**Киевский национальный университет имени Тараса Шевченка**

**Факультет компьютерных наук и кибернетики**

**Алгоритмы и сложность**

**Лабораторный проект № 3**

**d-арные пирамиды**

**Отчет**

**Подготовил:**

студент группы К-28

Шатохин Максим Сергеевич

**Киев-2018**

1. **Постановка задачи**

d-арные пирамиды похожи на бинарные, только их узлы, отличные от листьев, имеют не по 2, а по d дочерних элементов. Необходимо представить d-Арну пирамиду в виде массива и определить ее высоту для n элементов.

Разработать эффективные реализации процедур Extract\_Max, Insert и Increase\_Key, предназначенных для работы с d-Арно невозрастающая пирамидой. Проанализировать время работы этих процедур и выразить их в терминах n и d.

1. **Описание алгоритма**

Пирамида представлена в виде массива h [0..n-1].

Дочерние элементы для узла с индексом i имеют индексы:

d \* i + 1, d \* i + 2, ..., d \* i + d.

Для каждой вершины пирамиды ключи всех ее потомков меньше или равны собственному ключу вершины.

1. **Обновление свойвств кучи(поддерева) (heapify)**

Функция получает индекс вершины i, с которой надо начать восстановление. Если максимальный ключ имеет текущая вершина i, то алгоритм прекращается, поскольку известно, что свойства пирамиды выполняются для всех поддеревьев потомков данной вершины. Иначе находится потомок с максимальным ключом с индексом p. Меняем местами элементы i и p. Далее функция вызывается рекурсивно для узла p.

Сложность вытекает из соотношения:

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

Th (n) = O (d \* log\_d⁡n).

1. **Построение кучи (buildHeap)**

Для всех элементов массива, начиная с n-2 - го вызывается функция (heapify). Когда мы дойдем до 0-го узла поддерева, начинающиеся во всех потомках будут пирамидами. Итак, если мы восстановим свойства для 0-го узла, то весь входной массив будет пирамидой. Сложность: Tb (n) = Σ\_ (i = 1) ^ n▒ 〖Th (i)〗 = O (n).

Удаление узла с максимальным ключом (extractMax)

Из свойств пирамиды следует, что индекс узла с максимальным ключом находиться равен 0. Меняем 0-й и n-первая элементы местами. Уменьшаем размер кучи на 1-это. Вызывает heapify для узла 0 Сложность:

Te (n) = Th (n) = O (d \* log\_d⁡n).

1. **Увеличение ключа (increaseKey)**

Индекс предка вершины i определяется по формуле:

p = (i - 1) / d.

Пусть надо заменить ключ элемента i на большее значение v. Идем по потомкам вершины i до тех пор, пока не пришли в корень (вершину 0) или, пока не встретили вершину, в которой выполняются свойства пирамиды, при условии, что значение ее ключа будет v. То есть, если значение ключа предка меньше v мы меняем текущий ключ на ключ предка и вызываем эту же функцию для предка. Сложность: Tic (n) = Tic (n / d) + 1, Tic (n) = O (log\_d⁡n).

1. **Вставка елемента (insert)**

Пусть надо вставить элемент со значением v. Увеличим размер кучи на 1-это. Предполагая, что до этого там может стоять минус бесконечность, вызываем функцию increaseKey с параметром v для последней вершины. сложность:

Tis (n) = Tic (n) = O (log\_d⁡n).