ЛАБОРАТОРНАЯ РАБОТА №3
ПОСТРОЕНИЕ БИНАРНЫХ ДЕРЕВЬЕВ, представленных массивами

1 Цель работы

Целью работы является построение дерева заданной структуры и прохождение его в заданном порядке.

2 Основные теоретические положения

Представляемые массивами деревья находят применение в приложениях с пирамидами (heaps), являющимися законченными бинарными деревьями, имеющими упорядочение узлов по уровням.

Heap-tree, или куча— такое двоичное дерево, для которого выполнены три условия:

1. Значение в любой вершине не меньше, чем значения еѐ потомков.
2. Глубина всех листьев (расстояние до корня) отличается не более чем на 1 уровень.
3. Последний уровень заполняется слева направо без «дырок».

Такие кучи называются — ***max-heap.***

Существуют также кучи, где значение в любой вершине, наоборот, не больше, чем значения еѐ потомков. Такие кучи называются ***min-heap.***

В куче элементы хранятся в виде двоичного дерева, то есть у элементов есть два потомка - левый и правый.

В вершине кучи находится один элемент, у него - два потомка на следующем уровне, у них, в свою очередь, по два потомка на третьем уровне (итого - 4 элемента на третьем уровне) и т. д.

Уровни заполняются в порядке увеличения номера уровня, а сам уровень заполняется слева направо. У элементов последнего уровня нет ни одного потомка, возможно, что и у некоторых элементов предпоследнего уровня нет потомков. Также в куче может быть один элемент, у которого только один потомок (левый).

Структура данных для хранения двоичной кучи

Удобная структура данных для сортирующего дерева — массив A, у которого первый элемент A[1] — элемент в корне, а потомками элемента A[i] являются A[2i] и A[2i+1] (при нумерации элементов с первого).

При нумерации элементов с нулевого, корневой элемент — A[0], а потомки элемента A[i] — A[2i+1] и A[2i+2]. При таком способе хранения условия 2 и 3 выполнены автоматически.

Алгоритмы построения heap-дерева

Первый способ построить кучу из неупорядоченного массива – это по очереди добавить все его элементы.

Если новый элемент имеет значение большее, чем у родителя, узлы меняются местами.

Временная сложность такого алгоритма O(N*log2 N).

Второй способ. Можно построить кучу за N шагов. Для этого сначала следует построить дерево из всех элементов массива, не заботясь о соблюдении основного свойства кучи, а потом произвести упорядочение для всех вершин, у которых есть хотя бы один потомок (так как поддеревья, состоящие из одной вершины без потомков, уже упорядочены).

При этом из двух потомков нужно выбрать наибольший и, если этот наибольший потомок больше стоящего в вершине кучи, обменять их местами.

3 Последовательность выполнения работы

1. Сгенерировать n уникальных ключей с использованием датчика случайных чисел.
2. Построить дерево заданного типа, используя в качестве вершин сгенерированную последовательность ключей.
3. Обойти построенное дерево с помощью заданного метода прохождения и представить полученную последовательность в отчете по лабораторной работе.

4 Форма отчетности

1. На экране дисплея должны отображаться:
   - последовательность сгенерированных ключей,
   - процесс построения дерева,
   - графическое изображение построенного дерева.
2. Электронный вариант программы.
3. Отчет о проделанной лабораторной работе, содержащий следующие пункты:
   - задание на лабораторную работу;
   - определение заданного дерева;
   - алгоритм построения дерева;
   - алгоритм прохождения дерева;
   - входные и выходные данные;
   - примеры работы программы.

5 Контрольные вопросы

1. Какая структура называется деревом?
2. Какое дерево называется идеально сбалансированным?
3. Какой алгоритм построения идеально сбалансированного дерева?
4. Какое дерево называется поисковым?
5. Недостатки идеально сбалансированного и поискового деревьев.
6. Какое дерево называется Heap-Tree.

ВАРИАНТЫ ЗАДАНИЙ

1. Сгенерировать 26 трехзначных неповторяющихся чисел.
Вывести их на экран.
Показать процесс построения Min-Heap-Tree.
Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую симметричному прохождению дерева с приоритетом направо (RNL).

2. Сгенерировать 24 неповторяющихся трехзначных элементов.
Вывести их на экран.
Показать процесс построения Min-Heap-Tree.
Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую прямому порядку прохождения (NLR).

3. Сгенерировать 25 неповторяющихся двухзначных элементов. Вывести их на экран.
Показать процесс построения Min-Heap-Tree.
Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую прямому порядку прохождения с приоритетом направо (NRL).

4. Сгенерировать 25 неповторяющихся трехзначных элементов. Вывести их на экран.
Показать процесс построения Min-Heap-Tree.
Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую обратному порядку прохождения (LRN).

5. Сгенерировать 27 двухзначных неповторяющихся чисел (элементов).

Вывести их на экран.

Показать процесс построения Min-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую симметричному прохождению дерева (LNR).

6. Сгенерировать 26 двухзначных неповторяющихся чисел.

Вывести их на экран.

Показать процесс построения Min-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую обратному прохождению дерева с приоритетом направо (RLN).

7. Сгенерировать 25 трехзначных неповторяющихся чисел (элементов).

Вывести их на экран.

Показать процесс построения Max-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Max-Heap-Tree, соответствующую обратному прохождению дерева (LRN).

8. Сгенерировать 25 трехзначных неповторяющихся чисел (элементов).

Вывести их на экран.

Показать процесс построения Max-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Max-Heap-Tree, соответствующую обратному прохождению дерева с приоритетом направо (RLN).

9. Сгенерировать 26 трехзначных неповторяющихся чисел (элементов).

Вывести их на экран.

Показать процесс построения Max-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Max-Heap-Tree, соответствующую прямому порядку прохождения (NLR).

10. Сгенерировать 26 трехзначных неповторяющихся чисел (элементов).

Вывести их на экран.

Показать процесс построения Max-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Max-Heap-Tree, соответствующую прямому порядку прохождения с приоритетом направо (NRL).

11. Сгенерировать 25 неповторяющихся трехзначных элементов.

Вывести их на экран.

Показать процесс построения Min-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую симметричному прохождению дерева (LNR).

12. Сгенерировать 24 неповторяющихся трехзначных элементов.

Вывести их на экран.

Показать процесс построения Min-Heap-Tree.

Вывести на экран полученное дерево.

Дополнительное задание: в отчете перечислить последовательность вершин построенного Min-Heap-Tree, соответствующую симметричному прохождению дерева с приоритетом направо (RNL).