**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: Программирование алгоритма поиска с возвращением

Вариант 4

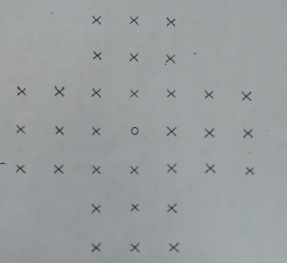
|  |  |  |
| --- | --- | --- |
| Студент гр. 6382 |  | Воропаев А.О. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2017

**Задание:**

Игра солитер. Тридцать две фишки расставляются, как показано ниже, на все клетки доски, кроме центральной. Фишка переносится на свободную клетку через любую соседнюю фишку, которая при этом снимается с доски (ходить можно только по вертикали и горизонтали).



Найдите последовательность «прыжков», такую, чтобы на доске осталась всего одна фишка в центральной клетке.

**Алгоритм:**

На поле 33 клетки. Нумеруем их от 0 до 32 в любом порядке. После этого каждую позицию можно представить в виде 33-битного двоичного числа (1 в k-м разряде означает, что на k-м поле есть фишка). Создаём массив возможных ходов. Каждый ход описывается двумя масками: в первой — 2 ненулевых бита, указывающие, на каких полях должны были находиться фишки до хода, во второй — 1 бит, указывающий, где будет фишка после хода.  
Для инициализации ходов берём карту поля (fld), просматриваем на ней все отрезки длины 3. Если все клетки принадлежат игровому полю (номера неотрицательны), то такой отрезок даёт два хода. Заводим массив, в котором каждой возможной позиции соответствует 1 бит. Позиций у нас 2^33, и такой массив займёт 1 гигабайт. Этот массив нам понадобится, чтобы помечать уже обработанные позиции.

**Backtracking:**  
Выписываем маску start, соответствующую стартовой позиции, и начинаем решение.

Функция Solve рекурсивная и вызывает цикл который, прогоняет условия (u & ~s) == 0 && (v & s) == 0. Поиск происходит в глубину. Функции Solve передаётся единственный параметр — маска s позиции. Проверяем, текущая позиция является ли завершающейся. Это делается проверкой условия s == 0x000010000LL. Если условие срабатывает, возвращаем true — позиция имеет решение. Если текущая позиция не является последней, то перебираем все ходы, и для каждого смотрим, возможен ли он. Если ход возможен, вызываем Solve для позиции, возникшей после этого хода, и смотрим результат. Если true — мы нашли решение, сохраняем наш ход в стеке Solution, и сами возвращаем true. Если же ни один из ходов не привел к результату, значит, позиция тупиковая — возвращаем false.  
После того, как функция Solve вычислилась (и вернула true), в стеке Solution оказалась в точности та последовательность ходов, которая привела к решению.

**Описание функций:**

\*Функция generate создает случайное поле фишками на нем.

long long generate(int n)

int n – число фишек

\*Функции AddStep и InitSteps создают массив всевозможных ходов.

void AddStep(int a, int b, int c)

int a, int b, int c- точки нашего отрезка

\*Функция Solve для поиска решений

bool Solve(long long s)

long long s-массив решений

\*Функция create\_A для простого представления поля

void create\_A(long long t)

long long t – стартовая позиция

\*Функция print для отрисовки поля

void print()

\*Функция save для сохранения последовательности, приводящей к выигрышу в файл

void save(char file[])

char file[] – файл, в который будет производиться вывод

\*Функция animation для вывода на консоль полной информации на основании полученной из файла информации

void animation(char file[])

char file[] – файл, где хранится последовательность, приводящая к выигрышу

\*Функция zeroing для обнуления исходного поля для последующих запусков программы.

void zeroing()

