МГТУ им. Баумана

Рубежный контроль №1

По курсу: "Анализ алгоритмов"

Обход полного бинарного дерева в глубину

Работу выполнил: Гаврилов Дмитрий, ИУ7-56Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019

Оглавление

Введение

Цель работы: изучение рекурсионоого и итерационного алгоритмов обхода полного бинарного дерева в глубину. Также требуется оценить затраты по памяти и замерить время работы алгоритмов .

В ходе рубжного контроля предстоит решить следующие задачи:

- изучить алгоритмы обхода дерева в глубину;
- реализовать рекурсионный и итерационный алгоритмы на одном из языков программирования;
- сравнить алгоритмы обхода дерева в глубину.

1 Аналитическая часть

Дерево — одна из наиболее широко распространённых структур данных в информатике, эмулирующая древовидную структуру в виде набора связанных узлов. Является связным графом, не содержащим циклы. Среди деревьев выделяют особый подкласс, называемый бинарными деревьями. Бинарное дерево - корневое дерево, каждая вершина которого имеет не более двух дочерних, чаще всего чётко упорядоченных: левую и правую. Среди бинарных деревьев отдельно выделяют полные бинарные деревья, все вершины которых имеют по две дочерних, кроме листьев, которые расположены на одинаковой глубине.

1.1 Поиск в глубину

Поиск в глубину (англ. Depth-first search, DFS) — один из методов обхода графа. Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно.

1.2 Рекурсивный алгоритм

Перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

1.3 Итеративный алгоритм

Обрабатываем текущий узел, при наличии правого поддерева добавляем его в стек для последующей обработки. Переходим к узлу левого поддерева. Если левого узла нет, переходим к верхнему узлу из стека.

1.4 Вывод

Были рассмотрены алгоритмы обхода в глубину полного бинарного дерева.

2 | Технологическая часть

2.1 Выбор ЯП

В качестве языка программирования был выбран Java [?], а средой разработки Intellij IDEA. Время работы алгоритмов было замерено с помощью класса Instant.

2.2 Сведения о модулях программы

Программа состоит из:

- Main.java - главный файл программы, в котором располагается точка входа в программу, реализации алгоритмов и функция замера времени.

2.3 Листинг кода алгоритмов

В листингах 3.1 - 3.2 будет рассмотрена реализация описанных алгоритмов.

Листинг 2.1: Рекурсивный алгоритм обхода дерева в глубину

```
1 public static int recDFS(Node root){
2         if (root.left!=null) recDFS(root.left);
3         if (root.right!=null) recDFS(root.right);
4 
5         return 0;
6     }
```

Листинг 2.2: Итерационный алгоритм обхода дерева в глубину

```
1  public static int contDFS(Node top){
2          Stack<Node> stack = new Stack<> ();
3          while (top!=null || !stack.empty()){
4              if (!stack.empty()){
5                  top=stack.pop();
6              }
7              while (top!=null){
8                  if (top.right!=null) stack.push(top.right);
9                  top=top.left;
10             }
11         }
12         return 0;
13     }
```

3 Исследовательская часть

3.1 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов.

Эксперимент производится для полного бинарных деревьев размером от 5 до 25. Сравним результаты для разных алгоритмов:

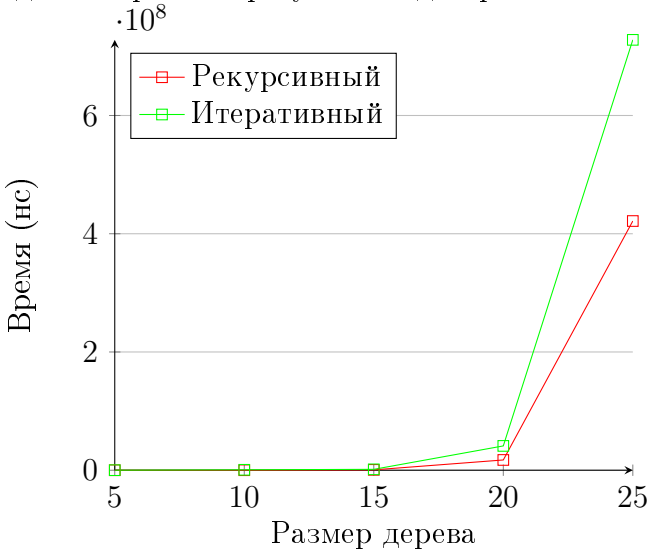

Рис. 4.1: Сравнение времени работы алгоритмов при разных размерах дерева

3.2 Сравнительный анализ на основе замеров памяти алгоритмов

Сравним потребление памяти алгоритмами при обходе дерева высотой 15.

Сравнительная таблица затрат по памяти.

Память	Рекурсивный алгоритм	Итерационный алгоритм
Потребляемая память	40 * 15	80 + 40 * 15
Итого	600	680

3.3 Вывод

По результатам тестирования времени на дереве высотой до 15 уровней алгоритмы продемонстрировали себя равно. С увеличением высоты дерева, рекурсивный алгоритм оказался быстрее. По результатам сравнения затрат памяти слегка выигрывает рекурсивный алгоритм. Но стоит учитывать, что при обходе дерева большой высотой рекурсивным способом, может переполниться стек вызовов.

Заключение

В ходе рубежного контроля я изучил алгоритмы обхода полного бинарного дерева в глубину: рекурсиивный и итеративный, реализовал алгоритмы обхода дерева на языке программирования Java и сравнил эти алгоритмы.

Литература

[1] И. В. Белоусов(2006), Матрицы и определители, учебное пособие по линейной алгебре, с. 1 - 16

[2] Le Gall, F. (2012), "Faster algorithms for rectangular matrix multiplication Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012), pp. 514–523

[3] Документация по языку Java[Электронный ресурс], - режим доступа: https://docs.oracle.com/en/java/javase/13/