

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

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Построение и Анализ Алгоритмов»
Тема: «Перебор с возвратом»

Студент гр. 7304

Петруненко Д.А

Преподаватель

Филатов А.Ю.

Санкт-Петербург

2019

Содержание.

1. Цель работы.

2. Задания лабораторной работы

3. Основные теоретические положения.

4. Ход работы

5. Вывод

Цель работы:

Ознакомится с алгоритмом перебора с возвратом, а затем написать программу с использованием данного алгоритма.

Задания лабораторной работы

Дан квадрат размера N на N , заполнить данный квадрат наименьшим числом квадратов меньшего размера от $N-1$ до 1 .

Основные теоретические положения.

Решение задачи методом поиска с возвратом сводится к последовательному расширению частичного решения. Если на очередном шаге такое расширение провести не удастся, то возвращаются к более короткому частичному решению и продолжают поиск дальше. Данный алгоритм позволяет найти все решения поставленной задачи, если они существуют. Для ускорения метода стараются вычисления организовать таким образом, чтобы как можно раньше выявлять заведомо неподходящие варианты. Зачастую это позволяет значительно уменьшить время нахождения решения.

Ход работы:

1. Для квадратов 2, 3, 5 и кратных для этих размерностей квадратов были написаны и используются отдельные функции:

`void multiple5(int N);`

`void multiple3(int N);`

`void multiple2(int N);`

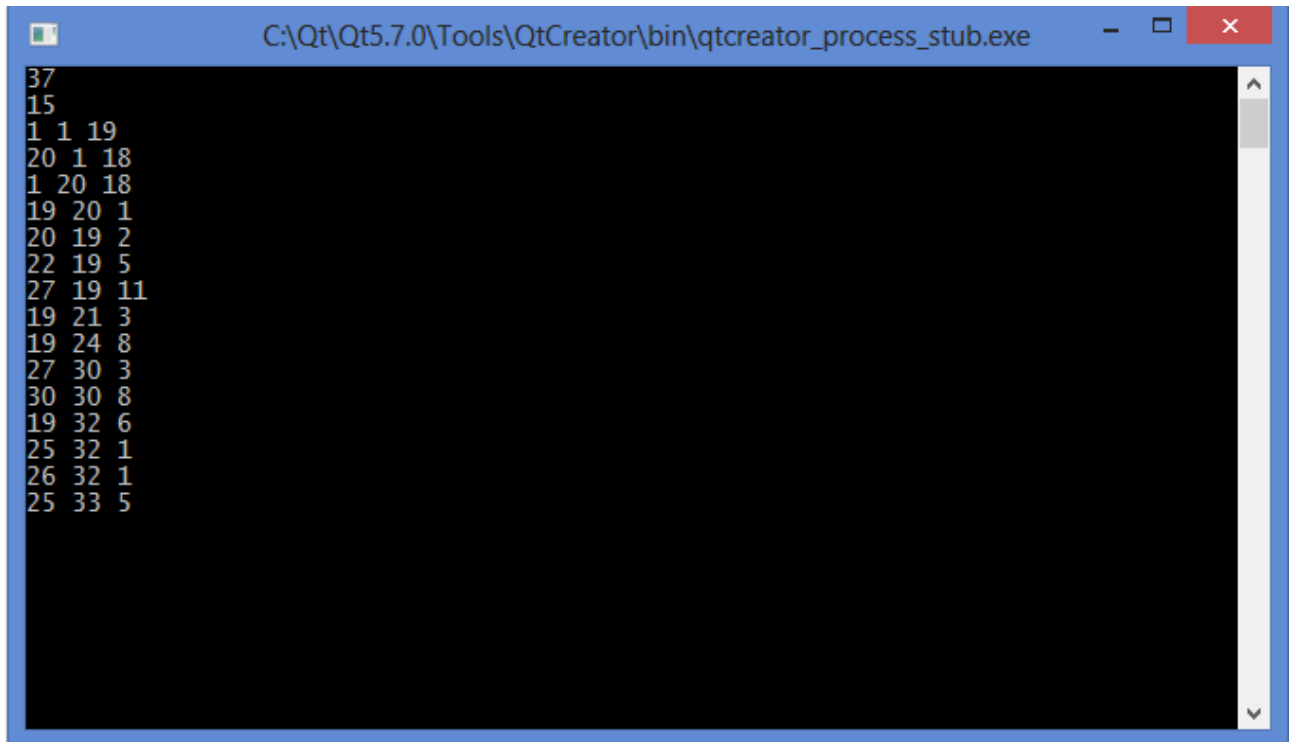
2. Для остальных размерностей используется класс `simpleNum`, в котором вычисляется минимальное количество меньших квадратов для простых чисел, в котором с помощью метода `void find(int x, int y, int min)`, который является основным методом для алгоритма перебора с возвратом и вспомогательных методов `void cleanSQ(int **tmp, int x, int y)` – очистка квадратов при возвращении алгоритма назад, `void Zero(int &x, int &y, int min)` – поиск ещё не заполненных мест в матрице, `void WriteB(int min)` – за данных для минимального случая на данном шаге алгоритма.

