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

ОТЧЁТ

по курсовой работе
по дисциплине «Алгоритмы и структуры данных»
Тема: Демонстрация работы пирамиды.

Студент гр. 1304 _____ Мусаев А.И.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2022

Задание на курсовую работу

Студент: Мусаев А.И.
Группа: 1304
Тема работы: Демонстрация работы пирамиды

Реализовать следующие алгоритмы на основе пирамиды:

- Вставка элемента
- Удаление наименьшего элемента
- Поиск элемента
- Пирамидальная сортировка

.

Содержание пояснительной записки:

- Введение
- Основные теоретические положения
- Реализация программы
- Тестирование
- Заключение

Дата выдачи задания: 25.10.2022
Дата сдачи работы: **.**.2022
Дата защиты работы: **.**.2022

Студент гр. 1304 _____ Мусаев А.И.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

В данной курсовой работе была реализована программа, имеющая следующий функционал на основе пирамиды:

- Вставка элемента
- Удаление наименьшего элемента
- Удаление наибольшего элемента
- Поиск элемента
- Пирамидальная сортировка

НАВИГАЦИЯ ПО ПОЯСНИТЕЛЬНОЙ ЗАПИСКЕ

Содержание

ВВЕДЕНИЕ

Целью данной работы является разработка программы, которая имеет функционал по работе со структурой данных пирамида.

Программа должна получать исходные данные из командной строки и выполнять поставленные ей задачи. Кроме того, программа должна демонстрировать выполняемые ею процессы.

Для реализации данной программы предстоит решить следующие задачи:

- Изучить структуры данных пирамида
- Изучить операции с пирамидой
- Изучить пирамидальную сортировку
- Реализовать удобный для пользователя интерфейс, так как работа программы демонстрируется пользователю
- Реализовать демонстрацию, понятную для пользователя

1 Основные теоретические положения

1.1 Что такое пирамида?

Пирамида или куча - абстрактная структура данных, поддерживающая следующие операции:

1. Нахождение минимума
2. Удаление минимума
3. Добавление нового элемента в кучу

Другое название, лучше отражающее функциональность — очередь с приоритетами.

Кучи используются во многих алгоритмах. Например, кучи используются в алгоритмах поиска кратчайшего пути, а также с помощью кучи можно проводить сортировку (путём превращения массива в кучу, а кучу в отсортированный массив).

Для кучи всегда выполнены условия:

1. Значение в любой вершине не больше, чем значения её потомков
2. У любой вершины не более двух сыновей
3. Слои заполняются последовательно сверху вниз и слева направо, без «дырок»

Заметим, что двоичная куча строится неоднозначно: например, значения сыновей, которые являются листами, всегда можно менять местами. Фиксирована только сама структура и предикат «родитель не больше детей».

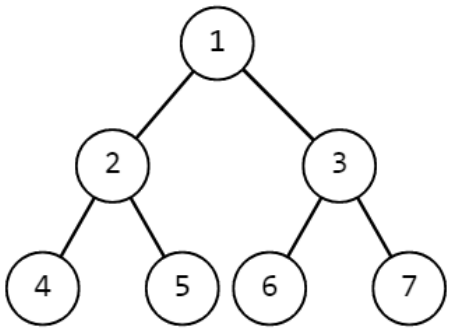

Рис. 1: Куча для минимума

Обозначим высоту дерева как h. Так как куча всегда состоит из нескольких слоев заполненных полностью и одного заполненного частично, и

каждый следующий слой содержит в два раза больше вершин, чем предыдущий, то высота дерева будет $\theta(logn)$.

Как и любая очередь с приоритетами, двоичная куча должна уметь выполнять операции:

1. Нахождение минимума за $O(1)$.
2. Удаление минимума за $O(h)$.
3. Добавление нового элемента в кучу за $O(h)$.

1.2 Как осуществляется вставка элемента?

Чтобы вставить элемент в дерево, мы выполняем следующий алгоритм:

1. Дописывем элемент на нижний уровень в крайнем правом пустом ребёнке.
2. Сравниваем элемент с родительским, если они расположены в верном порядке, останавливаемся.
3. Иначе меняем с родительским и переходим к предыдущему шагу.

2 и 3 шаги алгоритма называют просеиванием вверх.

Ассимптотика такого алгоритм $O(h)$, так как мы проходим все уровни дерева, а высота дерева, как мы ввели, равна h.

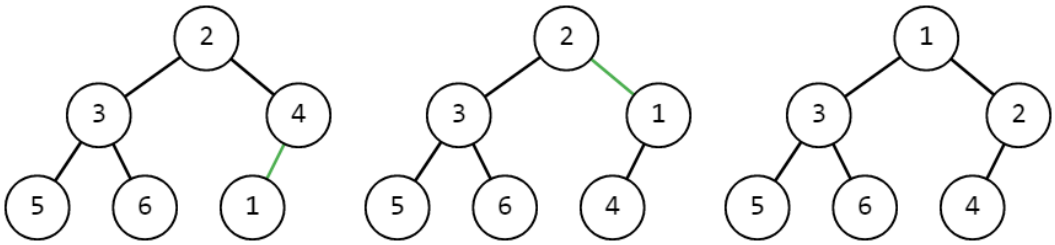

Рис. 2: Пример работы алгоритма вставки

1.3 Удаление наименьшего элемента

Процедура удаления корня из кучи с сохранением свойства кучи выглядит следующим образом:

1. Заменяем корень кучи последним элементом на последнем уровне.
2. Сравниваем новый корень с его дочерними элементами; если они расположены в правильном порядке, останавливаемся.

3. Если нет, заменяем элемент одним из его дочерних элементов и возвращаемся к шагу 2.

2 и 3 шаги называют просеиванием вниз.

Ассимптотика такого алгоритм тоже $O(h)$, так как мы проходим все уровни дерева.

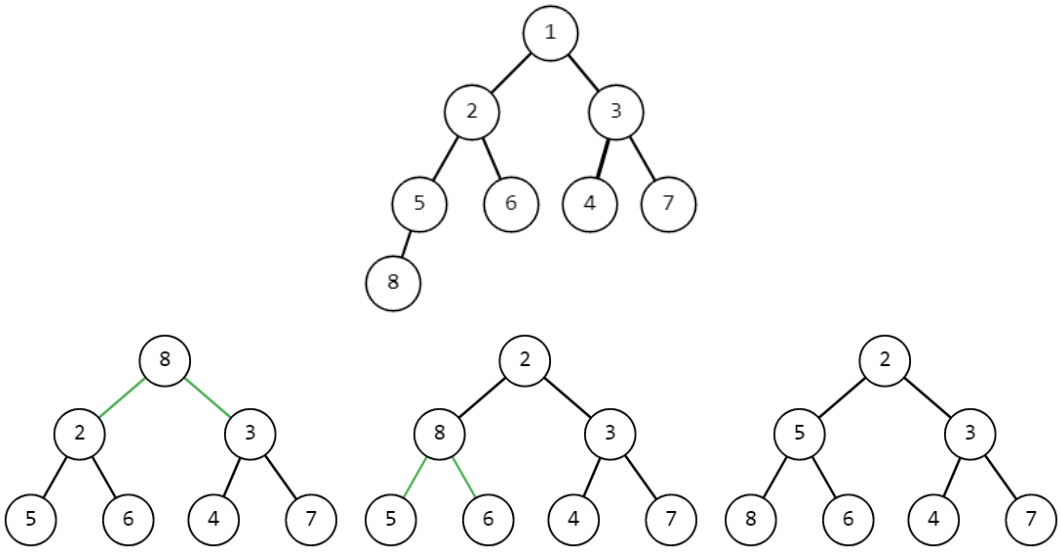

Рис. 3: Пример работы алгоритма удаления минимального

1.4 Удаление наибольшего элемента

Удаление наибольшего рассматриваем для мин-кучи, так как для макс-кучи оно описано в предыщем пункте.

1. Пробегаемся по нижне половине дерева.
2. Находим максимум.
3. Ставим на место максимума последний лист.
4. Просеиваем новое значение вверх.

Ассимптотика такого алгоритм - $\frac{n}{2} + logn$, то есть $O(n)$, так как пробежаться по нижней половине дерева - это $\frac{n}{2}$ операций, а просеить вверх - это $logn$ операций.

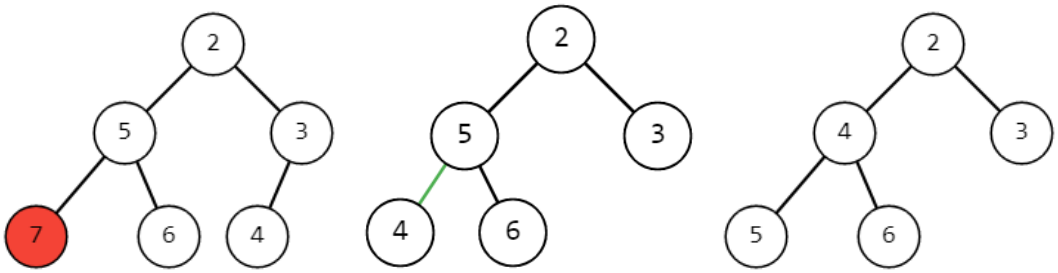

Рис. 4: Пример работы алгоритма удаления максимального

1.5 Поиск элемента

К сожалению, куча не рассчитана на оптимальный поиск элемента, поэтому поиск происходит за $O(n)$, нам нужно просто пройтись по все элементам дерева.

1.6 Пирамидальная сортировка

В пирамидальной сорировке такой алгоритм:

1. Построить макс-кучу из входного массива.
2. Извлечь максимальный элемент и переставить его в конец массива.
3. Вернуться к шагу 1 с массивом размером на 1 меньше.

Ассимптотика такого алгоритма $O(n * logn)$, так как мы извлекаем n элементов, а после каждого извлечения идёт просеивание вниз за $logn$.

Построение кучи работает так:

Мы запускаем рекурсивную функцию, которая наведёт порядок сначала на самых нижних уровнях кучи и дойдёт так до корня дерева. Так мы получим верное дерево и начнём извлекать элементы.

2 Реализация программы

2.1 Файл main

Здесь происходит взаимодействие с пользователем: тут происходит считывание команд от пользователя и проверка входных данных, которые подаёт пользователь.

Сначала происходит считывание максимального размера дерева, так как куча хранится на основе массива. Затем происходит создание кучи и вывод правил пользования.

2.2 Файл Heap

Хранить кучу будем в виде массива, где у корня индекс равен 0, а у вершины k индексы ее детей равны $2k + 1$ и $2k + 2$.

В этом файле хранится класс кучи. В нём есть следующие методы:

1. get_parent(index) - метод, возвращающий индекс родителя, то есть просто `return (index - 1) // 2`
2. get_left_child, get_right_child - методы, возвращающие индексы детей по формуле, написанной выше.
3. insert - добавление элемента в дерево. (пункт 1.2)
4. extract_min - удаление минимального. (пункт 1.3)
5. sift_up - просеивание вверх.
6. sift_down - просеивание вниз.
7. str - преобразование массива в строку.
8. extract_max - удаление наибольшего. (пункт 1.4)
9. search - поиск элемента (пункт 1.5)

2.3 Файл Sort

В этом файле хранятся функции, выполняющие сортировку пирамидой.

Функция heapify(a, n, index) выполняет правильную расстановку элементов в дереве, то есть:

1. Находит индексы правого и левого ребёнка поданного узла.
2. Находит максимальный элемент из 3: из родителя, правого и левого ребёнка.

3. Если один из детей больше родителя, меняем их местами, вызываем функцию heapify от нового индекса поданного элемента, чтобы проверить, что, когда мы навели порядок на нынешнем уровне, не сломали всё уровнем ниже.

Функция heap_sort(a, flag_want) сначала наводит порядок в массиве (превращает его в дерево): она вызывает heapify на всех уровнях, кроме последнего, на последнем не вызывается, так как у них нет детей.

После наведения порядка вызывается цикл, который извлекает максимальный элемент, переставляет его в конец массива, наводит порядок в куче на один меньше и делает ещё раз то же самое, пока не дойдёт до 0 элемента в массива.

2.4 Файл visual

Этот файл рисует дерево.

В нём есть:

- Класс узел (TreeNode), который может хранить в себе значение, ссылку на левого и ссылку на правого ребёнка.
- Функция deserialize создаёт из массива дерево, хранящееся через класс TreeNode и возвращается корень этого дерева
- Функция drawtree хранит в себе несколько функций:
  - height - находит высоту дерева
  - jumpto - переносит исполнителя turtle на нужные координаты
  - draw - вывод самого узла и его детей

  Это всё вспомогательные функции для того, чтобы делать основную работу вывода дерева. Сначала находим высоту дерева. Черепаха переходит на нужные координаты и рекурсивно рисует дерево.

3 Тестирование

4 Заключение

В ходе данной курсовой работы была изучена структура данных пирамида.

Реализованная программа принимает на вход комманды, исходя из которых проводит демонстрацию работы пирамиды.

В ходе курсовой:

- Был изучен теоретический материал по пирамиде
- Был разработан и реализован программный код
- Было проведено тестирование программы

5 Список используемых источников

- https://shkolkovo.online
- https://ru.algorithmica.org
- https://stackoverflow.com
- https://en.wikipedia.org/wiki/Heapsort