Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования
Направление подготовки 09.03.04 Программная инженерия
(код и наименование направления подготовки)
Профиль Проектирование и разработка программного обеспечения
(наименование профиля)

КУРСОВАЯ РАБОТА

по дисциплине Алгоритмы и структуры данных

на тему: «Исследование блочного умножения матриц»

Выполнил студент Астамиров Руслан Романович курса 1 группы 22-КБ-ПР1

Допущена к защите 11.05.23

Руководитель (нормоконтролер) работы _____ Марков В.Н.

Защищена 11.05.23 Оценка Отлично

Члены комиссии: _____ канд. техн. наук, доцент Мурлина В.А.

_____ ст. преп. Литовка Н.В.

Краснодар
2023

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки/специальность 09.03.04 Программная инженерия
(код и наименование направления подготовки/специальности)

Профиль/специализация Проектирование и разработка программного обеспечения
(наименование профиля/специализации)

УТВЕРЖДАЮ
Зав. кафедрой доц. Янаева М.В.
« 13 » февраля 2023 г.

ЗАДАНИЕ
на курсовую работу

Студентке Анисимовой Анастасии Романовне курса 1 группы 22-КБ-ПР1

Тема работы: «Разработка и исследование рекурсивного алгоритма решения задачи о независимых ферзях»

(утверждена указанием директора института № 50-кт от 10.01.2023 г.)

План работы:

1. Разработать рекурсивный алгоритм решения задачи о наибольшем числе ферзей, которые можно расставить на шахматной доске так, чтобы никакие два из них не угрожали друг другу. На доске $n \times n$ максимальное число независимых ферзей равно n.
2. Экспериментально определить функциональную зависимость времени получения первого решения от величины n и её чётности.
3. Экспериментально определить функциональную зависимость времени получения всех решений от величины n и ее чётности.

Объем работы:

а) пояснительная записка 20 с.

б) иллюстративная часть 4 листа.

Рекомендуемая литература:

1. Белов В.В., Чистякова В.И. Алгоритмы и структуры данных [Электронный ресурс]: Учебник / В.В. Белов, В.И. Чистякова. – Москва: КУРС: ИНФРА-М, 2020. – 240 с. Url: https://znanium.com/bookread2.php?book=1057212

2. Колдаев В.Д. Структуры и алгоритмы обработки данных [Электронный ресурс]: Учеб. пособие. – М.: РИОР: ИНФРА-М, 2014. – 296 с. Url: https://znanium.com/bookread2.php?book=418290

3. Марков В.Н. Алгоритмы и структуры данных: учеб. пособие/ В.Н. Марков. – Краснодар: Изд. ФГБОУ ВО «КубГТУ», 2022. – 207 с.

Срок выполнения: с « 13 » февраля по « 09 » июня 2023 г.

Срок защиты: « 09 » июня 2023 г.

Дата выдачи задания: « 13 » февраля 2023 г.

Дата сдачи работы на кафедру: « 09 » июня 2023 г.

Руководитель работы профессор кафедры (должность, подпись) Марков В.Н.

Задание приняла студентка (подпись) Анисимова А.Р.

Реферат

Пояснительная записка курсовой работы содержит: 15 с., 2 табл., 3 рис., 2 источника, 1 приложение.

РЕКУРСИВНАЯ ФУНКЦИЯ, ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ, НЕЗАВИСИМЫЕ ФЕРЗИ

Цель работы – найти функциональную зависимость времени размещения n независимых ферзей на доске $n \times n$ от величины n и её чётности.

Объект исследования – рекурсивный алгоритм решения задачи о независимых ферзях.

Предмет исследования – зависимость времени размещения n независимых ферзей на доске $n \times n$ от величины n и её чётности.

Программное обеспечение создано в среде Microsoft Visual Studio Community 2022 на языке программирования C#. Иллюстрации выполнены в Microsoft Excel 2010.

Содержание

Введение

Задача о независимых ферзях – это классическая задача в теории алгоритмов, которая заключается в размещении максимального числа ферзей на шахматной доске размером $n \times n$ так, чтобы никакие два ферзя не угрожали друг другу. В этой курсовой работе я разработала и исследовала рекурсивную функцию решения этой задачи на языке C#.

