Київський національний університет імені Тараса Шевченка

факультет комп’ютерних наук та кібернетики

кафедра інформаційних систем

Лабораторна робота №3

d-арна піраміда

Виконав студент 2 курсу

Групи К-29

Аксой Денис Гюралпович

2018

Постановка задачі

d-арні піраміди схожі на бінарні, лише їх вузли, відмінні від листя, мають не по 2, а по d дочірніх елементів.

Необхідно представити d-арну піраміду у вигляді масиву та визначити її висоту для n елементів.

Розробити ефективні реалізації процедур extractMax, insert та increaseKey, призначених для роботи з d-арною незростаючою пірамідою.

Проаналізувати час роботи цих процедур і виразити їх в термінах n та d.

Опис алгоритму

Піраміда представлена у вигляді масиву h[0..n-1].

Дочірні елементи для вузла з індексом i мають індекси:

d i + 1, d i + 2, …, d i + d.

Для кожної вершини піраміди ключі всіх її нащадків менше або дорівнюють власному ключу вершини.

Відновлення властивостей піддерева (heapify)

Функція отримує індекс вершини i, з якої треба почати відновлення. Якщо максимальний ключ має поточна вершина i, то алгоритм припиняється, оскільки відомо, що властивості піраміди виконуються для всіх піддерев нащадків даної вершини. Інакше знаходиться нащадок з максимальним ключем з індексом p. Міняємо місцями елементи i та p. Далі функція викликається рекурсивно для вузла p.

Складність випливає із співвідношення:

Th(n) = Th([n / d]) + d.

Th(n) = O(d ).

Побудова кучі (buildHeap)

Для всіх елементів масиву, починаючи з n-2 – го визивається функція (heapify). Коли ми дійдемо до 0-го вузла, піддерева, що починаються у всіх нащадках, будуть пірамідами. Отже, якщо ми відновимо властивості для 0-го вузла, то весь вхідний масив буде пірамідою. Складність: Tb(n) = = O(n).

Видалення вузла з максимальним ключем (extractMax)

З властивостей піраміди випливає, що індекс вузла с максимальним ключем дорівнює 0. Міняємо 0-ий та n-1-ий елементи місцями. Зменшуємо розмір кучі (піраміди) на 1-цю. Визиваємо heapify для вузла 0. Складність:

Te(n) = Th(n) = O(d ).

Збільшення ключа (increaseKey)

Індекс предка вершини i визначається за формулою:

p = [(i – 1) / d].

Нехай треба замінити ключ елемента i на більше значення v. Йдемо по нащадкам вершини i доти, доки не прийшли в корінь (вершину 0) або, доки не зустріли вершину, в якій виконуються властивості піраміди, за умови, що значення її ключа буде v. Тобто, якщо значення ключа предка менше за v ми змінюємо поточний ключ на ключ предка і викликаємо цю ж функцію для предка. Складність: Tic(n) = Tic([n / d]) + 1, Tic(n) = O().

Вставка елемента (insert)

Нехай треба вставити елемент зі значенням v. Збільшимо розмір кучі на 1-цю. Припускаючи, що на останній позиції може стояти мінус нескінченність, викликаємо функцію increaseKey з параметром v для останньої вершини. Складність:

Tis(n) = Tic(n) = O().

Інтерфейс

Програмна інтерактивна. Тобто користувач вводить дані з клавіатури і одразу отримує результати запитів.

**Вхідні дані**: У 1-му рядку 2 натуральні числа n і d (n, d <= 10^5), кількість елементів у масиві та арність піраміди відповідно. У 2-му рядку n дійсних чисел – елементи майбутньої піраміди. Далі t операцій 4-ьох типів: “inc” i v – змінити ключ в позиції i на більше значення v (0 <= i <= m – 1, v >= h[i], де m – поточний розмір кучі); “ins” v – вставити в кучу елемент зі значенням v; “ex” – видалити і вивести максимальний елемент кучі (на момент виклику операції куча повинна бути не пустою); “esc” – вихід з програми.

**Вихідні дані**: Побудована з масиву піраміда заданої арності. Далі відповіді на t запитів, після кожного з яких виводиться модифікована піраміда.

Тести

Вхідні дані 1:

8 3

4 2 1 5 7 2 3 0

inc

5 5

ins

9

ex

esc

Вихідні дані 1:

Heap: 7 4 1 5 2 2 3 0

Key in position 5 increased to 5

Heap: 7 5 1 5 2 4 3 0

Inserted element 9

Heap: 9 5 7 5 2 4 3 0 1

Extracted max = 9

Heap: 7 5 1 5 2 4 3 0

Вхідні дані 2:

11 4

5 10 1 6 7 20 3 7 4 2 1

ex

inc

6 8

ins

4

ins

11

esc

Вихідні дані 2:

Heap: 20 10 2 6 7 5 3 7 4 1 1

Extracted max = 20

Heap: 10 7 2 6 7 5 3 1 4 1

Key in position 6 increased to 8

Heap: 10 8 2 6 7 5 7 1 4 1

Inserted element 4

Heap: 10 8 4 6 7 5 7 1 4 1 2

Inserted element 11

Heap: 11 8 10 6 7 5 7 1 4 1 2 4