**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Построение и Анализ Алгоритмов»**

**Тема: Поиск с возвращением**

**Вариант 1h**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5382 |  | Забалуев Н.Е. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

1. *Содержательная постановка задачи*

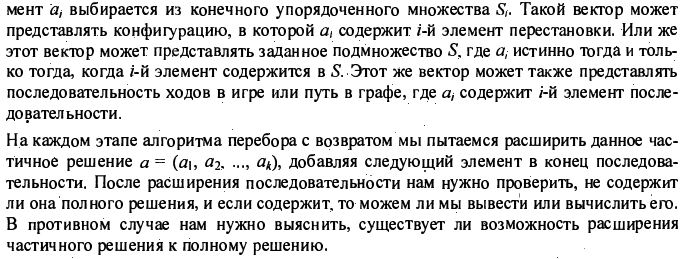
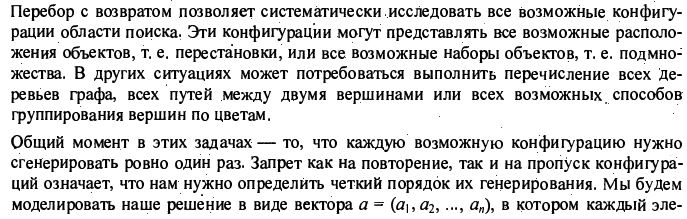
Маршрут коня на шахматной доске n\*n. Требуется определить, существует ли способ передвижения коня по шахматной доске, такой что конь посещает каждую клетку точно один раз и возвращается в исходную точку.

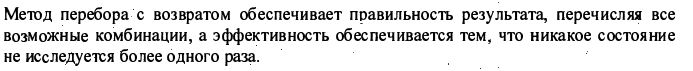
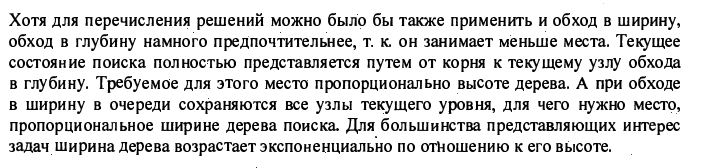
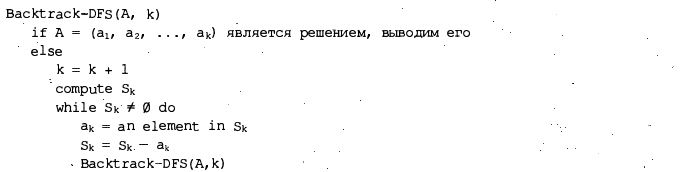
1. *Теоретические сведения*

Задача о ходе коня — задача о нахождении маршрута шахматного коня, проходящего через все поля доски по одному разу.

В терминах теории графов каждый маршрут коня, проходящий через все поля шахматной доски, соответствует гамильтонову пути (или циклу, если маршрут замкнутый) в графе, вершинами которого являются поля доски, и два поля соединены ребром, если с одного можно попасть на другое за один ход коня.

Замкнутые маршруты существуют на досках n\*n при n>=6. Также следует отметить, то что неважно из какой точки начинается обход, и то, что для нахождения верной последовательности быстрее стоит использовать в качестве допустимых ходов те клетки, у которых количество возможных ходов наименьшее.





1. *Описание функций*

bool is\_a\_solution(vector<int> sequence, int d)

//Проверяет, является ли текущая перестановка решением

void process\_solution(vector<int> sequence, int d)

//Функция выводит решение, если таковое нашлось

void construct\_candidates (vector<int> sequence, vector<bool> available\_cells, int d, vector<int> &candidates)

//Функция определения кандидатов на следующую позицию

void sort\_candidates(vector<int> sequence, vector<bool> available\_cells, int d, vector<int> &candidates)

//Сортировка кандидатов по наименьшему числу кандидатов у них самих

void backtrack( vector<int> sequence, vector<bool> available\_cells, int d)

//Поиск с возвращением

1. *Заключение*

В данной лабораторной мы познакомились с алгоритмом работы поиска с возвращением. Как и ожидалось метод прорабатывает все возможные ветви решения, что сказывается на его времени работы.