3. Функция `Main` выполняет считывание размерности квадрата `N` и вызов необходимых функций для заданного `N`.

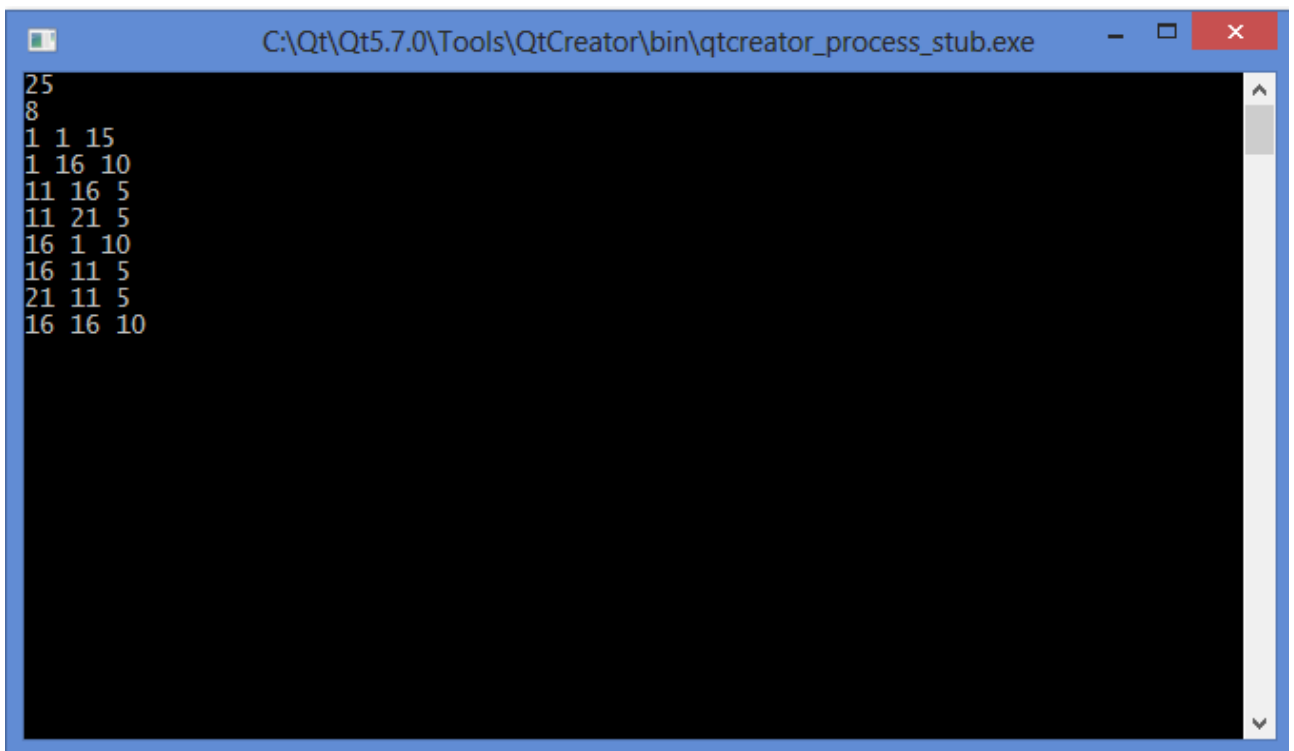
Структуры данных:

