**Київський національний університет імені Тараса Шевченка**

**Факультет комп'ютерних наук та кібернетики**

**Алгоритми і складність**

**Лабораторна робота 3**

**Звіт**

**Підготував:**

студент групи К-29

Григорович Олег Андрійович

**Київ-2019**

1. **Постановка задачі**

d-арні піраміди схожі на бінарні, тільки їх вузли, відмінні від листя, мають не по 2, а по d дочірніх елементів. Необхідно уявити d-Арну піраміду у вигляді масиву і визначити її висоту для n елементів.

Розробити ефективні реалізації процедур Extract\_Max, Insert і Increase\_Key, призначених для роботи з d-Арно незростаюча пірамідою. Проаналізувати час роботи цих процедур і висловити їх в термінах n і d.

**2. 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.

**3. Оновлення властивостей купи (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 і p. Далі функція викликається рекурсивно для вузла p.

Складність випливає зі співвідношення:

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

Th (n) = O (d \* log\_d⁡n).

**4. Побудова купи (buildHeap)**

Для всіх елементів масиву, починаючи з n-2 - го викликається 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) = Σ\_ (i = 1) ^ n▒ 〖Th (i)〗 = O (n).

Видалення вузла з максимальним ключем (extractMax)

З властивостей піраміди слідує, що індекс вузла з максимальним ключем перебування дорівнює 0. Міняємо 0-й і n-перший елементи місцями. Зменшуємо розмір купи на 1. Викликаємо heapify для вузла 0 Складність:

Te (n) = Th (n) = O (d \* log\_d⁡n).

**5. 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 нащадках вершини i до тих пір, поки не прийшли в корінь (вершину 0) або, поки не зустріли вершину, в якій виконуються властивості піраміди, за умови, що значення її ключа буде v. Тобто, якщо значення ключа предка менше v ми міняємо поточний ключ на ключ предка і викликаємо цю ж функцію для предка. Складність: Tic (n) = Tic (n / d) + 1, Tic (n) = O (log\_d⁡n).

6. Вставка елемента (insert)

Нехай треба вставити елемент зі значенням v. Збільшимо розмір купи на 1. Припускаючи, що до цього там може стояти мінус нескінченність, викликаємо функцію increaseKey з параметром v для останньої вершини. складність:

**Tis (n) = Tic (n) = O (log\_d⁡**n).