CSCI4470 Homework 1

Emily Sperring

August 2023

1. (10 points) Write a pseudo code that calculates frequency of a number in given array. Use the loop invariant technique i.e initialization, maintenance and termination steps and show that pseudo code calculates the frequency of a given number correctly.
   pseudocode:

   ```
   frequency(array, value) {
       int frequency = 0;
       for(int i = 0; i < array.length; i++) {
           if (array[i] == value) frequency++;
       } //for
       return frequency;
   } //frequency
   ```

   Initialization: At the beginning, frequency is initialized to 0 and i is set to 0. The loop invariant holds true because there are no elements before i, so the frequency of target in the subarray, array[0..i-1], is 0.
   Maintenance: With each iteration of the loop in the array[0...i] i is incremented by 1 and the value of a[i] is checked to see if it matches wanted value, and frequency is updated if it does at the end of the loop. This maintains the loop invariant as frequency is updated after every loop. So if array[0...i-1] is correct, then the following loops will also be correct.
   Termination: When the loop terminates, the variable i is equal to n which is array.length. All elements in the array have been examined. If $n + 1$ is substituted for i, the loop invariant holds true at this point because every time array[i] in array[0...n] has been added to frequency..

2. (10 points) Prove that $\lfloor \alpha n \rfloor + \lceil (1-\alpha)n \rceil = n$ for any integer n and real number α in the range $0 \le \alpha \le 1$

$$\begin{align} \lfloor \alpha n \rfloor + \lceil (1-\alpha)n \rceil &= n \tag{1}\\ \lfloor \alpha n \rfloor + \lceil (n-\alpha n) \rceil &= n \tag{2}\\ \lfloor \alpha n \rfloor + n + \lceil (-\alpha n) \rceil &= n \tag{3}\\ \lfloor \alpha n \rfloor + \lceil (-\alpha n) \rceil + n &= n \tag{4}\\ n &= n \tag{5} \end{align}$$

work showing $\lfloor \alpha n \rfloor + \lceil (-\alpha n) \rceil = 0$ from step 4 using the principles of ceiling and floor:

$$-\lfloor x \rfloor = \lceil -x \rceil \tag{6}$$
$$\lfloor x \rfloor + \lceil -x \rceil = 0 \tag{7}$$
$$x = \alpha n \tag{8}$$
$$\lfloor \alpha n \rfloor + \lceil (-\alpha n) \rceil = 0 \tag{9}$$
$$\tag{10}$$

3. (5 points each) Use the Big O definition to prove or disprove the following. You can assume that functions involved are asymptotically non negative functions.

   (a) $f(n) \in O(g(n)) \implies g(n) \in \Omega(f(n))$

   $f(n) \in O(g(n))$ means that $f(n) \leq cg(n)$ for some c and for $n \geq n_0$ where $c, n \geq 0$
   $g(n) \in \Omega(f(n))$ means that $g(n) \geq c_2 f(n)$ for some c_2 and for $n \geq n_0$ where $c, n \geq 0$
   $f(n) \leq cg(n)$
   $(1/c)f(n) \leq g(n)$
   $c_2 = 1/c$
   $g(n) \geq c_2 f(n)$ for $n \geq n_0$
   This proves that $f(n) \in O(g(n)) \implies g(n) \in \Omega(f(n))$ because reworking the Big O equations shows that $g(n) \geq c_2 f(n)$ for $n \geq n_0$
   Therefore, this statement is true.

   (b) $p(n) \in O(f(n)$ then $p(n)f(n) \neq O(f(n)$

   $f(n) \in O(g(n))$ means that $f(n) \leq cg(n)$ for some c and for $n \geq n_0$ where $c, n \geq 0$
   $p(n) \leq cf(n)$
   $p(n)f(n) \leq cf(n)f(n)$
   $p(n)f(n) \leq cf((n))^2$
   $cf((n))^2 \nleq cf(n)$
   An example of this is $p(n) = n^2$ and $f(n) = n^3$
   $n^2 \leq n^3$ for c=1 and $n_0 = 0$
   $(n^2)(n^3) = n^5$
   $n^5 \nleq n^3$ for any c or n_0.
   Therefore, this statement is true.