Двумерные массивы для хранения промежуточного массива и массива минимальных данных.

Результат работы:



```
C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
37
15
1 1 19
20 1 18
1 20 18
19 20 1
20 19 2
22 19 5
27 19 11
19 21 3
19 24 8
27 30 3
30 30 8
19 32 6
25 32 1
26 32 1
25 33 5
```



```
C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
25
8
1 1 15
1 16 10
11 16 5
11 21 5
16 1 10
16 11 5
21 11 5
16 16 10
```

Исходный код:

```
#include <iostream>

using namespace std;

class simpleNum{
public:
    simpleNum(const int Num = 0) : N(Num), best(0), Best_mas(NULL){
        Arr = new int*[(N/2)+1];
        for(int i=0;i<(N/2)+1;i++){
            Arr[i] = new int[(N/2)+1];
            for(int j=0;j<(N/2)+1;j++){
                Arr[i][j] = 0;}
            Arr[0][0] = 1;
            Arr[1][0] = 1;}

    void cleanSQ(int **tmp, int x, int y){
        int n = tmp[y][x];
        for(int i=y;i<n+y;i++){
            for(int j=x;j<n+x;j++){
                tmp[i][j] = 0;}

        void Zero(int &x, int &y, int min){
            while(Arr[y][x]!=0)
                if((x+1)<(N/2)+1) x++;
                else if(x==y) { WriteB(min);
                    break;}
                else{ x=0; if((y+1)<(N/2)+1) y++;}
        }

        void WriteB(int min){
            if(best==0) best = min;
            if(best>min){ best = min;
                if(Best_mas!=NULL) del(Best_mas, best);
                Best_mas = new int*[best];
```

```

    for(int i=0;i<best;i++)
        Best_mas[i] = new int[3];
    int **tmp = new int*[(N/2)+1];
    for(int i=0;i<(N/2)+1;i++){
        tmp[i] = new int[(N/2)+1];
        for(int j=0;j<(N/2)+1;j++){
            tmp[i][j] = Arr[i][j];}
    tmp[0][0] = 0;
    tmp[1][0] = 0;
    int k=0;
    for(int i=0;i<(N/2)+1;i++){
        for(int j=0;j<(N/2)+1;j++){
            if(tmp[i][j]!=0){
                Best_mas[k][0] = j;
                Best_mas[k][1] = i;
                Best_mas[k][2] = tmp[i][j];
                cleanSQ(tmp, j, i);
                k++;
                if(k==best) break;
            }
            if(k==best) break;
        }
        del(tmp, (N/2)+1);}
}

void find(int x, int y, int min) {
    if(best!=0&&best<=min) return;
    if(min>=(2*N-1)) return;
    if((x>(N/2)+1)||y>(N/2)+1) return;
    if(Arr[y][x]!=0) {
        Zero(x, y, min);
        if(x==y) {WriteB(min); return;}
    }
    int size = 0;
    while(((x+size)<(N/2)+1)&&Arr[y][x+size]==0){

```

```

        if((N/2)+1-y<size+1) break;
        size++;}
    int last_X = x, last_Y = y;
    for(int k=size;k>=1;k--){
        for(int i=last_Y;i<size+last_Y;i++)
            for(int j=last_X;j<size+last_X;j++)
                Arr[i][j] = size;
        ++min;
        x = 0; y = 0;
        Zero(x, y, min);
        find(x, y, min);
        cleanSQ(Arr, last_X, last_Y);
        min--;
        size--;
    }
}