Целью данной работы является поиск функциональной зависимости времени размещения n независимых ферзей на доске n×n от величины n и её чётности.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Разработать рекурсивную функцию решения задачи о наибольшем числе ферзей, которые можно расставить на шахматной доске так, чтобы никакие два из них не угрожали друг другу. На доске $n \times n$ максимальное число независимых ферзей равно n.

2. Экспериментально определить функциональную зависимость времени получения первого решения от величины n и её чётности.

3. Экспериментально определить функциональную зависимость времени получения всех решений от величины n и её чётности

1 Разработка алгоритма

1.1 Постановка задач

Разработать рекурсивную функцию решения задачи о наибольшем числе ферзей, которые можно расставить на шахматной доске так, чтобы никакие два из них не угрожали друг другу. На доске $n \times n$ максимальное число независимых ферзей равно n.

1.2 Разработка рекурсивной функции

Для решения задачи о независимых ферзях мы можем использовать рекурсивную функцию, которая будет пытаться разместить ферзя на каждой строке доски, начиная с первой строки. Для каждой позиции ферзя мы проверяем, не угрожает ли он другим ферзям, которые уже находятся на доске. Если ферзь может быть размещен на текущей позиции, мы помечаем эту позицию на доске и вызываем функцию для следующей строки. Если функция успешно разместила ферзей на всех строках, мы считаем, что мы нашли одно из возможных решений задачи

1.3 Программная реализация

Листинг

```
private static DateTime time = DateTime.Now;
private static int n = 10;
private static int countItem = 0;
private static int countSolution = 0;

public static void Main(string[] args)
{
char[,] chessItem = new char[n, n];
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;
        chessItem[i, j] = '\u25A1';
    }
}
PutItems(chessItem, 0);
DateTime time1 = DateTime.Now;
Console.WriteLine($"Всего рещений: {countSolution}");
Console.WriteLine($"Время всех решений:
{time1.Subtract(time).TotalSeconds}");
```

```
}

public static void Solution(char[,] chessItem)
{
Console.WriteLine("Первое решение");
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        Console.Write(chessItem[i, j] + " ");
    }
    Console.WriteLine();
}
Console.WriteLine();
}

public static void PutItems(char[,] chessItem, int line)
{
if (countItem == n)
{
    countSolution++;
    if (countSolution == 1)
    {
        Solution(chessItem);
        DateTime time1 = DateTime.Now;
        Console.WriteLine($"Время первого решения:
{time1.Subtract(time).TotalSeconds}");
    }
    return;
}

for (int i = 0; i < n; i++)
{
    if (SearchItem(chessItem, line, i))
    {
        chessItem[line, i] = '\u2655';
        countItem++;
        PutItems(chessItem, ++line);
        countItem--;
        chessItem[--line, i] = '\u25A1';
    }
}
}

public static bool SearchItem(char[,] chessItem, int line, int
column)
{
for (int i = 0; i < n; i++)// | _
{
    if (chessItem[i, column] == '\u2655' || chessItem[line, i]
== '\u2655') { return false; }
}
```

```
for (int i = line, j = column; i >= 0 && j >= 0; i--, j--) // \
{
    if (chessItem[i, j] == '\u2655') { return false; }
}

for (int i = line, j = column; i < n && j < n; i++, j++) // \
{
    if (chessItem[i, j] == '\u2655') { return false; }
}

for (int i = line, j = column; i >= 0 && j < n; i--, j++) // /
{
    if (chessItem[i, j] == '\u2655') { return false; }
}

for (int i = line, j = column; i < n && j >= 0; i++, j--) // /
{
    if (chessItem[i, j] == '\u2655') { return false; }
}

return true;
```

1.4 Описание алгоритма

Алгоритм начинается с инициализации пустой шахматной доски размером $n \times n$. Вначале она заполнена символами «□». Затем происходит вызов метода PutItems(), который вызывает метод SearchItem для каждой позиции на шахматной доске, начиная с верхнего левого угла. Если на данной позиции можно разместить ферзя, то она замещается символом «♕», увеличивается счётчик расставленных ферзей countItem и происходит рекурсивный вызов метода PutItems() для следующей строки доски. Если countItem равен n, то значит, все ферзи были размещены на доске. Метод Solution() я вызываю, когда найдено первое решение. Он выводит расстановку ферзей (рис. 1).