   (c) $max\{f(n), g(n)\} \in O(f(n) + g(n))$

   f(n) and g(n) are both non negative
   $max\{f(n), g(n)\}$ equals either f(n) or g(n).

$f(n) \leq f(n) + g(n)$
$g(n) \leq f(n) + g(n)$
Therefore, $max\{f(n), g(n)\} \leq f(n) + g(n)$
$f(n) \in O(g(n))$ means that $f(n) \leq cg(n)$ for some c and for $n \geq n_0$ where $c, n \geq 0$
$max\{f(n), g(n)\} \leq 1(f(n) + g(n))$ with c=1 and $n_0 = 0$
Therefore, this statement is true.

(d) $n! \in O(n^n)$

$n! = 1 * 2 * 3 * ... * (n-2) * (n-1) * n =$
$1 * 2 * 3 * ... * n \leq n * n * n * ... * n * n * n = n^n$
$n! \leq n^n$ with c=1 and $n_0 = 1$
$n! \in O(n^n)$
Therefore, this statement is true.

4. (5 points each) Prove the following :-

(a) $f(n) = 4n^2 - 50n + 10 \in o(n^3)$

$$\lim_{x\to\infty} f(n)/g(n)$$

$$= \lim_{x\to\infty} (4n^2 - 50n + 10)/(n^3)$$

$$= \infty/\infty$$

$$L'hopital Rules:$$

$$= \lim_{x\to\infty} (8n - 50)/(3n^2)$$

$$= \infty/\infty$$

$$L'hopital Rules:$$

$$= \lim_{x\to\infty} (8)/(6n)$$

$$0$$

$$f(n) = 4n^2 - 50n + 10 \in o(n^3)$$

(b) $f(n) = n^3 + 5n^2 - 5 \in \omega(n^2)$

$$\lim_{x\to\infty} f(n)/g(n)$$

$$= \lim_{x\to\infty} (n^3 + 5n^2 - 5)/(n^2)$$

$$= \infty/\infty$$

$$L'hopital Rules:$$

$$\begin{gathered}
= \lim_{x \to \infty} (3n^2 + 10n)/(2n) \\
= \infty/\infty \\
L'hopital Rules: \\
= \lim_{x \to \infty} (6n + 10)/(2) \\
= \infty \\
f(n) = n^3 + 5n^2 - 5 \in \omega(n^2)
\end{gathered}$$

5. (5 points each) Use substitution method to find the asymptotic upper bound of the following recurrence relations.

(a) $T(n) = T(n/2) + n^3$
assume: $T(n) \leq cn^3$

$$\begin{align}
T(n/2) &\leq c(n/2)^3 \tag{11} \\
T(n) &\leq c(n/2)^3 + n^3 \tag{12} \\
&= cn^3/8 + n^3 \tag{13} \\
&= cn^3 - 7cn^3/8 + n^3 \tag{14} \\
&= cn^3 - n^3(7c/8 - 1) \tag{15} \\
T(n) &= cn^3 - n^3(7c/8 - 1) \tag{16} \\
T(n) &\leq cn^3 \tag{17} \\
c &> 8/7 \tag{18} \\
n &\geq 1 \tag{19}
\end{align}$$

(b) $T(n) = 3T(n/3) + n$
assume: $T(n) \leq cnlogn$

$$\begin{align}
T(n) &\leq 3c(n/3)log(n/3) + n \tag{20} \\
&= (3cn/3)(logn - log3) + n \tag{21} \\
&= cn(logn - log3) + n \tag{22} \\
&= cnlogn - cnlog3 + n \tag{23} \\
&= cnlogn - n(clog3 - 1) \tag{24} \\
T(n) &\leq cnlogn \tag{25} \\
c &> 1/(log3) \tag{26} \\
n &\geq 1 \tag{27}
\end{align}$$