```

```

void print(){
    cout<<"1 "<<"1 "<<(N/2)+1<<endl;
    cout<<1+(N/2)+1<<" 1 "<<N/2<<endl;
    cout<<"1 "<<1+(N/2)+1<<" "<<N/2<<endl;
    cout<<1+(N/2)<<" "<<1+(N/2)+1<<" 1"<<endl;
    for(int i=0;i<best;i++)
        cout<<Best_mas[i][0]+(N/2)+1<<"          "<<Best_mas[i][1]+(N/2)+1<<"
"<<Best_mas[i][2]<<endl;
}

```

```

void del(int ** Del_mas, int n){
    for(int i=0;i<n;i++)
        delete [] Del_mas[i];
    delete [] Del_mas;}

```

```

~simpleNum(){
    del(Arr, (N/2)+1);
    del(Best_mas, best);
}

```

```

    }
    int Best(){return best;}

```

private:

```

    int ** Best_mas;
    int best;
    int N;
    int ** Arr;
};

```

```

void multiple5(int N){
    int min = 8, i = 1;
    cout<<min<<endl;
    cout<<i<<" "<<i<<" "<<3*N/5<<endl;
    cout<<i<<" "<<i+3*N/5<<" "<<2*N/5<<endl;
    cout<<i+2*N/5<<" "<<i+3*N/5<<" "<<N/5<<endl;
    cout<<i+2*N/5<<" "<<i+4*N/5<<" "<<N/5<<endl;
    cout<<i+3*N/5<<" "<<i<<" "<<2*N/5<<endl;
    cout<<i+3*N/5<<" "<<i+2*N/5<<" "<<N/5<<endl;
    cout<<i+4*N/5<<" "<<i+2*N/5<<" "<<N/5<<endl;
    cout<<i+3*N/5<<" "<<i+3*N/5<<" "<<2*N/5<<endl;}

```

```

void multiple3(int N){
    int min = 6, i = 1;
    cout<<min<<endl;
    cout<<i<<" "<<i<<" "<<2*N/3<<endl;
    cout<<i<<" "<<i+2*N/3<<" "<<N/3<<endl;
    cout<<i+N/3<<" "<<i+2*N/3<<" "<<N/3<<endl;
    cout<<i+2*N/3<<" "<<i<<" "<<N/3<<endl;
    cout<<i+2*N/3<<" "<<i+N/3<<" "<<N/3<<endl;
    cout<<i+2*N/3<<" "<<i+2*N/3<<" "<<N/3<<endl;}

```

```

void multiple2(int N){
    int min = 4, i = 1;
    cout<<min<<endl;

```



```

cout<<i<<" "<<i<<" "<<N/2<<endl;
cout<<i<<" "<<i+N/2<<" "<<N/2<<endl;
cout<<i+N/2<<" "<<i<<" "<<N/2<<endl;
cout<<i+N/2<<" "<<i+N/2<<" "<<N/2<<endl;}

int main(){
    int N;
    cin>>N;
    if(N>=2&&N<=40){
        if(N%2==0) multiple2(N);
        else if(N%3==0) multiple3(N);
            else if(N%5==0) multiple5(N);
                else{
                    simpleNum A(N);
                    A.find(1, 0, 0);
                    cout<<A.Best()+4<<endl;
                    A.print();}
                }else cout<<"Wrong N!"<<endl;
        return 0;
    }
}

```

Вывод:

В ходе данной лабораторной работы произошло ознакомление с алгоритмом перебор с возвратом, а затем была написана программа, которая производит, для квадрата размерности N на N , поиск минимального количества квадратов меньшего размера от 1 до $N-1$ с использованием алгоритма перебора с возвратом.