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

ОТЧЕТ

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

Студент гр. 3388 _____ Павлов А.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Задание (Вариант 1и. Итеративный бэктрекинг. Выполнение на Stepik двух заданий в разделе 2)

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 перебора вариантов с мемоизацией. Этот алгоритм по своей сути рекурсивно генерирует новые решения из текущего частичного (в данной работе развёрнуто в итеративный бэктрекинг на стеке). Т.е. на каждом шаге перебора (в данном случае добавление нового квадрата) происходит сверка с критериями перебора. Если это решение подходит — возвращаем. Если оно частичное — перебираем дальше его потомков. Если оно не подходит под критерии, то его потомки больше не буду перебираться.

Написанное решение работает менее чем за 1 секунду для натуральных чисел от 2 до 30.

Работа алгоритма:

1. Инициализировать начальную сетку нулями
2. Попытаться найти решение длины target <= N * N + 1:

2.1. Поместить в стек нулевое частичное решение (начальную сетку и пустое решение)

2.2. Пока стек не пуст:

2.2.1. Получить текущее частичное решение из стека

2.2.2. Если его длина больше target, отбросить

2.2.3. Найти первую (слева, сверху) пустую точку

2.2.4. Если её нет, а текущее решение длиной target, записываем как лучшее, в противном случае переход к следующее итерации

2.2.5 Пытаемся поместить в точку квадрат со стороной 1 <= i <= N – max(x, y) - 1:

2.2.6 Если помещается — добавляем в стек новую сетку и обновлённый массив решения, в противном случае — откат и проверка нового i.

3. Если решение было найдено — вывести как лучшее, в противном случае — увеличить target на 1 и повторить поиск.

Алгоритм корректен, так как наивное решение (замощение 1x1) обладает длиной NxN и будет являться его инвариантом. Также на каждой итерации в стек добавляются все возможные случаи, что по сути приводит к полному перебору различных вариантов размещения квадратов. Если есть оптимальное решение (то есть длины target), то алгоритм его сгенерирует и выведет.

Способ хранения частичных решений:

Частичное решение:

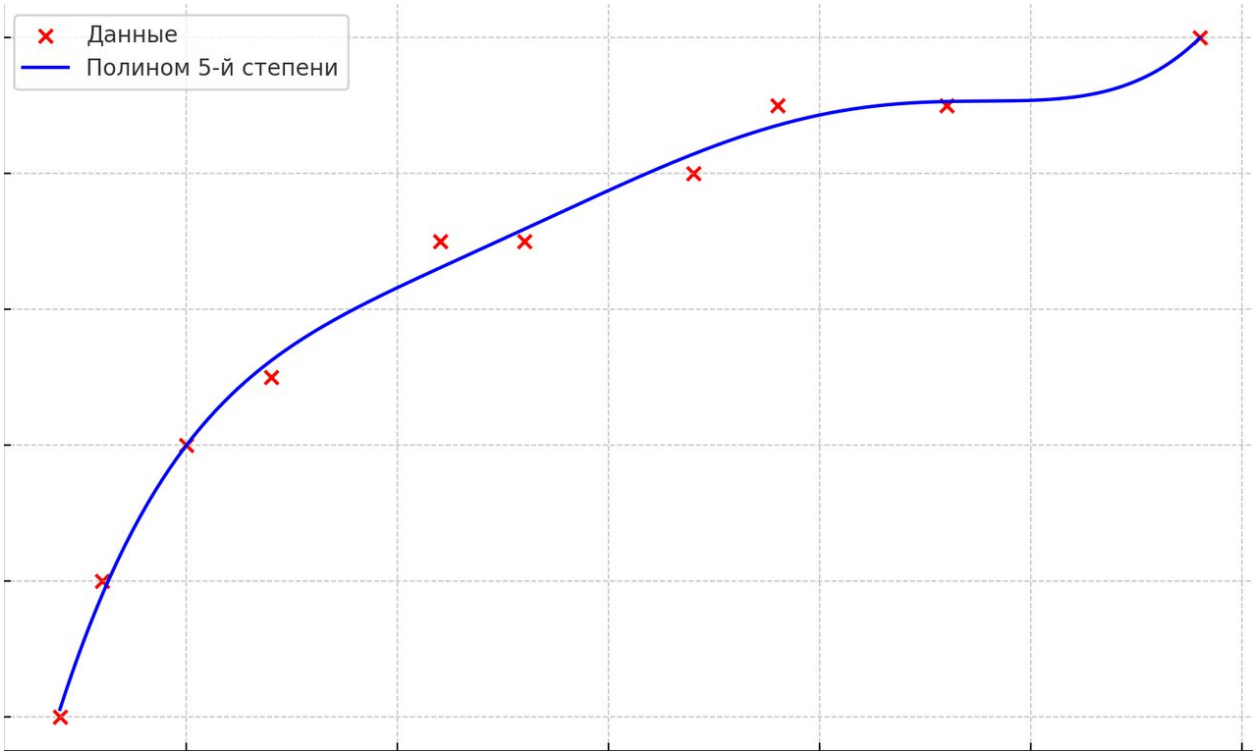
```
struct State {
   Grid grid;
   vector<Square> currentResult;
};
```

Хранит текущую сетку (структура, которая представляет строки как биты чисел и оперирующая с ними, как с битовыми масками) и набор вставленных квадратов в формате (x, y, side).

Оптимизации алгоритма:

1. Сведение задачи к малому порядку, если число составное — очевидно, что достаточно найти решение для наименьшего его делителя, а потом «растянуть» на оставшийся коэффициент.

2. Аппроксимация нижней оценки максимальной длины минимального решения для простых чисел полиномом 5-го поряд-ка:



3. Оперирование не с матрицей для проверки геометрии задачи, а с бинарными масками, подсчитанными на этапе компиляции

4. Ограничение по максимальной длине текущего минимального решения

5. При нахождении решения программа завершает работу

6. Перебор не всех размеров квадрата при генерации нового решения, а только до min(N - 1, N / 2 + 1), где N — размер доски проверяемого решения (всегда простое в силу п.1)

7. В стек последними помещаются решения с большей площадью.

8. Проверка возможности заполнить текущее поле одним квадратом со стороной < N или остались только маленькие со стороной 1.

Оценка сложности и памяти:

Пусть минимум отличается от оценки на константу, она учитываться не будет. Так же не смотря на то, что задача сокращена до более меньшего квадрата для оценки будет использовано N.

На каждом шаге находится одна точка: не более N^2 операций. Проверка возможности 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. Таким образом, итоговая сложность $O(N^{N^2})$.

Оценка памяти:

Размер стека 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.

Тестирование

Входные данные

2

Выходные данные (тривиальное замощение чётного числа)

4

1 1 1

1 2 1

2 1 1

2 2 1

Входные данные

7

Выходные данные (оптимальное решение из условия)

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