(c) $T(n) = 3T(n-1) + 1$
assume: $T(n) \leq c3^n$

$$\begin{align}
T(n) &\leq 3 * c3^{n-1} + 1 \tag{28} \\
&= 3 * c3^n * 3^{-1} + 1 \tag{29} \\
&= c3^n + 1 \tag{30}
\end{align}$$

assume now that: $T(n) \leq c_1 3^n - c_2 n$

$$\begin{aligned}
T(n) &\leq 3[c_1 3^{n-1} - c_2(n-1)] + 1 \qquad (31)\\
&= 3[c_1 3^n/3 - c_2 n + c_2] + 1 \qquad (32)\\
&= 3c_1 3^n/3 - 3c_2 n + 3c_2 + 1 \qquad (33)\\
&= c_1 3^n - 3c_2 n + 3c_2 + 1 \qquad (34)\\
&= c_1 3^n - 3c_2 n + 3c_2 + 1 \qquad (35)\\
&= c_1 3^n - c_2 n - 2c_2 n + 3c_2 + 1 \qquad (36)\\
&= c_1 3^n - c_2 n - (2c_2 n - 3c_2 - 1) \qquad (37)\\
&= c_1 3^n - c_2 n - (c_2(2n-3) - 1) \qquad (38)\\
T(n) &\leq c_1 3^n - c_2 n \qquad (39)\\
c &> 1 \qquad (40)\\
n &> 3/2 \qquad (41)
\end{aligned}$$

6. (10 points) The solution to recurrence relation $T(n) = 8T(n/2) + n^2$ is $O(n^3)$. Show that using the substitution proof with the assumption $T(n) \leq cn^3$ fails. Subtract a lower order term from the assumption and show that $T(n) \in O(n^3)$
assume that $T(n) \leq n^3$

$$\begin{aligned}
T(n) &\leq 8c(n/2)^3 + n^2 \qquad (42)\\
&= 8c(n/2)^3 + n^2 \qquad (43)\\
&= 8c(n^3/8) + n^2 \qquad (44)\\
&= cn^3 + n^2 \qquad (45)
\end{aligned}$$

nothing is being subtracted so it doesn't work
assume now that $T(n) \leq c_1 n^3 - c_2 n^2$

$$\begin{aligned}
T(n) &\leq 8[c_1(n/2)^3 - c_2(n/2)^2] + n^2 \qquad (46)\\
&= 8[c_1 n^3/8 - c_2 n^2/4] + n^2 \qquad (47)\\
&= c_1 n^3 - 2c_2 n^2 + n^2 \qquad (48)\\
&= c_1 n^3 - c_2 n^2 - c_2 n^2 + n^2 \qquad (49)\\
&= c_1 n^3 - c_2 n^2 - n^2(c_2 - 1) \qquad (50)\\
T(n) &\leq n^3 - n^2 \qquad (51)\\
c &> 2 \qquad (52)\\
n &\geq 1 \qquad (53)
\end{aligned}$$

7. (10 points) Use the recurrence tree method to solve the following recurrence relation $T(n) = 2T(n/2) + n^3$. Make sure to include a tree diagram and show all steps in your answer.

```
            n^3
           /   \
      T(n/2)    T(n/2)
```

```
             n^3
           /     \
      (n/2)^3    (n/2)^3
     /      \      /    \
T(n/4)  T(n/4) T(n/4)  T(n/4)
```

```
                  n^3     -------------------> n^3
                /      \
        (n/2)^3      (n/2)^3  ---------------> n^3/4
        /      \     /      \
    (n/4)^3  (n/4)^3 (n/4)^3  (n/4)^3 -------> n^3/16
    /   \      /  \    /  \      /  \
    .    .     .   .   .   .     .   .
    .    .     .   .   .   .     .   . -------> n^3/(4^i)
    .    .     .   .   .   .     .   .
    T(1)T(1) T(1)T(1) T(1)T(1)  T(1)T(1)
```

Subproblem size at level i is: $n/2^i$
Subproblem size hits 1 when $1 = n/2^i$: $i = logn$
Cost of the problem at level $i = n^3/4^i$

$$\begin{aligned}
T(n) &= \sum_{i=0}^{logn-1} n^3/4^i + 2^{logn} && (54)\\
&= n^3 \sum_{i=0}^{logn-1} (1/4)^i + n && (55)\\
&< n^3 \sum_{i=0}^{\infty}(1/4)^i + O(n) && (56)\\
&= n^3/(1-(1/4)) + O(n) && (57)\\
&= 4n^3/3 && (58)\\
T(n) &\in O(n^3) && (59)
\end{aligned}$$

8.