**Приложение А. Исходный код.**

main.cpp

#include <iostream>

#include <vector>

#include <windows.h>

#include <ctime>

using namespace std;

bool finished = false;

bool is\_a\_solution(vector<int> sequence, int d)

{

int n = sequence.size();

int last = sequence.back();

if ((n==d\*d) && (last==(2\*d+1)))

{

return true;

}

return false;

}

void process\_solution(vector<int> sequence, int d)

{

cout<<"\nНайдено решение!"<<endl;

for (int i=0; i<d\*d; i++)

{

cout<<sequence[i]<<" ";

if ((i+1)%d==0)

{

cout<<endl;

}

}

finished = true;

}

void construct\_candidates (vector<int> sequence, vector<bool> available\_cells, int d, vector<int> &candidates)

{

int last = sequence.back();

if(last/d > 1)

{

if((last%d >= 1) && (available\_cells[last-2\*d-1] == 0))

{

candidates.push\_back(last-2\*d-1);

}

if((last%d <= d-2) && (available\_cells[last-2\*d+1] == 0))

{

candidates.push\_back(last-2\*d+1);

}

}

if(last%d < d-2)

{

if((last/d >=1) && (available\_cells[last-d+2] == 0))

{

candidates.push\_back(last-d+2);

}

if((last/d <= d-2) && (available\_cells[last+d+2] == 0))

{

candidates.push\_back(last+d+2);

}

}

if(last/d < d-2)

{

if((last%d <= d-2) && (available\_cells[last+2\*d+1] == 0))

{

candidates.push\_back(last+2\*d+1);

}

if((last%d >= 1) && (available\_cells[last+2\*d-1] == 0))

{

candidates.push\_back(last+2\*d-1);

}

}

if(last%d > 1)

{

if((last/d <= d-2) && (available\_cells[last+d-2] == 0))

{

candidates.push\_back(last+d-2);

}

if((last/d >= 1) && (available\_cells[last-d-2] == 0))

{

candidates.push\_back(last-d-2);

}

}

for(int i=0; i<candidates.size(); i++)

{

if ((sequence.size()!=d\*d-1) && (candidates.at(i)==(2\*d+1)))

{

candidates.erase (candidates.begin()+i);

break;

}

if (candidates.at(i)==(d-1))

{

candidates.clear();

candidates.push\_back(d-1);

break;

}

if (candidates.at(i)==(d\*d-1))

{

candidates.clear();

candidates.push\_back(d\*d-1);

break;

}

if (candidates.at(i)==((d-1)\*d))

{

candidates.clear();

candidates.push\_back((d-1)\*d);

break;

}

}

}

void sort\_candidates(vector<int> sequence, vector<bool> available\_cells, int d, vector<int> &candidates)

{

int a=7;

int b=candidates.size();

vector<int> next\_candidates;

for (int i=0; i<b; i++)

{

sequence.push\_back(candidates.at(i));

construct\_candidates(sequence, available\_cells, d, next\_candidates);

if (next\_candidates.size()<a)

{

a=next\_candidates.size();

swap(candidates.at(i), candidates.front());

}

sequence.pop\_back();

}

}

void backtrack( vector<int> sequence, vector<bool> available\_cells, int d)

{

vector<int> candidates;

if (is\_a\_solution(sequence, d))

{

process\_solution(sequence, d);

}

else

{

construct\_candidates(sequence, available\_cells, d, candidates);

sort\_candidates(sequence, available\_cells, d, candidates);

for (int i=0; i<candidates.size(); i++)

{

sequence.push\_back(candidates.at(i));

available\_cells[sequence.back()]=1;

backtrack(sequence, available\_cells, d);

available\_cells[sequence.back()]=0;

sequence.pop\_back();

if (finished) return;

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int d=0;

cout<<"Введите размер доски (d>=6): "<<endl;

cin>>d;

while (d<6)

{

cout<<"Ошибка! Для доски меньше, чем 6\*6 не существует замкнутого обхода конём."<<endl;

cout<<"Введите размер доски (d>=6): "<<endl;

cin>>d;

system("cls");

}

vector<int> sequence;

vector<bool> available\_cells(d\*d, 0);

sequence.push\_back(0);

available\_cells[0]=1;

sequence.push\_back(d+2);

available\_cells[d+2]=1;

clock\_t t;

t = clock();

backtrack(sequence, available\_cells, d);

t = clock() - t;

printf ("It took me %d clicks (%f seconds).\n",t,((float)t)/CLOCKS\_PER\_SEC);

if (!finished)

{

cout<<"\nРешений не найдено.";

}

system("pause");

}