Thursday, 07 September 2017

LAB DEMO 02

Quick Class Roster Check

Today, I will call 4 names that I have remembered from last week ☺

- A, B, C, D

My target today is to remember at least 4 more names:

- E, F, G, H

PS1 Debrief – Common Mistakes (1)

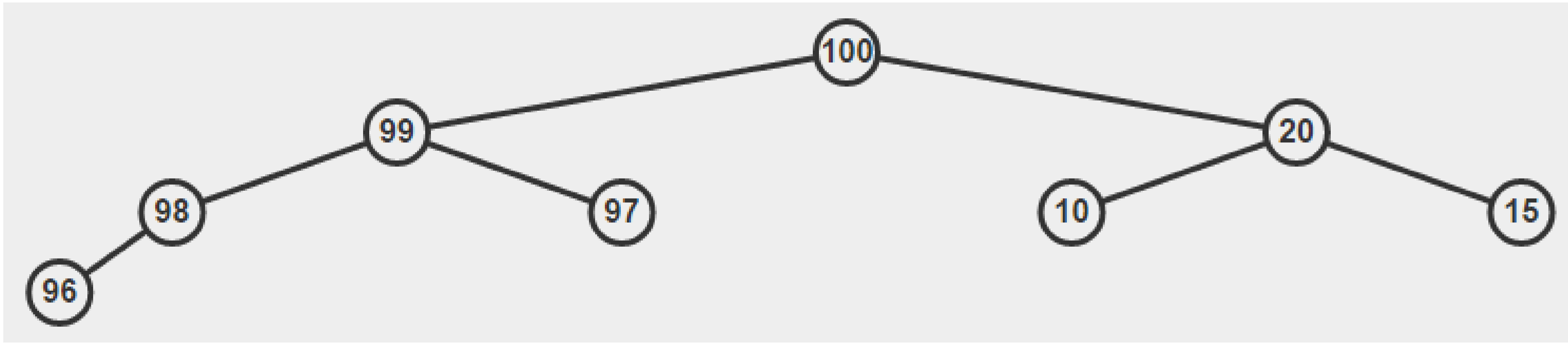
Typical common mistakes:

- TLE in C: Using **remove()** method in Java PriorityQueue for Treat or any **O(n)** like searching the entire PQ for a patient of a certain name
  - See the implementation note

PS1 Debrief – Common Mistakes (2)

Typical common mistakes:

- WA in C: Forgot this case, see below, Treat(15)



If you swap 96 (last vertex) with 15 (deleted vertex), notice that you have to do ShiftUp now, not just ShiftDown like ExtractMax :O

But the easiest implementation is IncreaseKey(15) to INF (101 for PS1), then ExtractMax ☺

PS1 Debrief – Our Answer<u>s</u> (1)

The expected solution for PS1 Subtask C

- Easiest: Use more than one bBSTs (eh? bBST??)
  - One bBST to map patient name to emergencyLvl and arrival time
  - Another bBST to emulate the PQ
  - You can simply use Java TreeMap/TreeSet (today's topic) to solve PS1… :O :O :O
    - ArriveInHospital() is a simple bBST insertion
    - Query() is a simple bBST FindMax() operation
    - Treat() is simple: search the patient and remove him/her
    - IncreaseKey(): search for the patient, delete his/her old data, reinsert his/her new data :O…

PS1 Debrief – Our Answers (2)

- Still easy: Lazy update using PQ (advanced topic, see CP3 page 148-149), very few of you use this…
- Longest to code, which most of you do: Write your own Binary Heap class (can extend from Lecture 02 copy) to do UpdateKey and Extract(any_pos) **(remember that this may entail calling either shiftUp and/or shiftDown to fix the binary heap property—a common mistake),** then use HashMap (CS1020) or TreeMap (today's topic) to map name to index

VisuAlgo Training Mode

PS2 is clearly about (Balanced) BST

Make sure that you understand the explanation in:

https://visualgo.net/bst

You can use VisuAlgo Online Quiz training mode to check your basic understanding about (Balanced) BST on “infinite” number of random questions:

https://visualgo.net/training

Introducing Java TreeSet & TreeMap

Both have the same underlying DS: Balanced BST

- http://docs.oracle.com/javase/8/docs/api/java/util/TreeSet.html
- http://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

Quick demo to explain their similarities and differences

- Using ch2_05_map_set.java from CP3 book ☺
- Source: http://cpbook.net/#downloads

Feel free to explore all other important methods of these two Java APIs that implement balanced BST

- You may use this for your other programs in the future

Case Study: Indexing City Names

Given a list of city pairs in several lines, e.g.

```
JAKARTA SINGAPORE
SINGAPORE SHANGHAI
SHANGHAI TOKYO
TOKYO SINGAPORE
TOKYO LOSANGELES
```

$\Rightarrow$

```
0 1
1 2
2 3
3 1
3 4
```

Replace these city names into integers from 0 to **V**-1

- **V** is the number of distinct city names in the list
  - (**V** = 5 in the example above)
- The first city name that you see should be given integer 0
- The <u>**next different**</u> city name is integer 1 and so on
- The same city name should be given **the same index**!

PS2 – The Patient Names Problem

Subtask A/Very Easy:

- How many names start with a certain *letter*?
- Be careful of corner case with START and END

Subtask B/Very Easy, with Java API… as explained today:

- How many names start with a certain *prefix*?

Subtask C+D/Tedious:

- Subtask B+D have the same test data, but stricter TL of 1s
- Subtask C has no RemoveSuggestion

Easy Solution for PS2 Subtask A+B

There is one method in Java TreeSet (and TreeMap) that can be **very useful** for PS2 Subtask A+B

- Method **subSet(fromKey, toKey)** in TreeSet
- Method **subMap(fromKey, toKey)** in TreeMap, or
- Near "One liner" solution with just this…

Discussion of which one to use for PS2 Subtask A+B!

- Should be subSet in TreeSet, Q: Why?

"Free" 20+50 = 70 points for those who use this hint

- Plagiarism check is off for A+B, potentially many similar code

What about PS2 Subtask C+D? (1)

There is one constraint that makes these two subtasks **super tedious**

- You have to **emulate subSet(fromKey, toKey)** of TreeSet

First of all, you BST has to be balanced ☺

- You have gone through the entire Lecture 03+04 for this
  - But AVLDemo.java is not provided ….. ☹

What about PS2 Subtask C+D? (2)

Second, your “subSet” method has to run in **O(log n)**

- Non-sublinear solutions will most probably get **TLE** !
- Hint: Scrutinize the “Rank” method that is briefly touched in Lecture 03 and discussed more in Lecture 04

You have almost 2 weeks before PS2 is due to do all these ☺