DAA Lab - Session 9 - Heapsort

**Transform-and-Conquer:** Implementation of Heap Sort algorithm.

**Problem Definition:** Sort an array of records (a record is a structure with an “id” and a “value” field) using Heap Sort algorithm in nondecreasing order on the “id” field of the records.

**Input:** Input begins with n (1 ≤ n ≤ 220) of number of records indicating the size of the input array. The following n lines has a record per line with an 8-digit id field and a 6-digit value field separated by a space.

**Output:** Print the sorted array. Each record is displayed in a single line with a space between the id and the value field. Finally, the last line is for the execution time of sort upto 6 decimal places.

**Sample Input:**

5

4 10

1 20

3 20

5 30

2 40

**Sample Output:**

1 20

2 40

3 20

4 10

5 30

0.000000 sec.

**Sample Input:**

3

12345678 201030

92323245 400100

53413233 800909

**Sample Output:**

12345678 201030

53413233 800909

92323245 400100

0.000000 sec.

**HeapSort(H[1..n])**

**HeapBottomUp(H[1..n])** //Construct heap

**for i ← n downto 2 do**

**H[1] ↔ H[i]** //H[1] has the max element.

**Heapify(H[1..i-1], 1)** //Sift down H[1]

**HeapBottomUp(H[1..n])**

**if(n ≤ 1) return**

**for i ← ⌊n/2⌋ downto 1 do**

**Heapify(H, i)**

**Heapify(H[1..n], k)**

**if(2\*k > n) return** //if H[k] is a leaf

**j ← 2\*k** //j points to left child of H[k]

**if(j+1 ≤ n)** //if there exists a right child of H[k]

**if(H[j+1] > H[j]) j ← j+1**

**if(H[j] > H[k])** //if greater child is greater than H[k]

**H[j] ↔ H[k]**

**Heapify(H, j)** //Heapify the subtree rooted at j

**Practice Problems:**

1. .