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

ОТЧЕТ

по лабораторной работе №1 (Вар. 4р)
по дисциплине «Построение и анализ алгоритмов»
Тема: Поиск с возвратом

Студент гр. 3388 _____ Раутио И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Задание

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 N
Он может получить ее, собрав из уже имеющихся обрезков(квадратов).
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
($2 \le N \le 20$).

Выходные данные

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

Пример входных данных

7

Соответствующие выходные данные

9
1 1 2
1 3 2
3 1 1
4 1 1
3 2 2
5 1 3
4 4 4
1 5 3
3 4 1

Выполнение работы

Для выполнения работы был использован алгоритм итеративного поиска с возвратом. Это переборный алгоритм применяемый при наличии частичных решение и критериев их сравнения. Т.е. на каждом шаге перебора (в данном случае добавление нового квадрата) происходит сверка с критериями перебора. Если это решение полное — сохраняем его. Если оно частичное — перебираем дальше его потомков. Если оно не подходит под критерии, то его потомки больше не буду перебираться.

Написанное решение работает для натуральных числе от 2 до 20

Работа алгоритма:

1. Вычисление оптимальных сторон квадрата.
2. Расстановка оптимальных квадратов
3. Поиск точки с минимальными координатами для установки квадрата. Рекурсивный вызов функции для всех возможных длин сторон на этой точке.
4. Если найдется хоть одно решение, то программа выводит ответ и завершается.
5. Возврат к шагу 3

Способ хранения частичных решений:

```
struct Square {
   int x, y, side;
};
class Data {
public:
   int area;
   int n;
   int m;
   vector<int> matrix;
   vector<Square> squares;
}
```

Data хранит текущую площадь, псевдо-матрицу (вместо строк числа, которые при работе читаются как двоичные), список квадратов в данном решении.

Оптимизации алгоритма:

1. Подсчет оптимального размера сторон решения исходя из N
2. На каждом шаге выбирается только одна точка, а не все доступные
3. В приоритете проверяются решения с большой площадью
4. При нахождении решения программа завершает работу

Оценка сложности и памяти:

Поиск точки - N^2 операций. Проверка возможности установки квадрата за O(1). Так как квадрат заполняется сверху вниз, а строка проверяется побитовыми операциями. На точку не более N — 1 квадратов. Длина решения линейно зависит от N. Тогда N позиций, поиск места N^2 и вариантов N — 1. => Сложность O(N^(N^2)).

Оценка памяти:

Размер стека вызова рекурсии на каждом шагу увеличивается не более чем на N — 1, на одно частичное решение выделяется O(N) пямяти. Тогда для N позиций потребуется O(N^3) памяти.

Выводы

В ходе лабораторной работы был исследован рекурсивный алгоритм поиска с возвратом. Был реализован алгоритм на его основе, для самого эффективного заполнения квадрата квадратами меньшего размера.