Data Structures (Spring 2020)

Heap (8th Lab)

2020.05.08

Seoul National University

Database Systems Lab

Today’s Lab

- Announcement
- Min Heap Implementation
  - insert, remove, removemin, siftdown, buildheap

Announcement

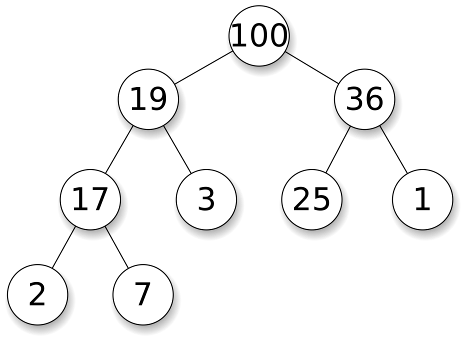
- Data Structure's lab lecture will be continued as Zoom online class
  - Due to the university's COVID-19 policy, we can operate offline class in the essential case.
  - But Data Structure's lab lecture is not an essential case.
  - If you want to offline lecture or have any opinions, please send us mail.

Announcement

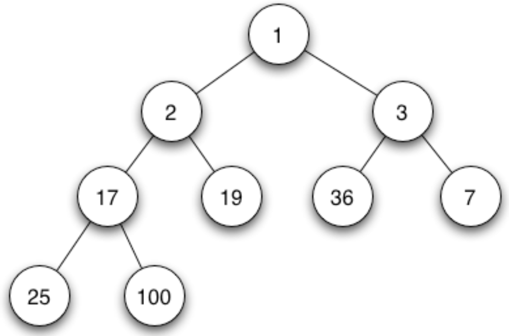
- Programming Assignment #2
  - LinearHash's insertUnique() returns the index of the hash table entry to which the word is inserted. (If it is splited, the final location would be returned.) However, **we will not check if it returns the exact location while grading**. We will only check if it returns -1 when the word exists already in the hash table.
  - LinearHash's lookup() returns **the negative value of the number** if the word is not found. Here, the number means the number of words in the hash table entry. This will be used to calculate the total number of hash table accesses.
  - CPU time which are calculated on the output file will be compared to professor's result. If one of testcases takes more than 10 (100 or 1000) times slower, you will lose 10% (25% or 50%) of the credits **only to that testcase**. For example, if one of testcases is 10 points (total points are 100 points) and you got a 15 times slower result in that testcase, you will lose 1 point on that testcase, not 10 points of total. If there are two 15 times slower results in that testcase, you will lose 2 points. **You will lose points for each result in each testcase.**
  - Grading will be finished on next week.
  - Consider the above things while submitting regrading.

Heap

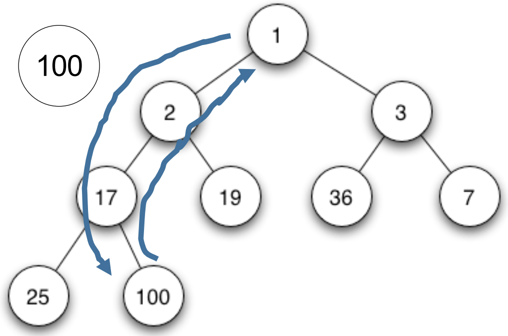
- **A heap** is a specialized **tree-based data structure**
  - essentially an almost complete tree
  - satisfies **the heap property**
    - (Min heap) For any given node C, if P is a parent node of C, then the key of P is less than or equal to the key of C.



Max heap



Min heap



Shiftdown

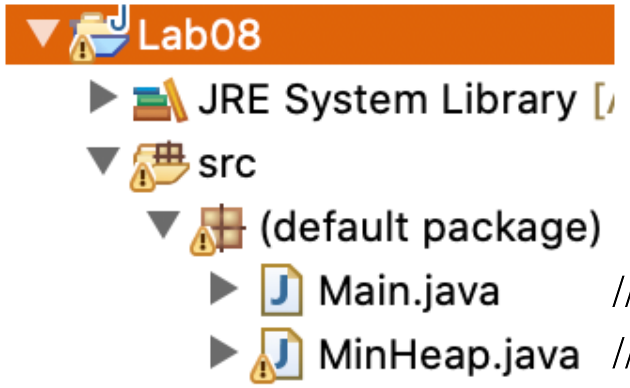
From: https://en.wikipedia.org/wiki/Binary_heap, https://en.wikipedia.org/wiki/Heap_(data_structure)

Min Heap Spec

- **void insert(int val)**
  - Insert the argument into min-heap
  - @param val the value which is inserted into min-heap
- **int removemin()**
  - Remove minimum value (equivalent to DeleteMin)
  - @return E Removed value
- **int remove(int pos)**
  - Remove element at specified position
  - @param pos the position of node which you want to remove
  - @return int Removed value
- **void siftdown(int pos)**
  - Put element in its correct place
  - @param pos the position of node which you want to siftdown
- **void BuildHeap()**
  - Heapify contents of Heap