Метод SearchItem проверяет, не бьёт ли ферзь на данной позиции других ферзей, ранее размещенных на доске. Проверка происходит по горизонтали, вертикали и двум диагоналям. Если на данной позиции не бьется ни один из ферзей, метод SearchItem возвращает true, в противном случае – false.

Таким образом, алгоритм решения задачи о восьми ферзях основывается на последовательной проверке возможности расстановки ферзей на доске с помощью метода обратной связи и рекурсии. Это позволяет перебрать все возможные варианты расположения ферзей без повторений и найти все решения задачи.

1.5 Вывод программы

Рисунок 1 – Результат работы алгоритма

В конце алгоритма выводится расстановка ферзей, время нахождения первого решения, количество найденных решений и время, затраченное на их поиск.

2 Экспериментальная часть

2.1 Определение функциональной зависимости времени получения первого решения

Для определения функциональной зависимости времени получения первого решения от величины *n* и её чётности был проведён эксперимент по измерению времени выполнения функции PutItems() для разных значений *n* (табл. 1).

Таблица 1 – Результаты первого эксперимента

Четные	t	Нечетные	t
20	0,7207194	23	0,1174791
22	6,7520933	25	0,3388385
24	2,0363227	27	1,9216334
26	2,2472722	29	7,8704393
28	14,316322	31	67,712007
30	250,89101	33	1048,1057
32	792,47239		

Чтобы найти функциональную зависимость, я построила график времени выполнения в зависимости от *n* с помощью Microsoft Excel (рис. 2).

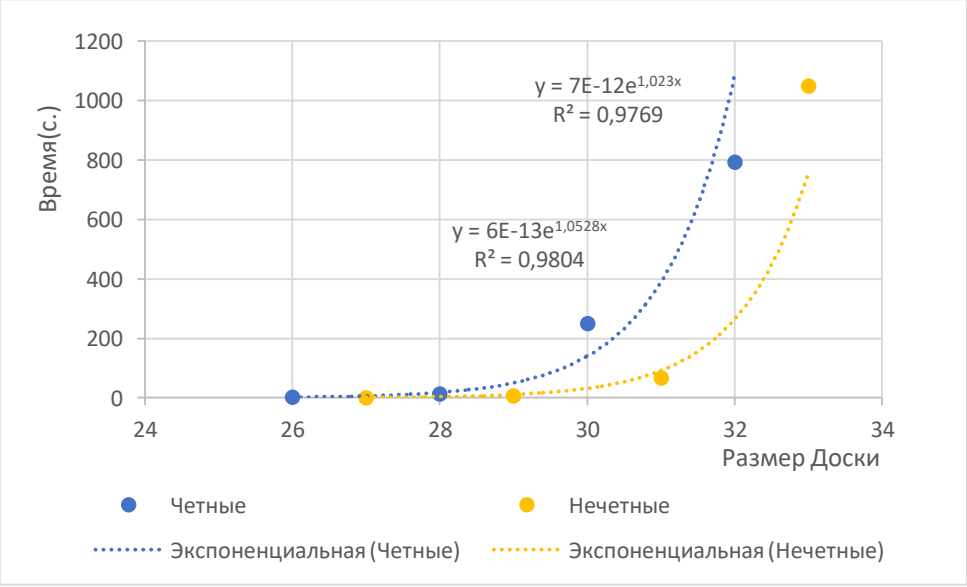

Рисунок 2 – Функциональная зависимость времени получения первого решения от величины доски и её чётности

На графике видно, что время выполнения функции PutItems растёт экспоненциально с увеличением *n*. Кривая, соответствующая чётным

значениям n, проходит выше кривой, соответствующей нечётным значениям *n*. Это может быть связано с тем, что на доске с чётным n количество клеток чёрного и белого цвета одинаково, а на доске с нечётным *n* количество клеток разное.

2.2 Определение функциональной зависимости времени получения всех решений

Для определения функциональной зависимости времени получения всех решений от величины *n* и её чётности был проведён второй эксперимент для разных значений *n* (табл. 2).

