Nov 17
2024

Assignment 9

COMP 2230_02

COLTON ISLES AND KAYLEE CROCKER

THOMPSON RIVERS UNIVERSITY

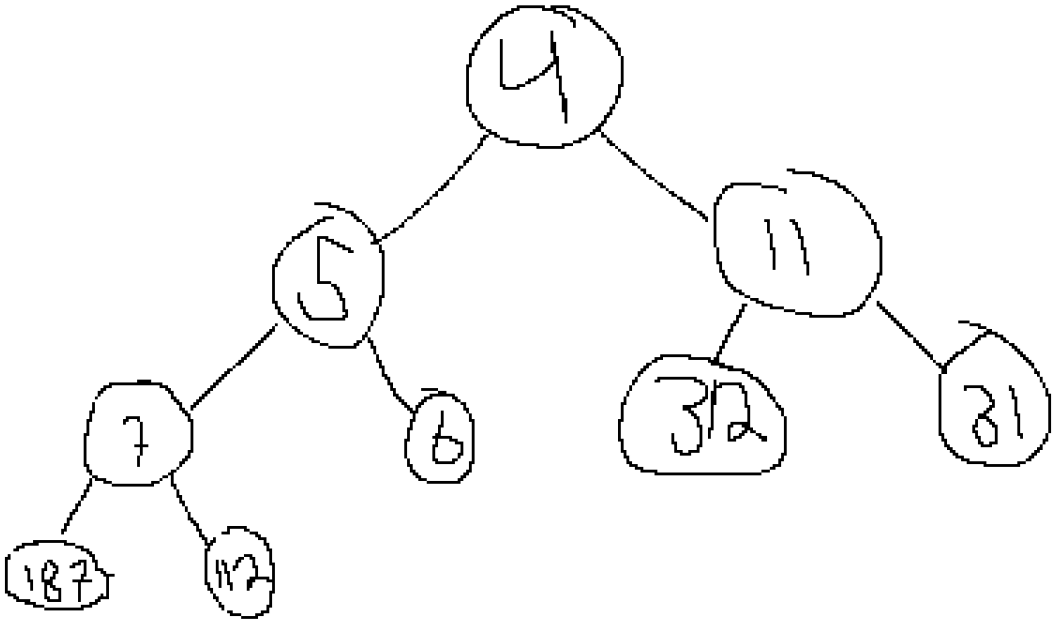
COMP 2230 – Data Structures and Algorithm Analysis

Assignment #9: Priority Queues and Hashing

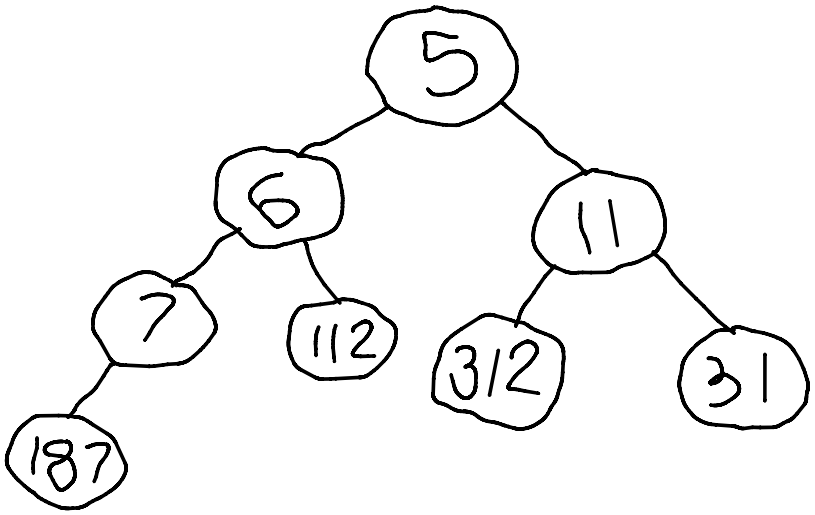
Due Date: Section 01 Nov. 21st Section 02 Nov 22nd, 2024

