**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №**1

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: Поиск с возвратом**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Птухов Д.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2019

## Цель работы.

Построение и анализ алгоритма бэктрекинг (поиск с возвратом) на основе решения задачи квадратирования квадратов.

**Вариант 3р.**

## Основные теоретические положения.

Размер столешницы - одно целое число N  (2 <= N <= 20). Необходимо найти и вывести: одно число K, задающее минимальное количество обрезков(квадратов), из которых можно построить столешницу(квадрат) заданного размера *N*. Далее должны идти *K* строк, каждая из которых должна содержать три целых числа *x*, *y* и *w*, задающие координаты левого верхнего угла и длину стороны соответствующего обрезка(квадрата).

## Описание алгоритма.

Для решения поставленной задачи был использован бэктрекинг – алгоритм, осуществляющий перебор всех возможных вариантов. Для данной задачи была реализована функция get, которая осуществляет следующую последовательность действий:

1. Находит первую свободную ячейку в матрице.
2. Пытается вставить на ее место квадрат со стороной 1, 2, …
3. Зарисовывает квадрат с ранее найденной стороной.
4. При возникновении пересечений и/или других ошибок выполнение передается функции вызывающей данную.
5. Записывает результат в массив min\_res.

## Описание основных структур данных и функций.

Рекурсивная функция get – принимающая одномерный массив, соответствующий текущей зарисовке квадрата, одномерный массив tmp\_res, хранящий данные о квадратах уже поучаствовавших в текущей зарисовке, массив min\_res, хранящий в себе данные о текущей минимальной зарисовке всего квадрата, параметр n – длину стороны закрашиваемого квадрата и параметр buf, отвечающий за шаг увеличения длины для рассматриваемого квадрата.

**Тестирование.**

Таблица 1 – Результаты тестирование

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 5 | 8  3 3 2  0 0 2  2 0 3  0 2 1  0 3 1  0 4 1  1 2 1  1 3 2  Iter cnt: 332  Time: 0 |
| 9 | 6  0 0 3  0 3 3  0 6 3  3 0 3  3 3 6  6 0 3  Iter cnt: 1984  Time: 0 |
| 13 | 11  7 7 6  0 0 6  6 0 7  0 6 2  0 8 2  0 10 3  2 6 3  2 9 1  3 9 4  5 6 1  5 7 2  Iter cnt: 250424  Time: 0.002 |
| 15 | 6  0 0 5  0 5 5  0 10 5  5 0 5  5 5 10  10 0 5  Iter cnt: 5296  Time: 0 |
| 17 | 12  9 9 8  0 0 8  8 0 9  0 8 2  0 10 2  0 12 5  2 8 4  5 12 1  5 13 4  6 8 2  6 10 3  8 9 1  Iter cnt: 4155001  Time: 0.033 |

# Вывод.

В ходе работы был построен и анализирован алгоритм бэктрекинг на основе решения задачи квадратирования квадратов. Исходный код программы представлен в приложении А.

**ПРИЛОЖЕНИЕ А.  
ИСХОДНЫЙ КОД**

#include <iostream>

#include <vector>

#include <algorithm>

#include <ctime>

#include <algorithm>

struct kv

{

int x;

int y;

int len;

};

void get(std::vector<int>& a, std::vector<kv>& tmp\_res, std::vector<kv>& min\_res, int n, int buf)

{

kv p;

bool f = false;

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

{

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

if (a[i\*n + j] == 0)

{

p.x = i;

p.y = j;

f = true;

break;

}

if (f)

break;

}

if (f == false)

{

min\_res = tmp\_res;

return;

}

for (int tmp\_len = buf; tmp\_len < n; tmp\_len += buf)

{

if (p.x + tmp\_len > n || p.y + tmp\_len > n)

return;

if (tmp\_res.size() + 1 >= min\_res.size())

return;

for (int i = p.x; i < p.x + tmp\_len; ++i)

for (int j = p.y; j < p.y + tmp\_len; ++j)

if (a[i\*n + j] == 1)

return;

for (int i = p.x; i < p.x + tmp\_len; ++i)

for (int j = p.y; j < p.y + tmp\_len; ++j)

a[i\*n + j] = 1;

p.len = tmp\_len;

tmp\_res.push\_back(p);

get(a, tmp\_res, min\_res, n, buf);

for (int i = p.x; i < p.x + tmp\_len; ++i)

for (int j = p.y; j < p.y + tmp\_len; ++j)

a[i\*n + j] = 0;

tmp\_res.pop\_back();

}

}

int check(int n)

{

int ans = 1;

for (int i = 2; i < n; ++i)

{

if (n % i == 0)

ans = i;

}

return ans;

}

int main()

{

int n = 0;

std::cin >> n;

int res = check(n);

std::vector<kv> tmp\_res;

std::vector<kv> min\_res(n \* 2 + 1);

std::vector<int> a(n\*n, 0);

if (res == 1 && n != 2)

{

for (int i = 0; i < n / 2; ++i)

for (int j = 0; j < n / 2; ++j)

a[i \* n + j] = 1;

for (int i = n / 2; i < n; ++i)

for (int j = 0; j < n / 2 + 1; j++)

a[i \* n + j] = 1;

tmp\_res.push\_back({0, 0, n / 2});

tmp\_res.push\_back({n / 2, 0, n / 2 + 1});

}

get(a, tmp\_res, min\_res, n, res);

std::cout << min\_res.size() << "\n";

for (auto i : min\_res)

std::cout << i.x << " " << i.y << " " << i.len << "\n";

return 0;

}