Таблица 2 – Результаты второго эксперимента

Четные	t	Нечетные	t
10	0,0621143	11	0,2579941
12	1,3891622	13	7,4502442
14	35,495664	15	244,83155
16	1730,5929		

Функциональная зависимость найдена в Microsoft Excel (рис. 3).

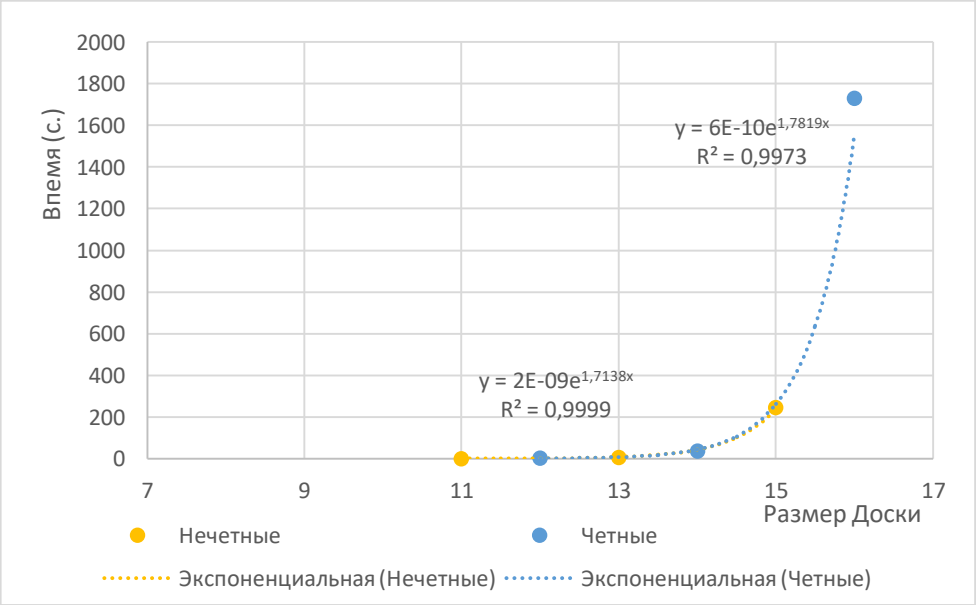

Рисунок 3 – Функциональная зависимость времени получения всех решений от величины доски и ее четности

На рисунке видно, что время выполнения функции PutItems также растёт экспоненциально с увеличением *n*. Кривая, соответствующая чётным

значениям n, почти совпадает с кривой, соответствующей нечётным значениям n.

Заключение

В этой курсовой работе я разработала и исследовала рекурсивную функцию решения задачи о независимых ферзях на языке C#. Функция работает при различных значениях n. Были построены графики зависимости времени выполнения от размера n и его чётности. Эта задача может быть полезна для работы с рекурсией и поиска решений на шахматной доске.

Были рассмотрены две задачи: нахождение максимального числа ферзей, которые можно расставить на доске так, чтобы никакие два из них не угрожали друг другу, и нахождение всех возможных решений задачи о независимых ферзях.

В работе проведено экспериментальное исследование временной сложности реализованных функций, в результате которого была определена функциональная зависимость времени работы функций поиска решений от величины n и её чётности. Выяснилось, что временная сложность решения задачи о независимых ферзях на шахматной доске увеличивается экспоненциально с ростом размера доски.

Также было выявлено, что для чётных значений n задача о независимых ферзях решается быстрее, чем для нечётных значений n. Это может быть связано с тем, что при чётных значениях n на доске одинаковое количество белых и чёрных клеток, что упрощает процесс расстановки ферзей.

Таким образом, разработанные рекурсивные функции представляют собой эффективный способ решения задачи о независимых ферзях на шахматной доске, а экспериментальное исследование позволило выявить особенности и зависимости решения данной задачи от параметров входных данных.

Список используемых материалов

1. Алгоритмы и структуры данных: учеб. пособие/ В.Н. Марков. – Краснодар: Изд. ФГБОУ ВО «КубГТУ», 2022. – 207 с.

2. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. – СПб.: Питер, 2014. – 432 с.: ил.

Приложение

Проверка на антиплагиат