SNU CSE: Data Structure　　　　　　　　Due Date : 11 pm, May 24 2019

Assignment 4

Lecturer: Prof. Satti

1- Sorting

In this task, you should implement a variety of sorting algorithms and compare their performances. Here is the list you should implement:

1. heapsort, without using `heap initialization' (i.e., by inserting the numbers repeatedly into an initially empty heap)
2. heapsort using heap initialization
3. using a binary search tree (insert the numbers into an initially empty binary search tree, and report the numbers in in-order)
4. using a splay tree (as in 3)

Write a program `Sorts.java` that sorts integers by each algorithm and write each and every sorted integer into `output.txt` and prints the running time (take the average running time after repeating the sorting procedure enough number of times, to improve the accuracy of each. The input data must be randomly created in the given *length ≤ 10,000* and the range of each element must be among *1 and length* `x` *20*. Write a procedure that verifies that the output produced by a sorting algorithm is indeed sorted.

Sample Execution

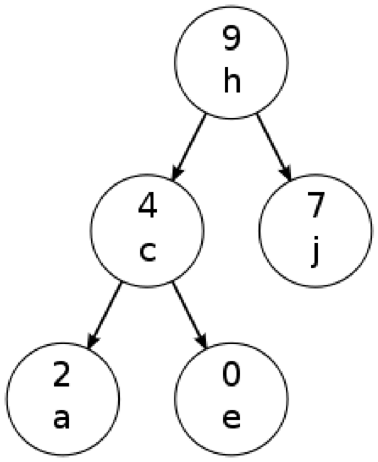
```
$ java Sort 1000

x1 ms heapsort (without heap initialization)

x2 ms heapsort

x3 ms BST sort

x4 ms splay tree sort

the sorted file is “output.txt”.
```

// (where x1, x2, x3, x4, are the times, in milliseconds, taken by the corresponding sorting algorithms for the given input).

You should experiment by changing the number of integers and analyze the result. To make these numbers non-zero for a small number of integers, repeat the sorting procedure 100 times on your input. Submit your report, justifying the above run-times, as a `pdf` file, along with your program and `output.txt`.

2- Treap Data Structure

Treap data structure combines the best of both *heaps* and *binary search trees*. When we construct a *heap*, we basically build an ordered *binary tree* and make it satisfy the *heap* property. We store pairs(X, Y) with the help of treap. X values are the keys and also Y values are called priorities. The figure shows a treap with alphabetic key and numeric max heap order.



Implement Treap Abstract Data Type in `Treap` class. Define the following operations for your class. The object is either integer or string your programme must manage to perform a suitable action based on each object.

- `void insert(Object X):`
  - Adds a new node to the tree. generate a random priority `y` for `x.` Perform binary search tree insert. Use heap operations to make sure that the inserted node's priority follows max heap.
- `search (Object X):`
  - Perform Binary search tree search to find `x`.
- `remove(Object X):`
  - Looking for a node with the specified key value `X` and removes it from the tree. You should handle remove situations based on the location of `X` (leaf, internal node, and root)
- `build (Object[] X):`
  - Builds a tree from a list of values. Assuming that given `X`s are sorted.
- `union (Treap T1, Treap T2):`

    - Merges two trees, assuming that all the elements are different.

In addition, One incomplete sample main method has been given and you should complete it, `TestTreap.java` that contains a single method `main`, so that it

- reads the input elements from the file `input.txt`, which has an integer as a key per line. You need to use `build` method to insert all the given keys at once.
- The `input.txt` is provided. The Objects we use for tests are either string or integer. You do not need to care about other types of objects and object comparison.
- The purpose of the `TestTreap` class is, performing inorder traversal of your *treap. You* ***must*** *provide Inorder traversal method.*
- A sample execution should be on a par with as follows. Due to the fact that you use a random priority your output's priority is different:

Sample Execution:

```
$ java TestTreap input.txt

Treap

(key,priority): (A , 56)  , right child  B

(key,priority): (B , 26)  , right child  D

(key,priority): (C , 7)

(key,priority): (D , 9)  left child C

(key,priority): (E , 94)  left child A , right child  F

(key,priority): (F , 75)  , right child  G

(key,priority): (G , 66)  , right child  H

(key,priority): (H , 11)



Delete C

New Treap

(key,priority): (A , 56)  , right child  B

(key,priority): (B , 26)  , right child  D
```

```
(key,priority): (D , 9)

(key,priority): (E , 94)  left child A , right child  F

(key,priority): (F , 75)  , right child  G

(key,priority): (G , 66)  , right child  H

(key,priority): (H , 11)


Delete H

New Treap

(key,priority): (A , 56)  , right child  B

(key,priority): (B , 26)  , right child  D

(key,priority): (D , 9)

(key,priority): (E , 94)  left child A , right child  F

(key,priority): (F , 75)  , right child  G

(key,priority): (G , 66)


Delete A

New Treap

(key,priority): (B , 26)  , right child  D

(key,priority): (D , 9)

(key,priority): (E , 94)  left child B , right child  F

(key,priority): (F , 75)  , right child  G

(key,priority): (G , 66)
```

```
A Not Found
```

Assessment

Your code will be judged on its correctness. Please make sure that your code runs on Linux terminal. TA will not accept any complain if your code doesn't run on Linux terminal correctly, even if it runs on other IDEs or working environment correctly.

Submission

Compress all your source (`Sorts.java, Treap.java, TestTreap.java`) and both input and output files (your codes along with its executable file and your result description(pdf)) into a single file as your-full-name_SID.zip.