(a) (5 points each) Assume you have two fair dice. What is the expected value of a roll of these dice.

$$E[X] = E[\sum_{i=1}^{2}]X_i \tag{60}$$

$$= \sum_{i=1}^{2} E[X_i] \tag{61}$$

$$E[X_i] = \sum_{i=1}^{6} P_i X_i \tag{62}$$

$$= 1/6 * 1 + 1/6 * 2 + 1/6 * 3 + 1/6 * 4 + 1/6 * 5 + 1/6 * 6 \tag{63}$$

$$= 3.5 \tag{64}$$

$$E[X] = 3.5 + 3.5 \tag{65}$$

$$E[X] = 7 \tag{66}$$

(b) Now assume that the two dice are biased and has the following probability distributions. What is the expected value of a roll of these dice?

$$E[X] = E[\sum_{i=1}^{2}]X_i \tag{67}$$

$$= \sum_{i=1}^{2} E[X_i] \tag{68}$$

$$E[X] = E[X_1] + E[X_2] \tag{69}$$

$$E[X_1] = \sum_{i=1}^{6} P_i X_i \tag{70}$$

$$= 0.11 * 1 + 0.04 * 2 + 0.07 * 3 + 0.30 * 4 + 0.33 * 5 + 0.15 * 6 \tag{71}$$

$$= 4.15 \tag{72}$$

$$E[X_2] = \sum_{i=1}^{6} P_i X_i \tag{73}$$

$$= 0.17 * 1 + 0.28 * 2 + 0.21 * 3 + 0.07 * 4 + 0.03 * 5 + 0.24 * 6 \tag{74}$$

$$= 3.23 \tag{75}$$

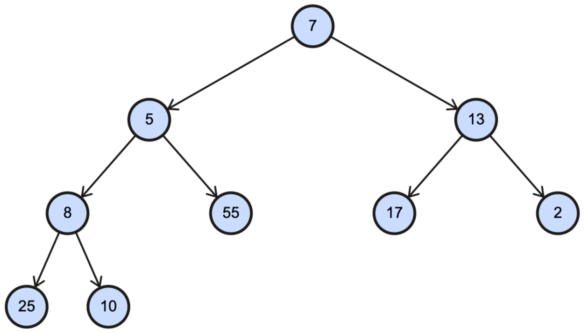
$$E[X] = 4.15 + 3.23 \tag{76}$$

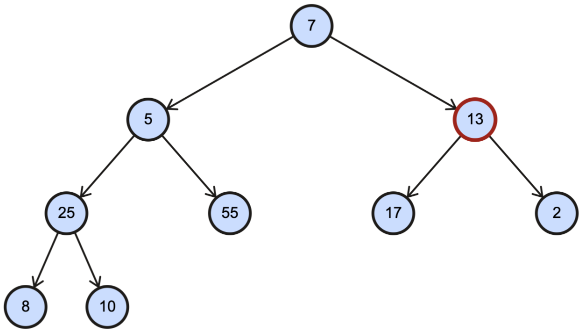
$$E[X] = 7.38 \tag{77}$$

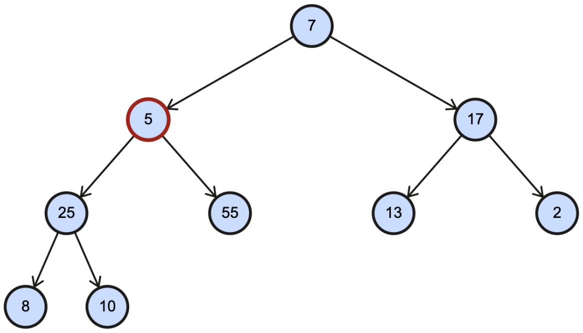
$E[X] = 7.38$ or rounded to 7

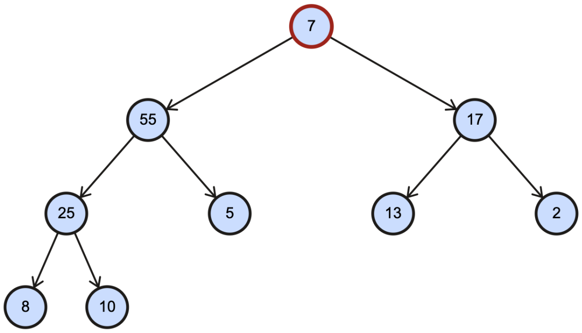
9.

