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 _____ Абдульманов Э.М

Преподаватель _____ Филатов А.Ю

г. Санкт-Петербург
2019

Цель работы:

Изучение алгоритма «backtracking» на примере задачи «Разбиение квадрата N*N наименьшим числом квадратов»

Описание алгоритма:

Используемые структуры данных:

- Struct Cell – является ячейкой матрицы, которая имеет два поля. c_lenght – длина квадрата, которому принадлежит ячейка. c_count – какой по счету поставленный квадрат идет.
- Class Table – является столом, который нужно построить. Имеет такие методы:
  - FindFreeCellForSquare(int &x,int &y,int startX,int startY) – ищет пустую клетку на столе начиная с точки (startX,startY) и найденную пустую клетку записывает в точку (x,y).
  - setSquare(int x,int y,int lenght) – ставит квадрат на стол, начиная с точки (x,y), длинною length.
  - clearSquare(int x,int y,int lenght) - удаляет квадрат на столе, начиная с точки (x,y), длинною length.
  - canSetSquareThisLenght(int x,int y,int lenght) – проверяет можно ли поставить квадрат на стол заданной длинною length, начиная с точки (x,y) и если можно, то вызывает метод setSquare(x,y,length).
  - IsMinSquarePacking() – метод, который вызывается в тот момент, когда весь стол покрыт обрезками и если число обрезков минимальное, то данная конфигурация сохраняется.
  - IsPrime() – проверяет, длина стола является простым числом или нет.
  - Build() – строит стол из наименьшего числа обрезков.
  - startBuildNoPrimeTable(int lenght,int BigDivider) – строит стол, если его длина является составным числом.
  - startBuildPrimeTable(int lenght,int xx,int yy) – строит стол, если его длина является простым числом.
  - writeTable(vector<vector<Cell>> matrix) – выводит данную конфигурацию на экран.

Описание алгоритма:

В самом начале вызывается функция IsPrime(), если число составное то:

- Вызывается функция startBuildNoPrimeTable(int lenght,int BigDivider), которой на выход подается длина квадрата с

которого нужно начинать строить стол, и наибольший делитель длины стола. Рассмотрим работу алгоритма на примере числа 25. Вначале ставится квадрат 20, затем пытаемся поставить квадрат опять со стороной 20, если нельзя, то пробуем поставить квадрат длиной 20-5=15 и так далее. Когда весь стол был покрыт обрезками, мы находим квадрат со стороной больше чем, BigDivider и заменяем его на сторону length- BigDivider и строим опять до конца. И таким образом обходятся все возможные варианты построения стола за исключением тех, что если число обрезков уже больше, чем минимальное, дальше стол не строится и возвращается на последние возможное разветвление.

- Если стол имеет длину, которая является простым числом, то вызывается функция startBuildPrimeTable(int lenght,int xx,int yy). На вход ей подается длина, с которой нужно пытаться построить квадрат и точку (xx,yy), с которой начинается поиск пустого места. Алгоритм работает таким же образом, как и для составного числа за исключением того, что когда стол был построен, мы возвращаем и ищем квадрат, сторона которого больше >1 и когда такой квадрат был найден, он меняется на квадрат стороной на 1 меньше и опять рекурсивно вызывается функция startBuildPrimeTable(int lenght,int xx,int yy).
  Оптимизации:
  1) Если число квадратов на столе уже больше или равно, чем число наименьших квадратов, то мы возвращаемся на последние разветвление.
  2) Задачу можно свести к тому, что число квадратов со стороной 1 на столе должно быть наименьшее. Следовательно, когда была построена конфигурация и если число квадратов на ней наименьшее, то запоминается еще и число единичных квадратов на столе. Если в какой-то момент число единичных квадратов больше, чем наименьшее, то мы возвращаемся на последнее разветвление.
  3) Решение задачи можно начинать с расстановки сразу трех наибольших квадратов. К примеру, квадрат со стороной N=37. Можно разложить на 3 наибольших квадрата N/2+1,N/2 и N/2.
  4) Начинать поиск пустой точки можно не с точки (0,0),а в начале с точки (N/2,N/2). А потом можно начинать поиск с координаты y, которая является координатой y последнего поставленного квадрата
  5) Оптимизация симметричность. Допустим мы нашли наименьшую расстановку в данный момент времени,

тогда перебор симметричной ситуации будет давать те же результаты.

Примеры работы программы

1) N=25

```
8
1 1 15
16 1 10
16 11 10
1 16 10
11 16 5
11 21 5
16 21 5
21 21 5
```

2) N=29

```
14
1 1 15
16 1 14
1 16 14
16 15 2
18 15 5
23 15 7
15 16 1
15 17 3
15 20 3
18 20 3
21 20 2
21 22 1
22 22 8
15 23 7
```

3) N=37

```
15
1 1 19
20 1 18
1 20 18
20 19 2
22 19 5
27 19 11
19 20 1
19 21 3
19 24 8
27 30 3
30 30 8
19 32 6
25 32 1
26 32 1
25 33 5
```

4) N=40

Вывод

В ходе данной лабораторной работы был реализован алгоритм «backtracking» и решена задача квадратирования квадрата стороной N наименьшим числом квадратов, начиная с квадратов со сторонами N-1.