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: ПЕРЕБОР С ВОЗВРАТОМ.

Студент гр. 7304 Моторин Е.В.

Преподаватель Филатов А.Ю.

Санкт-Петербург

2019

Цель работы:

Ознакомиться с алгоритмом перебора с возвратом, получить навыки его программирования и применения на языке программирования C++.

Задача:

Разработать программу, которая делит квадрат со стороной N на квадраты со стороной не более, чем N-1. Количество получившихся квадратов должно быть наименьшим из всех возможных вариантов разбиения.

Основные теоретические положения:

Поиск с возвратом, бэктрекинг (англ. *backtracking*) —
общий метод нахождения решений задачи, в которой требуется полный перебор всех возможных вариантов в некотором множестве M. Как правило позволяет решать задачи, в которых ставятся вопросы типа: «Перечислите все возможные варианты …», «Сколько существует способов …», «Есть ли способ …», «Существует ли объект…» и т. п.

Решение задачи методом поиска с возвратом сводится к последовательному расширению частичного решения. Если на очередном шаге такое расширение провести не удается, 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.

Метод поиска с возвратом является универсальным. Достаточно легко проектировать и программировать алгоритмы решения задач с использованием этого метода. Однако время нахождения решения может быть очень велико даже при небольших размерностях задачи (количестве исходных данных), причём настолько велико (может составлять годы или даже века), что о практическом применении не может быть и речи. Поэтому при проектировании таких алгоритмов, обязательно нужно теоретически оценивать время их работы на конкретных данных. Существуют также задачи выбора, для решения которых можно построить уникальные, «быстрые» алгоритмы, позволяющие быстро получить решение даже при больших размерностях задачи. Метод поиска с возвратом в таких задачах применять неэффективно.

Ход работы:

- Реализован class Square, который содержит все методы необходимые для решения задачи.

- Реализована логика решения задачи для квадратов с четными сторонами. (x ,y – начало координат, size – длина стороны)

```
void evenSide(int size,int x,int y){
  printf("4\n%d %d %d\n", x, y, size);
  printf("%d %d %d\n", x+size, y, size);
  printf("%d %d %d\n", x, y+size, size);
  printf("%d %d %d\n", x+size, y+size, size);
}
```

- Реализована рекурсивная функция, применимая для квадратов с длинами сторон не кратными двум.

```
inline void searchSubSquares(int** SQUARE, //основной квадрат (матрица с
цветами)
                             int subLen, // длина стороны подквадрата
                             int x, int y, // координаты подквадрата
                             int color, // цвет раскраски
                             bool &exitFlag) {

        //условия выхода
        if (exitFlag || color == minCount || x+subLen > sideLen || y+subLen >
sideLen) return;
        for (int j = y; j < y+subLen; j++)
            if (SQUARE[x][j] != 0) return; // если подквадрат уже закрашен

        //сохранение подквадрата
        Data* d = new Data();
        d->x = x+1;
        d->y = y+1;
        d->length = subLen;
        answerArr[color] = d;

        //раскраска
        color++;
        for (int i = x;i<x+subLen;i++)
            for (int j = y; j < y+subLen; j++)
                SQUARE[i][j] = color;


        // поиск начала нового подквадрата / выход
        int k = x,m = y;
        for (struct {int i=0; bool flag = true;} s; s.i<sideLen,
s.flag;s.i++) for (int j=0;j<sideLen;j++) {
                if (SQUARE[s.i][j] == 0){
                    x = s.i;
                    y = j;
                    s.flag = false;
                    break;
                }
```

```
            if (s.i == sideLen-1 && j == sideLen-1){
                if (minCount == color){
                    exitFlag = true;
                    s.flag = false;
                    break;
                }
            }
        }


    int minLen = min(sideLen-x,sideLen-y);
    if ((x == (sideLen+1)/2 && y == 0) || (x == 0 && y == (sideLen+1)/2))
        searchSubSquares(SQUARE,minLen,x,y,color,exitFlag);
    else
        for (int i=minLen;i>0;i--){
            searchSubSquares(SQUARE,i,x,y,color,exitFlag);
    }

    for (int i = k;i<k+subLen;i++)
        for (int j = m;j<m+subLen;j++)
            SQUARE[i][j] = 0;

}
```

Результат:

Из рисунка 1 видно, что разработанная программа выполняет поставленную задачу, а именно: находит минимальное количество ‘подквадратов’ и их положение.

```
Enter side length: 37
15
1 1 19
1 20 18
19 20 2
19 22 5
19 27 11
20 1 18
20 19 1
21 19 3
24 19 8
30 27 3
30 30 8
32 19 6
32 25 1
32 26 1
33 25 5
Program ended with exit code: 0
```

Рисунок 1.

Вывод:

Таким образов, в ходе данной лабораторной работы было подробно изучено написание алгоритма с использованием перебора с возвратом. Полученный результат удовлетворяет задание лабораторной работы.