Chapter 21

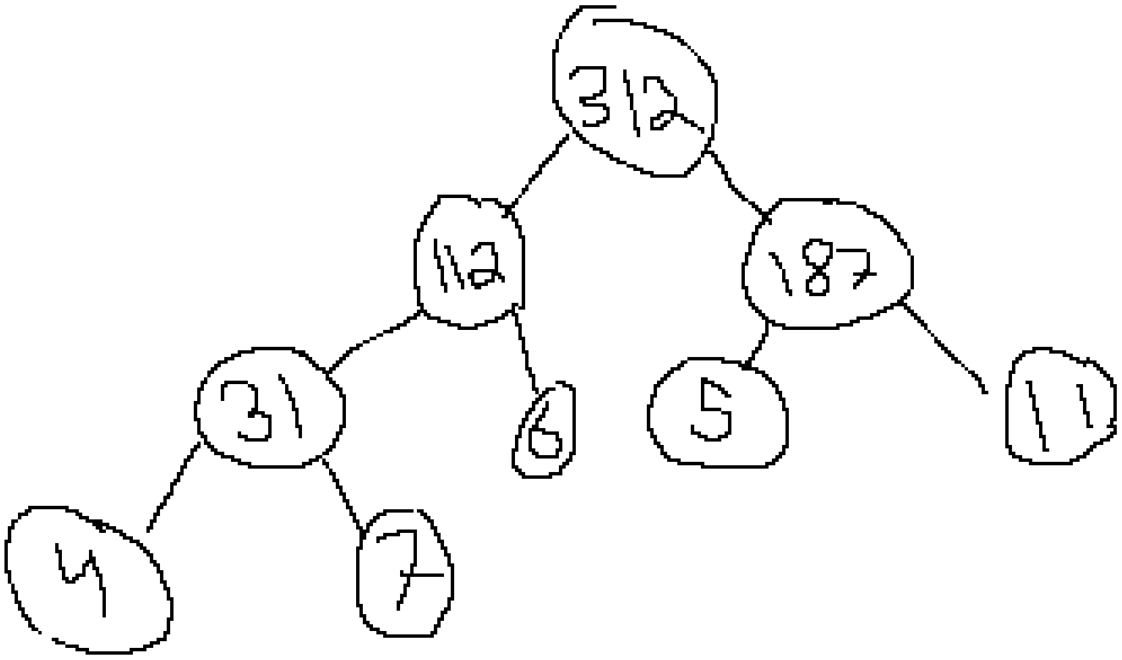
1. Draw the priority queue (based on min heap) that results from adding the following integers: 4, 5, 31, 187, 6, 312, 11, 112, 7

