 and k<4) { // условие добавление левого потомка  
 tree->left = addnode(x, tree->left, k+1);  
 tree->right = addnode(x, tree->right, k+1);  
 }  
 return(tree);  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
void treewrite(tnode \*tree,int\* koefof) {  
 if (tree!=NULL) { //Пока не встретится пустой узел  
 treewrite(tree->left,koefof); //Рекурсивная функция для левого поддерева  
 treewrite(tree->right,koefof); //Рекурсивная функция для правого поддерева  
 }  
 if(tree->left==NULL or tree->right==NULL){  
 for(int i=0;i<4;i++){  
 if(koefof[i]!=0){  
 tree->num = koefof[i]; // Запись оценки из массива оценок  
 koefof[i]=0;  
 break;  
 }  
 }  
 }  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
int minmax(tnode \*tree, int koef){  
 int readln1=0,readln2=0;  
 if (tree->left!=NULL or tree->right!=NULL) { //Пока не встретится пустые дети  
 readln1=minmax(tree->left,koef); //Рекурсивная функция для левого поддерева  
 readln2=minmax(tree->right,koef); //Рекурсивная функция для правого поддерева  
 if(koef%2!=0){  
 if(readln1>=readln2){  
 tree -> num=readln2;  
 }  
 else{  
 tree -> num=readln1;  
 }  
 }  
 if(koef%2==0){  
 if(readln1>=readln2){  
 tree -> num=readln1;  
 }  
 else{  
 tree -> num=readln2;  
 }  
 }  
 }  
 if(koef>=0){  
 return tree-> num;  
 }  
  
}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/