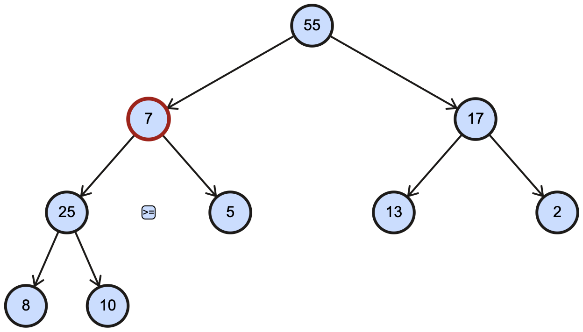
(a) (10 points) Similar to figure 6.3 illustrate the operation of BUILD-MAX-HEAP function (from the text book) on the following array A = < 7, 5, 13, 8, 55, 17, 2, 25, 10 >

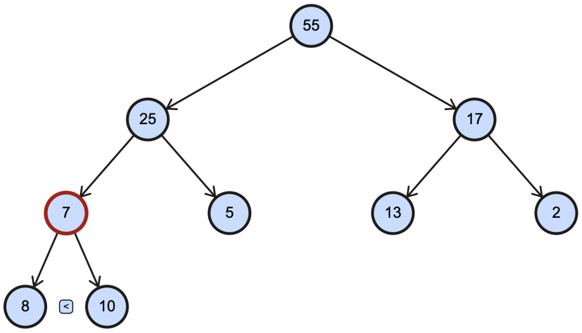


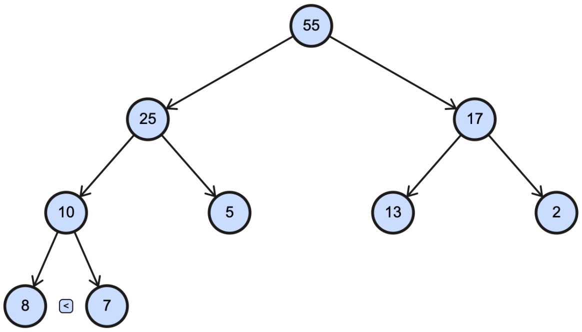




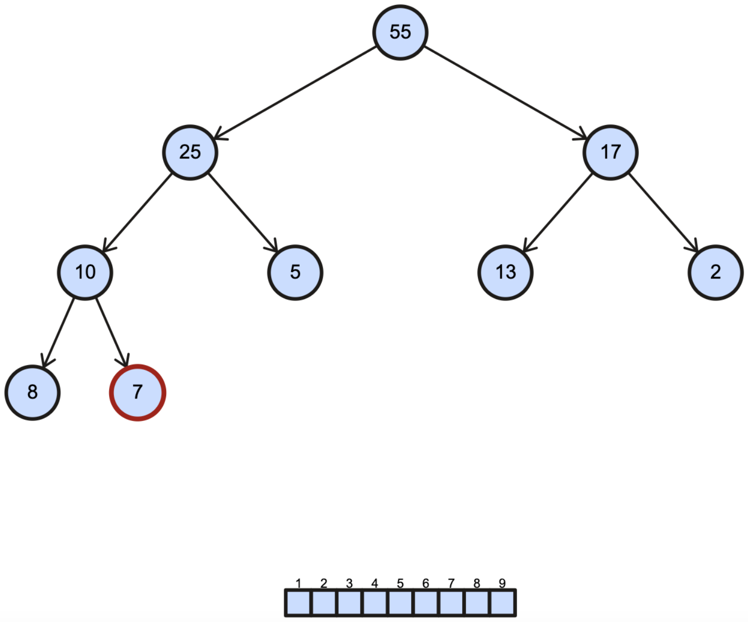

7
55
17
25
5
13
2
8
10

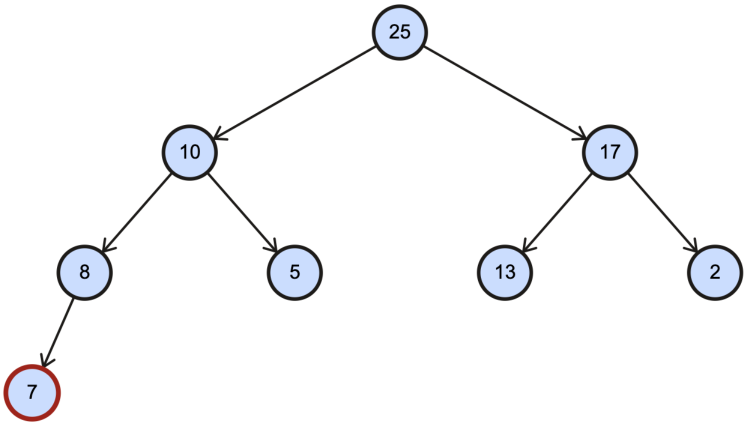

55
7
17
25
>=
5
13
2
8
10


55
25
17
7
5
13
2
8
<
10

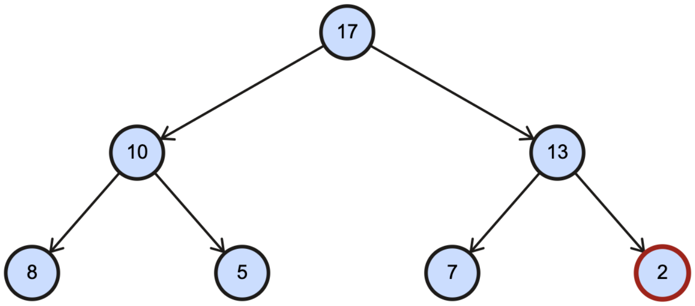


(b) 10 points) Use the heap constructed in part a and similar to figure 6.4 illustrate the operation of HEAPSORT function (from the text book) on the heap from part a
