**aМИНОБРНАУКИ 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. 6381 |  | Cергухин В.Ю. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2018

**Цель работы**

Изучение алгоритма поиска с возвратом на основе задачи минимального заполнения квадрата размером NxN квадратами размера от 1 до N-1.

**Описание алгоритма**

1. На вход подается число - стороны квадрата. Проверяется её делимость на 2, 5 и 3.
2. В случае если число делится на 2, 3 или 5, то используется оптимизация построенная на следующих доводах:

Наименьше число квадратов на которое можно разбить большой квадрат, зависит от наименьшего делителя числа.

Если число не является простым, то его наименьшим делителем является 2, 3 или 5 (поскольку в задачи речь идет о числах до 40).

Для оптимизации алгоритма в этом случае функция бэктрекинга строит разбиение для простого числа являющегося наименьшем делителем исходного числа, а затем, перед выводом получившегося разбиения на экран, все координаты и длинны умножается на коэффицент scale = исходное число / наименьший делитель числа ( за исключением единицы ). Приведенный шаблон оптимизации имеет место быть, т.к. для программы, не использующий этот шаблон, то есть выполняющий бэктрэкинг «c нуля», результат разбиения получается тот же, но программы работает дольше.

1. Далее происходит заполнение квадратом со стороной (n+1)/2 и двумя квадратами со стороной (n-1)/2 в левый верхний, правый верхний и левый нижний углы соответственно (вновь в целях сокращения работы программы). Затем осуществляется вход в рекурсивную функцию для оставшейся незаполненной части. В программе с полным перебором вход в рекурсию осуществляется без этой вставки квадратов. Результаты работы двух программ совпадают, но программу использующая такую ставку работает эффективнее.
2. В рекурсивной функции начинается заполнение пустого места квадратами, начиная с квадратов длиной 1х1 и двигаясь слева направо сверху вниз. Наименьшее число полученных квадратов хранится в отдельной переменной, и как только во время перебора текущее число квадратов становится больше или равно наименьшему, либо заканчиваются пустые ячейки и число квадратов оказывается больше полученного на тот момент наименьшего, решение забывается и делается шаг назад в поисках более оптимального решения. Если же по окончании пустых ячеек квадратов получилось меньше, чем наименьшее их число, полученное на тот момент, программа перезаписывает лучшее решение. В итоге программа получит разделение на наименьшее количество квадратов.

**Описание функций.**

1. **void removeSquare(int\*\* &squareTmp, int color, int x, int y, int n)** – удаление определенного квадрата;
2. **bool putSquare(int\*\* &squareTmp, int color, int size, int x, int y, int n)**– установка квадрата размера size;
3. **bool hasEmptySquare(int\*\* &squareTmp, int& x, int& y, int n)** – поиск незаполненной ячейки;
4. **void backTracking (int& min\_square\_num, int n, int\*\* &squareTmp, int\*\* &squareBest, int x, int y, int& temp\_square) –** Функция выполняющая заполения квадрата с использованием backtracking

**Тестирование**

* Тестирование оптимизированной программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

* Тестирование программы использующей только backtracking.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

Сравнив результаты работы двух программ, нетрудно обнаружить что числа на которое минимально можно разбить квадрат совпадают. Сам способ разбивки может отличаться, как это продемонстрировано выше для квадратов со сторонами 9 и 23. Это объясняется тем, что квадрат может иметь разные минимальные разбивки. Кроме того, при попытках тестировать вторую программу на больших числах, её время работы было значительно больше, чем время работы первой программы на этих же числах. Из этого можно сделать вывод, что использование обоих шаблонов допустимо и значительно улучшает скорость работы программы.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Рисунок 1. Демонстрация результата работы оптимизированной программы для числа 9.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Рисунок 2. Демонстрация результата работы программы, реализующей backtracking «с нуля» для числа 9.

**Вывод**

В процессе выполнения лабораторной работы был реализован алгоритм поиска с возвратом (backtracking) и на его основе была решена задача минимального заполнения квадрата размером NxN.