**Тестирование:**

1 случай

Шаг 1

0| X X X

1| X X X

2| X X X X X X X

3| X X X O X X X

4| X X X X X X X

5| X X X

6| X X X

--------------------

1. 1 2 3 4 5 6

Шаг 2

Начало: x = 1 y = 3

Удаляем: x = 2 y = 3

Конец: x = 3 y = 3

Шаг 3

Начало: x = 2 y = 1

Удаляем: x = 2 y = 2

Конец: x = 2 y = 3

Шаг 4

Начало: x = 0 y = 2

Удаляем: x = 1 y = 2

Конец: x = 2 y = 2

Шаг 5

Конец: x = 1 y = 2

Начало: x = 2 y = 2

Удаляем: x = 3 y = 2

Шаг 6

Конец: x = 1 y = 3

Начало: x = 2 y = 3

Удаляем: x = 3 y = 3

Шаг 7

Начало: x = 0 y = 3

Удаляем: x = 1 y = 3

Конец: x = 2 y = 3

Шаг 8

Конец: x = 2 y = 1

Начало: x = 3 y = 1

Удаляем: x = 4 y = 1

Шаг 9

Начало: x = 2 y = 0

Удаляем: x = 2 y = 1

Конец: x = 2 y = 2

Шаг 10

Начало: x = 1 y = 2

Удаляем: x = 2 y = 2

Конец: x = 3 y = 2

Шаг 11

Конец: x = 2 y = 0

Начало: x = 3 y = 0

Удаляем: x = 4 y = 0

Шаг 12

Конец: x = 2 y = 2

Начало: x = 2 y = 3

Удаляем: x = 2 y = 4

Шаг 13

Начало: x = 0 y = 4

Удаляем: x = 1 y = 4

Конец: x = 2 y = 4

Шаг 14

Конец: x = 2 y = 3

Начало: x = 2 y = 4

Удаляем: x = 2 y = 5

Шаг 15

Конец: x = 2 y = 1

Начало: x = 2 y = 2

Удаляем: x = 2 y = 3

Шаг 16

Начало: x = 2 y = 0

Удаляем: x = 2 y = 1

Конец: x = 2 y = 2

Шаг 17

Конец: x = 4 y = 1

Начало: x = 4 y = 2

Удаляем: x = 4 y = 3

Шаг 18

Начало: x = 2 y = 2

Удаляем: x = 3 y = 2

Конец: x = 4 y = 2

Шаг 19

Конец: x = 3 y = 2

Начало: x = 4 y = 2

Удаляем: x = 5 y = 2

Шаг 20

Конец: x = 4 y = 3

Начало: x = 5 y = 3

Удаляем: x = 6 y = 3

Шаг 21

Конец: x = 4 y = 2

Начало: x = 4 y = 3

Удаляем: x = 4 y = 4

Шаг 22

Начало: x = 3 y = 2

Удаляем: x = 4 y = 2

Конец: x = 5 y = 2

Шаг 23

Конец: x = 4 y = 2

Начало: x = 5 y = 2

Удаляем: x = 6 y = 2

Шаг 24

Начало: x = 4 y = 1

Удаляем: x = 4 y = 2

Конец: x = 4 y = 3

Шаг 25

Конец: x = 4 y = 4

Начало: x = 4 y = 5

Удаляем: x = 4 y = 6

Шаг 26

Начало: x = 2 y = 6

Удаляем: x = 3 y = 6

Конец: x = 4 y = 6

Шаг 27

Начало: x = 4 y = 3

Удаляем: x = 4 y = 4

Конец: x = 4 y = 5

Шаг 28

Конец: x = 4 y = 4

Начало: x = 5 y = 4

Удаляем: x = 6 y = 4

Шаг 29

Начало: x = 3 y = 4

Удаляем: x = 4 y = 4

Конец: x = 5 y = 4

Шаг 30

Конец: x = 4 y = 4

Начало: x = 4 y = 5

Удаляем: x = 4 y = 6

Шаг 31

Конец: x = 3 y = 4

Начало: x = 4 y = 4

Удаляем: x = 5 y = 4

Шаг 32

Конец: x = 3 y = 3

Начало: x = 3 y = 4

Удаляем: x = 3 y = 5

2 случай:

