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

Вариант 2р

Студент гр. 3388 _____ Потоцкий С.С.

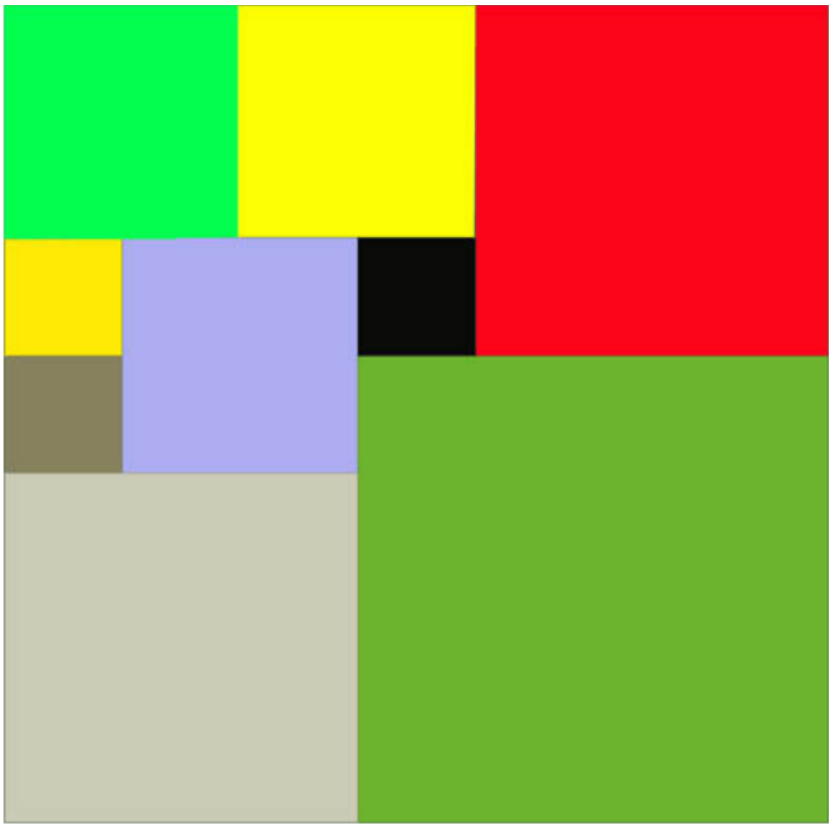
Преподаватель _____ Жангиров Т.Р.

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). Например, столешница размера 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 N (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 и 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

Вар. 2р. Рекурсивный бэктрекинг. Исследование времени выполнения от размера квадрата.

Описание алгоритма

Данный алгоритм решает задачу покрытия таблицы размером N * N квадратами, используя метод backtracking (возврата). Цель - найти минимальный набор квадратов, которые полностью покрывают таблицу.

1. Подготовка:

Определяется делитель divisior для заданного N для последующего масштабирования алгоритма, если это потребуется.

Создается логическая таблица table размером divisior x divisior, представляющая собой поле, которое нужно покрыть квадратами. Изначально все ячейки таблицы помечены как свободные (false).

Определяются и добавляются в tempSolution три начальных квадрата, покрывающие углы таблицы. Эти квадраты имеют размеры, близкие к половине divisior. Соответствующие ячейки в table отмечаются как занятые (true).

2. Backtracking (рекурсивная функция `backtracking`)

Базовый случай 1 (Превышение длины): если текущая длина решения (currentLength) уже больше или равна минимальной найденной длине (minLength), то дальнейший поиск не имеет смысла, и функция возвращается.

Базовый случай 2 (Решение найдено): если в таблице не осталось свободных клеток (функция findFree возвращает {-1, -1}), значит, таблица полностью покрыта квадратами. Если текущая длина решения меньше minLength, то текущее решение сохраняется как лучшее (bestSolution), и minLength обновляется.

Рекурсия:

Находится первая свободная клетка в таблице (cell) с помощью функции findFree. Вычисляется максимально возможная ширина квадрата (maxSide), который можно разместить в этой клетке, с помощью функции calculateMaxWidth. Перебираются все возможные размеры квадрата, начиная с maxSide и до 1.

Для каждого размера side создается квадрат square с левым верхним углом в cell и стороной side, состояние клеток в table, покрытых квадратом, изменяется на true (занято), квадрат добавляется во временное решение tempSolution.

Рекурсивно вызывается функция backtracking для продолжения поиска решения с добавленным квадратом, увеличивая currentLength на 1.

После возврата из рекурсивного вызова состояние клеток в table, покрытых квадратом, восстанавливается в false (свободно). Квадрат удаляется из временного решения tempSolution (backtracking).

Оценка сложности

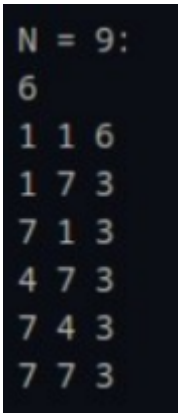
- В худшем случае время работы алгоритма экспоненциально (например, $O(2^{(d^2)})$), где d = divisior – размер таблицы. Это связано с тем, что backtracking может перебрать все возможные варианты размещения квадратов.

- Дополнительно алгоритм использует $O(d^2)$ памяти для хранения таблицы и $O(d^2)$ для глубины рекурсии, что дает суммарную пространственную сложность $O(d^2)$.

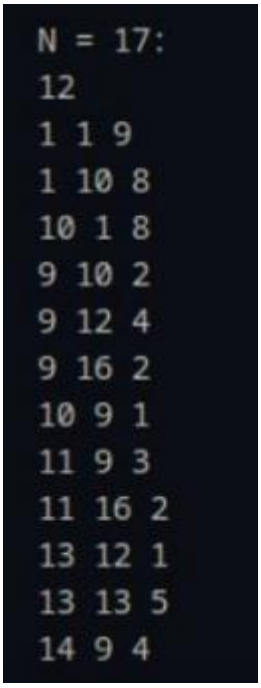
Тестирование

Для длины, являющейся четным числом

Для длины, являющейся нечетным числом



Для длины, являющейся простым числом



Исследование времени выполнения от размера квадрата.

Протестировав алгоритм на значениях от 2 до 20, можно построить график зависимости времени выполнения от размера стороны квадрата, получится следующее изображение:

На нем можно заметить, что самыми быстрыми для выполнения являются стороны, значения которых это четное число, при нечетном числе, не являющемся простым, время выполнения незначительно увеличивается, самое же долгое выполнение происходит, если сторона квадрата это простое число, время выполнения отличается в разы.