

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе № 1
по дисциплине «Построение и анализ алгоритмов»
Тема: Поиск с возвратом.

Студент гр.7304

Сергеев И.Д.

Преподаватель

Филатов А.Ю.

Дата выполнения работы

17.02.2019

Санкт-Петербург

2019

1. Постановка задачи

1.1. Цель работы

Исследование алгоритмов поиска с возвратом, реализация программы заполнения квадрата минимальным количеством квадратов.

1.2. Формулировка задачи

У Вовы много квадратных обрезков доски. Их стороны (размер) изменяются от 1 до $N-1$, и у него есть неограниченное число обрезков любого размера. Но ему очень хочется получить большую столешницу - квадрат размера NN . Он может получить ее, собрав из уже имеющихся обрезков(квадратов).

Например, столешница размера 7×7 может быть построена из 9 обрезков.

Внутри столешницы не должно быть пустот, обрезки не должны выходить за пределы столешницы и не должны перекрываться. Кроме того, Вова хочет использовать минимально возможное число обрезков.

Входные данные

Размер столешницы - одно целое число NN ($2 \leq N \leq 20$).

Выходные данные

Одно число KK , задающее минимальное количество обрезков(квадратов), из которых можно построить столешницу(квадрат) заданного размера NN . Далее должны идти KK строк, каждая из которых должна содержать три целых числа x , y и w , задающие координаты левого верхнего угла ($1 \leq x, y \leq N$) и длину стороны соответствующего обрезка(квадрата).

2. Ход работы

2.1. Была написана программа на языке `c++`, реализующая алгоритм заполнения. Были использованы функции:

- 1) Для квадратов с четными сторонами - `even_quad`
- 2) Для квадратов с нечетными – `simple_quad`

2.2. Код программы:

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cstdlib>

using namespace std;

void even_quad(int size,int x,int y){//Функция для четных квадратов
    cout << "4" << endl;
    cout << x << " " << y << " " << size << endl;
    cout << x+size << " " << y << " " << size << endl;
    cout << x << " " << y+size << " " << size << endl;
    cout << x+size << " " << y+size << " " << size << endl;
}
```

```

void simple_quad(int** arr,int size,int n,int x,int y, int &minimal,int c,int* output,int
&exit_flag){//Функция для остальных квадратов
    if (exit_flag == 1) //Флаг для быстрого выхода, когда нашли минимальную комбинацию
        return;
    if (c == minimal) //Если кол-во итераций превысило минимум, выходим
        return;
    if (x+size > n || y+size > n) //если подаем в функцию, который не поместиться в область, уходим
        return;
    for (int j = y;j<y+size;j++) //Если в зоне для закраски есть другой квадрат, то выходим
        if (arr[x][j] != 0)
            return;
    c++; //увеличиваем счетчик кол-ва квадратов и индекс закраски
    for (int i = x;i<x+size;i++)
        for (int j = y;j<y+size;j++)
            arr[i][j] = c; //Закраска квадрата
    int x1 = x,y1 = y; //запоминаем координаты, чтобы при неудаче стереть закрашенные значения
    int flag = 1; //Флаг нахождения места для очередного квадрата
    for (int i=0;i<n;i++){
        if (flag == 0)
            break;
        for (int j=0;j<n;j++){
            if (arr[i][j] == 0) //Если находим пустое место то запоминаем координаты, идем по квадрату
слева направо сверху вниз
                x = i;
                y = j;
                flag = 0;
                break;
        }
        if (i == n-1 && j == n-1) { //условие полной закраски квадрата
            if (minimal == c) { //если счетчик равен минимальному, то нужная комбинация
                int count = 0,numb = 1,ind = 0;
                for (int i=0;i<n;i++){ //заполнение данных в массив (индексы углов и размеры квадратов)
                    for (int j=0;j<n;j++){
                        if (arr[i][j] == numb){
                            output[ind++] = j;
                            output[ind++] = i;
                            int z = j;
                            while(arr[i][z]==numb){
                                count++;
                                z++;
                            }
                            output[ind++] = count;
                            count = 0;
                            numb++;
                        }
                    }
                }
                exit_flag = 1; //так как нашли комбинацию выходим из рекурсии
                flag = 0;
                break;
            }
        }
    }
}

int quad_size = min(n-x,n-y); //определяем максимальный квадрат для вставки, чтобы сокращать
ко-во рекурсивных вызовов
if ((x == (n+1)/2 && y == 0) || (x == 0 && y == (n+1)/2)) //если мы находимся на этих координатах,
то берем только максимальный квадрат(стратегия)
    simple_quad(arr,quad_size,n,x,y,minimal,c,output,exit_flag);
else for (int i=quad_size;i>0;i--){ //иначе идем в цикл с разными размерами квадратов
    simple_quad(arr,i,n,x,y,minimal,c,output,exit_flag);
}
}

```

```

        for (int i = x1; i < x1 + size; i++) //очистка неправильных квадратов, чтобы массив был правильный
            for (int j = y1; j < y1 + size; j++)
                arr[i][j] = 0;
    }
    int main() {
        int key[] = {0,6,8,9,6,11,11,6,12,13,6,13,8,6,15,15,6,8,15,6}; //массив минимальных значений для
        нечетных чисел
        int n;
        cin >> n; //ввод числа
        int size, c = 0, exit_flag = 0;
        int min = key[(n-1)/2]; //выбор минимума
        int x = 0, y = 0;
        int* output = new int[key[(n-1)/2]*3]; //массив чисел для вывода
        int** arr = new int*[n]; //массив чисел для квадрата
        for (int i = 0; i < n; i++) {
            arr[i] = new int[n];
            for (int j = 0; j < n; j++)
                arr[i][j] = 0;
        }
        if (n < 2 || n > 40) { //проверка на некорректный выбор
            cout << "Wrong size" << endl;
            return 0;
        }
        if (n % 2 != 0) { //если нечетное
            if (n % 3 == 0) //если число кратно 3 выбираем число n так, чтобы было минимальное кол-во
            квадратов
                size = 2 * n / 3;
            else if (n % 5 == 0) //если число кратно 5 выбираем число n так, чтобы было минимальное кол-во
            квадратов
                size = 3 * n / 5;
            else size = (n + 1) / 2; //иначе выбираем стандартно
            simple_quad(arr, size, n, x, y, min, c, output, exit_flag);
            cout << min << endl;
            for (int i = 0; i < key[(n-1)/2]*3; i++) { //вывод информации
                cout << output[i+1]+1 << " " << output[i+1] << " " << output[i+2] << endl;
                i += 2;
            }
        }
        else { //если четное
            size = n / 2;
            even_quad(size, x+1, y+1);
        }

        delete [] output; //очистка памяти
        for (int i = 0; i < n; i++)
            delete[] arr[i];
        delete[] arr;
        return 0;
    }
}

```

В результате работы программы был исследован алгоритм работы поиска с возвратом при помощи рекурсии. Также была написана программа, реализующая заполнение квадратной области минимальным количеством квадратов.