Шаг 2

Начало: x = 1 y = 3

Удаляем: x = 2 y = 3

Конец: x = 3 y = 3

Шаг 3

Начало: x = 2 y = 1

Удаляем: x = 2 y = 2

Конец: x = 2 y = 3

Шаг 4

Начало: x = 0 y = 2

Удаляем: x = 1 y = 2

Конец: x = 2 y = 2

Шаг 5

Конец: x = 1 y = 2

Начало: x = 2 y = 2

Удаляем: x = 3 y = 2

Шаг 6

Конец: x = 1 y = 3

Начало: x = 2 y = 3

Удаляем: x = 3 y = 3

Шаг 7

Начало: x = 0 y = 3

Удаляем: x = 1 y = 3

Конец: x = 2 y = 3

Шаг 8

Конец: x = 2 y = 1

Начало: x = 3 y = 1

Удаляем: x = 4 y = 1

Шаг 9

Начало: x = 2 y = 0

Удаляем: x = 2 y = 1

Конец: x = 2 y = 2

Шаг 10

Начало: x = 1 y = 2

Удаляем: x = 2 y = 2

Конец: x = 3 y = 2

Шаг 11

Конец: x = 2 y = 0

Начало: x = 3 y = 0

Удаляем: x = 4 y = 0

Шаг 12

Конец: x = 2 y = 2

Начало: x = 2 y = 3

Удаляем: x = 2 y = 4

Шаг 13

Начало: x = 0 y = 4

Удаляем: x = 1 y = 4

Конец: x = 2 y = 4

Шаг 14

Конец: x = 2 y = 3

Начало: x = 2 y = 4

Удаляем: x = 2 y = 5

Шаг 15

Конец: x = 2 y = 1

Начало: x = 2 y = 2

Удаляем: x = 2 y = 3

Шаг 16

Начало: x = 2 y = 0

Удаляем: x = 2 y = 1

Конец: x = 2 y = 2

Шаг 17

Конец: x = 4 y = 1

Начало: x = 4 y = 2

Удаляем: x = 4 y = 3

Шаг 18

Начало: x = 2 y = 2

Удаляем: x = 3 y = 2

Конец: x = 4 y = 2

Шаг 19

Конец: x = 3 y = 2

Начало: x = 4 y = 2

Удаляем: x = 5 y = 2

Шаг 20

Конец: x = 4 y = 3

Начало: x = 5 y = 3

Удаляем: x = 6 y = 3

Шаг 21

Конец: x = 4 y = 2

Начало: x = 4 y = 3

Удаляем: x = 4 y = 4

Шаг 22

Начало: x = 3 y = 2

Удаляем: x = 4 y = 2

Конец: x = 5 y = 2

Шаг 23

Конец: x = 4 y = 2

Начало: x = 5 y = 2

Удаляем: x = 6 y = 2

Шаг 24

Начало: x = 4 y = 1

Удаляем: x = 4 y = 2

Конец: x = 4 y = 3

Шаг 25

Конец: x = 4 y = 4

Начало: x = 4 y = 5

Удаляем: x = 4 y = 6

Шаг 26

Начало: x = 2 y = 6

Удаляем: x = 3 y = 6

Конец: x = 4 y = 6

Шаг 27

Начало: x = 4 y = 3

Удаляем: x = 4 y = 4

Конец: x = 4 y = 5

Шаг 28

Конец: x = 4 y = 4

Начало: x = 5 y = 4

Удаляем: x = 6 y = 4

Шаг 29

Начало: x = 3 y = 4

Удаляем: x = 4 y = 4

Конец: x = 5 y = 4

Шаг 30

Конец: x = 4 y = 4

Начало: x = 4 y = 5

Удаляем: x = 4 y = 6

Шаг 31

Конец: x = 3 y = 4

Начало: x = 4 y = 4

Удаляем: x = 5 y = 4

Шаг 32

Конец: x = 3 y = 3

Начало: x = 3 y = 4

Удаляем: x = 3 y = 5

**Вывод:**

В данной лабораторной работе мы познакомились алгоритмом перебора решений «Backtracking».

**ИСХОДНЫЙ КОД.**

#include "stdafx.h"

#include<iostream>

#include<memory.h>

#include<time.h>

#include<conio.h>

#include<bitset>

#include<windows.h>

#include<fstream>

using namespace std;

int \*Arr; // 2^30 bytes, 2^28 ints

int L = 1 << 28;

long long start = 0x1fffeffffLL;

long long start2 = 0x1fffeffffLL;

int fld[49] = { -1,-1,0,1,2,-1,-1, -1,-1,3,4,5,-1,-1, 6,7,8,9,10,11,12, 13,14,15,16,17,18,19, 20,21,22,23,24,25,26, -1,-1,27,28,29,-1,-1, -1,-1,30,31,32,-1,-1 };

//создаем поле,которое дано по условию

int nsteps = 0;//шаг

long long Steps[76][2];

void AddStep(int a, int b, int c)

{

if (a < 0 || b < 0 || c < 0) return;

Steps[nsteps][0] = (1LL << a) + (1LL << b);

Steps[nsteps][1] = (1LL << c);

Steps[nsteps + 1][0] = (1LL << c) + (1LL << b);

Steps[nsteps + 1][1] = (1LL << a);

nsteps += 2;

}

void InitSteps()

{

for (int i = 0; i < 49; i++)