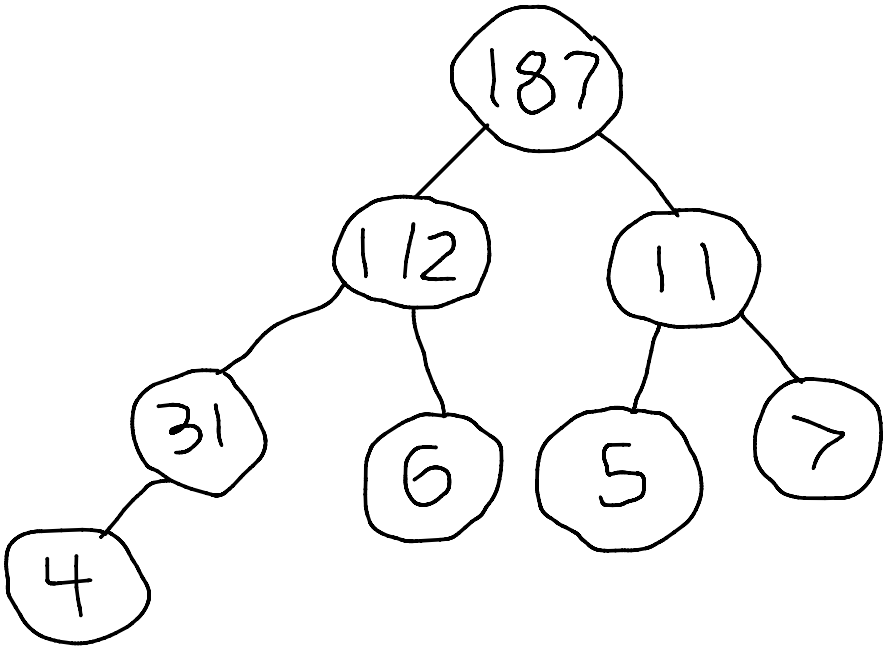


2. Starting with the previous priority queue draw the Priority Queue that results from performing a remove operation.



3. Repeat question 1 and 2, this time with a priority queue (based on max heap).





Appendix I

Problem 1:

Using a fixed size array of size 17 insert the following numbers into the table using number % table size as the hashing function. Use linear probing to resolve collisions.

10, 20, 30, 40, 50, 60, 70, 80, 90

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		70	20		90	40			60	10		80	30			50

No collisions

Problem 2:

Using a fixed size array of size 17 insert the following numbers into the table using number % table size as the hashing function. Use linear probing to resolve collisions

11, 23, 31, 43 , 53 , 61, 79, 89 , 97

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		53		89		23			43	61	11	79	97	31		

Two collisions.

Problem 3:

Is there a difference in the collision rates for problem 1 and problem 2 above?

Yes, the first problem had no collisions, whereas the second problem had two. So, problem two had a higher collision rate.

Problem 4:

Using a fixed size array of size 16 insert the following numbers into the table using number % table size as the hashing function. Use linear probing to resolve collisions.

10, 20, 30, 40, 50, 60, 70, 80, 90

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
80		50		20		70		40		10	90	60		30	

One collision.

Problem 5:

Using a fixed size array of size 16 insert the following numbers into the table using number % table size as the hashing function. Use linear probing to resolve collisions.

11, 23, 31, 43, 53, 61, 79, 89, 97

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
79	97				53		23		89		11	43	61		31

Two collisions.

Problem 6:

Is there a difference in the collision rates for problem 4 and problem 5 above?

The was a small difference. Problem 4 had one collision, and problem 5 had two. So, the number of collisions increased with a hashtable size of 16 instead of 17.

Problem 7:

Write a program to check the hash index for all numbers between 1 and 1000 using a hash table of 17 and a hash table of size 16, this program is to count the distribution of hash keys generated for each table size. i.e. how many numbers may to key 1, 2, 3 etc for table size 17 and for table size 16. **What do your results tell you?**

The keys are mostly evenly spread in both table sizes. Overall, table size 16 has more numbers mapping to each index than table size 17 (because there is less room in the table). Size 16 has roughly half of the key with 63 numbers mapped to them and half with 62. Size 17 is a more even distribution because most of the keys have 59 numbers, and only a few have 58. Taking this into account, size 17 is a better hash size choice. It is also a prime number.

HashIndexCheck.java

```
package Ass9_2230;

public class HashIndexCheck {
    public static void main(String[] args) {
        System.out.println("Size 16 hash table");
        printHashDistribution(16);
        System.out.println("Size 17 hash table");
        printHashDistribution(17);
    }

    static void printHashDistribution(int hashsize) {
        int [] keyCount = new int[hashsize];
        for(int i = 1; i <= 1000; i++){
            int key = i % hashsize;
            keyCount[key]++;
        }
        for(int i = 0; i < hashsize; i++) {
            System.out.println(i + ": " + keyCount[i] + " ");
```

```
            }
        }
}
```

Output

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intelli
Size 16 hash table
0: 62
1: 63
2: 63
3: 63
4: 63
5: 63
6: 63
7: 63
8: 63
9: 62
10: 62
11: 62
12: 62
13: 62
14: 62
15: 62
13: 62
14: 62
15: 62
Size 17 hash table
0: 58
1: 59
2: 59
3: 59
4: 59
5: 59
6: 59
7: 59
8: 59
9: 59
10: 59
11: 59
12: 59
13: 59
14: 59
15: 58
16: 58

Process finished with exit code 0
```