Exercises

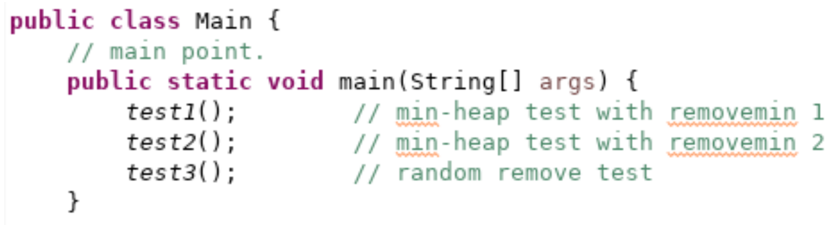
- Fill the blank of codes
  - Update your code in MinHeap.java ("// TODO: " section)
  - Write insert(), remove(), removemin(), siftdown(), buildheap(), and some of heap methods



// Main

// Min Heap Impl **(TODO)**

Project Structure

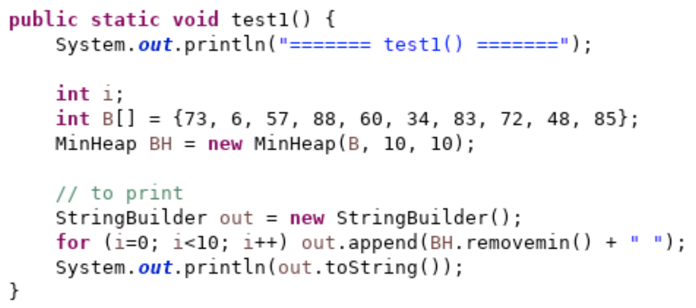


```
public class Main {
    // main point.
    public static void main(String[] args) {
        test1();        // min-heap test with removemin 1
        test2();        // min-heap test with removemin 2
        test3();        // random remove test
    }
```

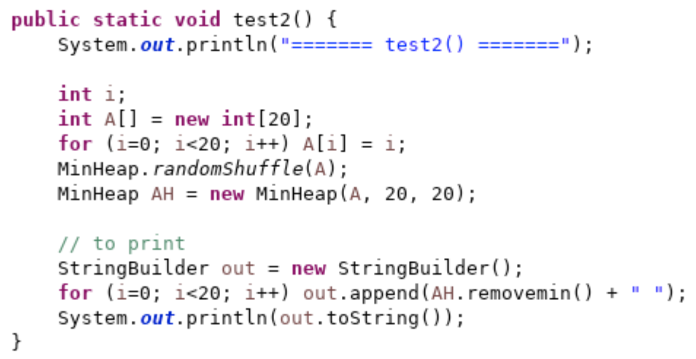
Main.java

Exercises

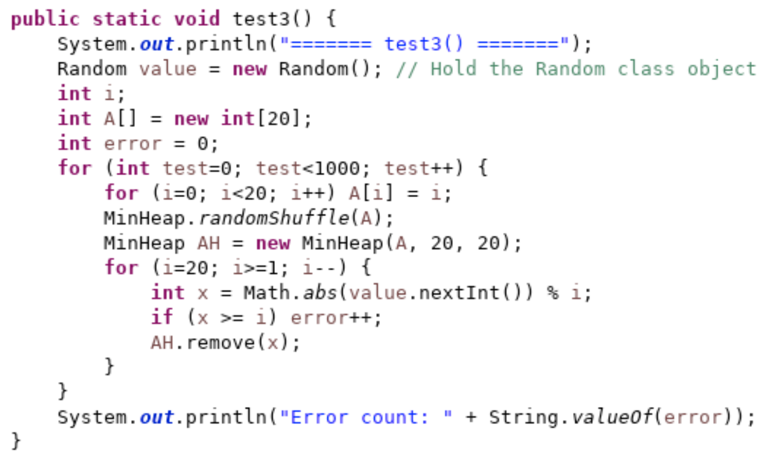
- Test cases



```java
public static void test1() {
    System.out.println("======= test1() =======");

    int i;
    int B[] = {73, 6, 57, 88, 60, 34, 83, 72, 48, 85};
    MinHeap BH = new MinHeap(B, 10, 10);

    // to print
    StringBuilder out = new StringBuilder();
    for (i=0; i<10; i++) out.append(BH.removemin() + " ");
    System.out.println(out.toString());
}
```



```java
public static void test2() {
    System.out.println("======= test2() =======");

    int i;
    int A[] = new int[20];
    for (i=0; i<20; i++) A[i] = i;
    MinHeap.randomShuffle(A);
    MinHeap AH = new MinHeap(A, 20, 20);

    // to print
    StringBuilder out = new StringBuilder();
    for (i=0; i<20; i++) out.append(AH.removemin() + " ");
    System.out.println(out.toString());
}
```



```java
public static void test3() {
    System.out.println("======= test3() =======");
    Random value = new Random(); // Hold the Random class object
    int i;
    int A[] = new int[20];
    int error = 0;
    for (int test=0; test<1000; test++) {
        for (i=0; i<20; i++) A[i] = i;
        MinHeap.randomShuffle(A);
        MinHeap AH = new MinHeap(A, 20, 20);
        for (i=20; i>=1; i--) {
            int x = Math.abs(value.nextInt()) % i;
            if (x >= i) error++;
            AH.remove(x);
        }
    }
    System.out.println("Error count: " + String.valueOf(error));
}
```

- Result

```
$ java Main
======= test1() =======
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
======= test2() =======
6 34 48 57 60 72 73 83 85 88
======= test3() =======
Error count: 0
```