{

if ((i % 7) < 5) AddStep(fld[i], fld[i + 1], fld[i + 2]);

if (i < 35) AddStep(fld[i], fld[i + 7], fld[i + 14]);

}

}

//Массив содержит последовательность ходов приводящих к выигрышу

int Solution[33];

int LS = 0;//переменная для поиска решения

//long h = 0;

bool Solve(long long s)//функция поиска решений

{

Arr[s >> 5] = 1;

if (s == 0x000010000LL) return true;

for (int k = 0; k < nsteps; k++)

{

long long u = Steps[k][0], v = Steps[k][1];

if ((u & ~s) == 0 && (v & s) == 0)

{

long long s1 = (s & ~u) | v;

if (Arr[s1 >> 5] == 0)

{

if (Solve(s1))

{

Solution[LS++] = k;

return true;

}

}

}

}

return false;

}

int A[7][7] = { 0 };

void create\_A(long long t)//генерация на основании массива А

{

A[0][0] = A[0][1] = A[1][0] = A[1][1] = A[0][5] = A[0][6] = A[1][5] = A[1][6] =

A[5][0] = A[5][1] = A[6][0] = A[6][1] = A[5][5] = A[5][6] = A[6][5] = A[6][6] = -1;

if ((t >> 32) == 1) A[0][2] = 1;

unsigned int dec = t;

int x = 0, y = 3;

for (int i = 0; i < 32; i++, dec <<= 1)

{

if (i == 2 || i == 26 || i == 29) { y = 2; x++; }

else if (i == 5 || i == 12 || i == 19) { y = 0; x++; }

if ((48 + (dec >> 31)) == 49) A[x][y] = 1;

y++;

}

}

void print()//замена

{

for (int i = 0; i < 7; i++)

{

cout << i << "| ";

for (int j = 0; j < 7; j++)

{

if (A[i][j] == 0) cout << "O";

else if (A[i][j] == 1) cout << "X";

else cout << " ";

}

cout << endl;

}

cout << " ------------" << endl << " ";

for (int i = 0; i < 10; i++)

cout << i;

cout << endl;

}

void save(char file[])//сохранение файла

{

ofstream out(file);

for (int m = LS; --m >= 0; )

{

int k = Solution[m];

long long u = Steps[k][0], v = Steps[k][1];

for (int i = 0; i < 33; i++)

if (u & 1LL << i)

out << i << " ";

for (int i = 0; i < 33; i++)

if (v & 1LL << i)

out << " " << i << endl;

}

}

void animation(char file[])//перемещение

{

ifstream infile(file);

int xran = 0;

int x, y, z, q = 0;

cout << "Шаг " << 1 << endl;

print();

Sleep(1500);//время ожидания

//system( "cls" );

for (int i = 0; i < 31; i++)

{

cout << "Шаг " << i + 2 << endl;

infile >> x >> y;

infile >> z;

for (int j = 0; j < 7; j++)

{

for (int k = 0; k < 7; k++)

{

if (A[j][k] != -1)

{

if (x == q || y == q) {

A[j][k] = 0;

if (xran == 0) {

cout << "Начало: x = ";

cout << j << " y = " << k << endl;

xran++;

}

else {

cout << "Удаляем: x = ";

cout << j << " y = " << k << endl;

xran = 0;

}

}

if (z == q) {

A[j][k] = 1;

cout << "Конец: x = ";

cout << j << " y = " << k << endl;

}

q++;

}

}

}

q = 0;

print();

// getch();

Sleep(1500);

if (i != 30)

{

// system( "cls" );

}

}

}

void zeroing()//обнуляем матрицу для след.запусков

{

for (int i = 0; i < 7; i++)

{

for (int j = 0; j < 7; j++)

{

if (A[i][j] != -1)

A[i][j] = 0;

}

}

}

void main()

{

setlocale(LC\_ALL, "rus");

srand((int)time(0));

InitSteps();

// create\_A( start2 );

while (true)

{

system("cls");

create\_A(start2);

cout << "Выделяем память под массив...\n";

Arr = new int[L];

memset(Arr, 0, L \* 4);

cout << "Начинаем поиск решения...\n";

long c = clock();

Solve(start2);

cout << "Решение найдено за " << (float)(clock() - c) / 1000 << " секунд \n\n";

Sleep(3000);

save("out.txt");

system("cls");

animation("out.txt");

LS = 0;

delete[] Arr;

system("pause");

}

system("pause");

}