**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1 (Вар. 4и)**

**по дисциплине «Построение и анализ алгоритмов»**

Тема: Поиск с возвратом

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Глебова В.С. |
| 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≤N≤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≤x,y≤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 возратом. Это алгоритм перебора, который на каждом шаге отсекает дальнейший перебор „плохих“ частичных решений

Работа алгоритма:

1. Вычисление оптимальных сторон и верхней оценки числа квадратов.

2. Расстановка оптимальных квадратов

3. Инициализация стека, комбинаций квадратов

4. Поиск точки с минимальными координатами для установки квадрата. Запись в стек всех возможных квадратов на этой точке.

5. Если размер комбинации достиг предела, то он добавляется в второй стек.

6. Если найдется хоть одно решение, то программа выводит ответ и завершается. Иначе верхний предел размера комбинации увеличивается на 1 и стек инициализируется стеком, в котором находились решения длина которых была равна верхней границе.

7. Возврат к шагу 4.

**Способ хранения частичных решений:**

Частичное решение:

class IntermediateSolution:

def \_\_init\_\_(self, n, m):

self.square = 0

self.squares\_matrix = [[0]\*m for \_ in range(n)]

self.squares = []

Хранит матрицу, которая отображает занятые и свободные ячейки, текущую площадь, квадраты которые находятся в решении.

**Оптимизации алгоритма:**

1. Подсчет оптимального размера сторон и верхнего предела длинны решения исходя из N

2. На каждом шаге выбирается только одна точке, а не все доступные (иначе получим одинаковые решения с разным порядом)

3. В приоритете проверяются решения с большой стороной квадрата

4. Ограничение по длине решения

5. При нахождении решения программа завершает работу

**Оценка сложности и памяти:**

Максимальная оценка количества квадратов линейно зависит от N (N + 4, для простых чисел, берется один квадрат N/2 + 1 и 3 N/2. Остаток заполняется квадратами со стороной 1). Комбинаторно: N позиций, на каждой позиции до N — 1 квадратов + поиск места N^2 => O(N^(N^2))

Оценка памяти:

Размер стека на каждом шагу увеличивается максимум на N — 1, на одно частичное решение выделяется O(N) пямяти. Тогда для N частичных решений O(N^3) памяти.

**Выводы**

В ходе выполнения лабораторной работы был разработан и реализован алгоритм для решения задачи покрытия прямоугольной столешницы минимальным количеством квадратов. Алгоритм основан на методе итеративного бэктрекинга с использованием бинарных масок для оптимизации проверки возможности размещения квадратов. В процессе работы были достигнуты следующие результаты: