Лабораторная работа № 5

Тема работы:

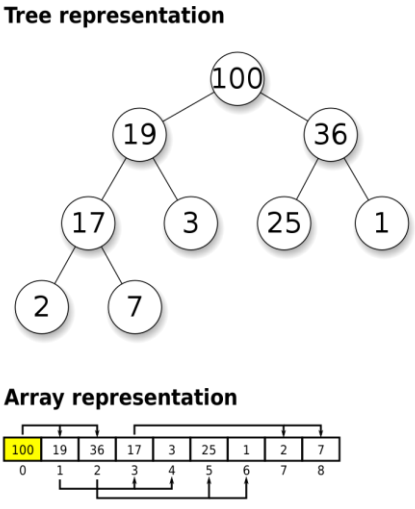
Адресация памяти и хранение переменных в стеке и куче (heap).

Теоретическая часть:

В информатике **ку́ча** (англ. **heap**) — это специализированная структура данных типа **дерево**, которая удовлетворяет свойству кучи: если B является узлом-потомком узла A, то ключ(A) ≥ ключ(B). Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом кучи, поэтому иногда такие кучи называют max-кучами (в качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют min-кучами). Не существует никаких ограничений относительно того, сколько узлов-потомков имеет каждый узел кучи, хотя на практике их число обычно не более двух. Куча является максимально эффективной реализацией абстрактного типа данных, который называется очередью с приоритетом. Кучи имеют решающее значение в некоторых эффективных алгоритмах на графах, таких, как алгоритм Дейкстры на d-кучах и сортировка методом пирамиды.

Структуру данных куча не следует путать с понятием куча в динамическом распределении памяти. Впервые термин использовался именно для структур данных. В некоторых ранних популярных языках программирования типа ЛИСП обеспечивалось динамическое распределение памяти с использованием структуры данных «куча», которая и дала своё имя выделяемому объёму памяти.

Кучи обычно реализуются в виде массивов, что исключает наличие указателей между её элементами.



Источник: Википедия

Операции над структурой данных Heap:

- **Heapify** – процесс создания кучи из массива.
- **Вставка** (Insertion) – процесс вставки элемента в существующую кучу временной сложностью O(log N).
- **Удаление** (Deletion) – удаление верхнего элемента кучи или элемента с наивысшим приоритетом, а затем организация кучи и возвращение элемента с временной сложностью O(log N).
- **Peek** – проверка или поиск первого (или, можно сказать, верхнего) элемента кучи.

Типы структур данных кучи

В общем случае кучи могут быть двух типов:

- **Max-Heap** – в Max-Heap ключ, находящийся в корневом узле, должен быть наибольшим среди ключей, находящихся во всех его дочерних узлах. Это же свойство должно быть рекурсивно истинным для всех поддеревьев данного двоичного дерева.
- **Min-Heap** – В Min-Heap ключ, находящийся в корневом узле, должен быть минимальным среди ключей, находящихся во всех его дочерних узлах. Это же свойство должно быть рекурсивно истинным для всех поддеревьев данного двоичного дерева.

Переведено с помощью www.DeepL.com/Translator (бесплатная версия)

Источник: https://www.geeksforgeeks.org/heap-data-structure/

Основное задание:

Реализовать класс **Heap**, со всеми возможными операциями для манипуляции с данным классом. Используя класс **List** или **Map** написать стандартные методы класса Heap для дальнейшего внесения информации и её обработки.

Варианты для реализации задания:

- **Легкий вариант:**
  Написать метод heap sort, который проводит сортировку схожую с selection sort. Для этого необходимо построить кучу из заданного входного массива. Затем повторять следующие шаги до тех пор, пока куча не будет содержать только один элемент:

  - Поменять местами корневой элемент кучи (самый большой элемент) с последним элементом кучи.
  - Удалить последний элемент кучи (который теперь находится в правильном положении).
  - Упорядочить оставшиеся элементы кучи.

  Отсортированный массив получается путем изменения порядка следования элементов во входном массиве.

Средний вариант:
Для массива из N элементов, в котором каждый элемент находится на расстоянии не более K от своего целевого положения, разработайте алгоритм, который выполняет сортировку за время O (N log K).

Input: arr[] = {6, 5, 3, 2, 8, 10, 9}, K = 3
Output: arr[] = {2, 3, 5, 6, 8, 9, 10}

Input: arr[] = {10, 9, 8, 7, 4, 70, 60, 50}, K = 4
Output: arr[] = {4, 7, 8, 9, 10, 50, 60, 70}

Сложный вариант:
Даны K отсортированных массивов размером N каждый, объедините их и выведите отсортированный результат.

Пример:
Входные данные: K = 3, N = 4, arr = { {1, 3, 5, 7}, {2, 4, 6, 8}, {0, 9, 10, 11}}.
Выход: 0 1 2 3 4 5 6 7 8 9 10 11
Пояснения: Выходной массив представляет собой отсортированный массив, содержащий все элементы входной матрицы.

Вход: k = 4, n = 4, arr = { {1, 5, 6, 8}, {2, 4, 10, 12}, {3, 7, 9, 11}, {13, 14, 15, 16}}.
Выходные данные: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Пояснения: Выходной массив представляет собой отсортированный массив, содержащий все элементы входной матрицы.

- **Легкий вариант:**
Iterative HeapSort. HeapSort - это метод сортировки на основе сравнения, при котором сначала строится Max Heap, а затем корневой элемент меняется местами с последним элементом (количество раз) и каждый раз сохраняется свойство кучи для окончательной сортировки.

Input : 10 20 15 17 9 21
Output : 9 10 15 17 20 21

Input: 12 11 13 5 6 7 15 5 1
Output: 5 5 6 7 11 12 13 15 19

Средний вариант:
Дано N ветвей разной длины, задача состоит в том, чтобы с минимальными затратами соединить эти веревки в одну веревку так, чтобы стоимость соединения двух веревок была равна сумме их длин.

Вход: arr[] = {4,3,2,6} , N = 4
Выходные данные: 29
Пояснения:
  - Сначала соединим веревки длиной 2 и 3. Теперь у нас есть три веревки длиной 4, 6 и 5.
  - Теперь соедините веревки длиной 4 и 5. Теперь у нас есть две веревки длиной 6 и 9.

- Наконец, соедините две веревки, и все веревки соединились.

Сложный вариант:
Дано N машин. Каждая машина содержит некоторое количество чисел в отсортированном виде. Но количество чисел, имеющихся на каждой машине, не фиксировано. Вывести числа из всех машин в отсортированном неубывающем виде.

Machine M1 contains 3 numbers: {30, 40, 50}
Machine M2 contains 2 numbers: {35, 45}
Machine M3 contains 5 numbers: {10, 60, 70, 80, 100}
Output: {10, 30, 35, 40, 45, 50, 60, 70, 80, 100}

- **Лёгкий вариант:**
Задача состоит в том, чтобы из массива arr[] размера N вывести K наибольших элементов в массиве.

Примечание: Элементы в выводимом массиве могут располагаться в любом порядке
Входные данные: [1, 23, 12, 9, 30, 2, 50], K = 3
Выходные данные: 50, 30, 23

Вход: [11, 5, 12, 9, 44, 17, 2], K = 2
Выходные данные: 44, 17

Средний вариант:
Дан массив arr[], содержащий n элементов. Задача состоит в том, чтобы найти максимальное количество отличимых элементов (неповторяющихся) после удаления из массива k элементов.

Примечание: 1 <= k <= n.
Вход : arr[] = {5, 7, 5, 5, 1, 2, 2}, k = 3
Выходные данные : 4

Удалить 2 вхождения элемента 5 и 1 вхождение элемента 2

Вход : arr[] = {1, 2, 3, 4, 5, 6, 7}, k = 5
Выход : 2

Вход : arr[] = {1, 2, 2, 2}, k = 1
Выход : 1

Сложный вариант:
Учитывая последовательность S = {1, 2, 3 принадлежащей N} найдите лексикографически наименьшее (самое раннее по порядку в словаре) отклонение из S.
Перестановкой S называется любая перестановка S, при которой ни один из двух элементов S и ее перестановки не встречаются в одной и той же позиции.

Вход: 3

Выход : 2 3 1
Пояснения: Последовательность равна 1 2 3.
Возможны следующие перестановки: (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2) (3, 2, 1).
Производные: (2, 3, 1), (3, 1, 2).
Наименьшее отклонение: (2, 3, 1)

- **Лёгкий вариант:**
  Даны массив arr[] размера N и число K, где K меньше размера массива. Найти K-й наименьший элемент данного массива. Учитывая, что все элементы массива различны.

  Вход: arr[] = {7, 10, 4, 3, 20, 15}, K = 3
  Выходные данные: 7

  Вход: arr[] = {7, 10, 4, 3, 20, 15}, K = 4
  Выход: 10

  Отсортировать входной массив в порядке возрастания
  Вернуть элемент по индексу K-1 (0 - базисная индексация) в отсортированном массиве

  Средний вариант:
  Даны два одинаковых по размеру массива (A, B) и N (размер обоих массивов). Сумма складывается из одного элемента из массива A и другого элемента из массива B. Выведите максимальное K допустимых сочетаний сумм из всех возможных сочетаний сумм.

  Вход:
  A[] : {3, 2}
  B[] : {1, 4}
  K : 2 [Количество комбинаций максимальной суммы комбинаций, которые будут выведены на печать]
  Выходные данные:
  7 // (A : 3) + (B : 4)
  6 // (A : 2) + (B : 4)

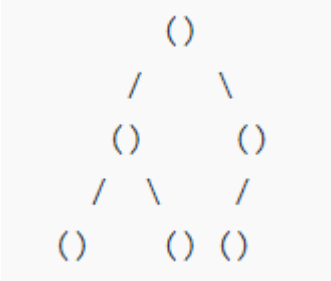
  Вход:
  A[] : {4, 2, 5, 1}
  B[] : {8, 0, 3, 5}
  K : 3
  Выход:
  13 // (A : 5) + (B : 8)
  12 // (A : 4) + (B : 8)
  10 / (A : 2) + (B : 8)
  Сложный вариант:
  Если в качестве массивов заданы две двоичные максимальные кучи, то задача состоит в том, чтобы объединить эти кучи.

  Вход: a = {10, 5, 6, 2}, b = {12, 7, 9}

Выход: {12, 10, 9, 2, 5, 7, 6}

- **Лёгкий вариант:**
  Рассмотрим двоичную кучу размера N. Требуется найти ее высоту.

  Вход : N = 6
  Выход : 2



  Средний вариант:
  В бесконечном потоке целых чисел найти K-й наибольший элемент в любой момент времени.

  Примечание: Здесь у нас не целый массив, а поток, и нам разрешено хранить только K элементов.
  Вход: stream[] = {10, 20, 11, 70, 50, 40, 100, 5, ...}, K = 3
  Выход: {_, _, 10, 11, 20, 40, 50, 50, ...}

  Вход: stream[] = {2, 5, 1, 7, 9, ...}, K = 2
  Выход: {_, 2, 2, 5, 7, ...}

  Сложный вариант:
  Дан массив целых чисел. Написать программу для нахождения K-й наибольшей суммы смежных подмассивов в массиве чисел, содержащих как отрицательные, так и положительные числа.

  Вход: a[] = {20, -5, -1}, K = 3
  Выходные данные: 14
  Пояснения: Все суммы смежных подмассивов равны (20, 15, 14, -5, -6, -1). поэтому третья по величине сумма равна 14.

  Вход: a[] = {10, -10, 20, -40}, k = 6
  Выходные данные: -10
  Пояснение: Шестая по величине сумма среди сумма всех смежных подмассивов равна -10.

- **Лёгкий вариант:**
  Задав массив элементов, отсортировать его в порядке убывания с помощью min heap.

Примеры:

Вход : arr[] = {5, 3, 10, 1}
Выход : arr[] = {10, 5, 3, 1}

Вход : arr[] = {1, 50, 100, 25}
Выход : arr[] = {100, 50, 25, 1}

Средний вариант:
Даны массив из N чисел и целое положительное число K. Задача состоит в том, чтобы найти K чисел с наибольшим числом повторений, т. е. K чисел, имеющих максимальную частоту. Если два числа имеют одинаковую частоту, то предпочтение должно быть отдано числу с большим значением. Числа должны отображаться в порядке убывания их частот. Предполагается, что массив состоит не менее чем из K чисел.

Вход: arr[] = {3, 1, 4, 4, 5, 2, 6, 1}, K = 2
Выходные данные: 4 1
Пояснения:
Частота повторения 4 – 2 раза, Частота повторения 1 – 2 раза.
Эти два числа имеют максимальную частоту повторения, и 4 больше 1.

Сложный вариант:
Дан массив из n целых положительных чисел. Требуется написать программу, выводящую минимальное произведение k целых чисел заданного массива.

Вход: [198 76 544 123 154 675], k = 2
Выход: 9348
Получаем минимальное произведение после умножения 76 и 123.

По завершению работы, составьте отчет, в котором должно быть – Ваша фамилия, имя, группа, тема работы, Ваш вариант для реализации задания, краткое описание реализации задания, ссылку на исходный код на GitHub. Исходный код push-ите в вашу ветку в соответствующем репозитории - https://github.com/FCIM-SO/Practice-Work-RU. Сохранить отчет в формате PDF и отправить на ELSE - https://else.fcim.utm.md/mod/assign/view.php?id=43457.