in this visualization, the nodes are removed from the heap after they are added to the final sorted array, however they are actually still there but are just swapped with the last value in the heap and then ignored in the future.

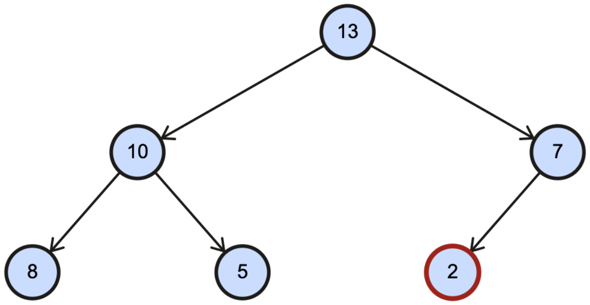




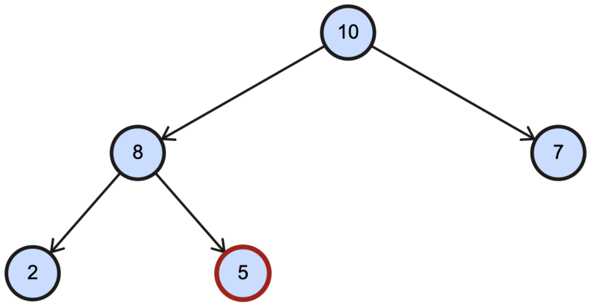
1	2	3	4	5	6	7	8	9
								55



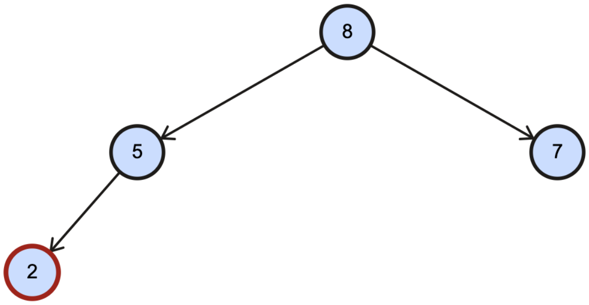
1	2	3	4	5	6	7	8	9
							25	55



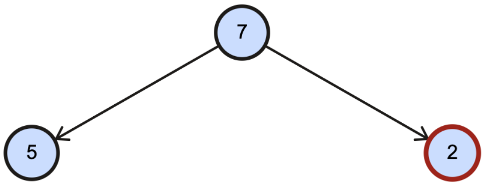
1	2	3	4	5	6	7	8	9
							17	25	55



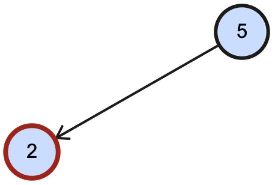
1	2	3	4	5	6	7	8	9
					13	17	25	55



1	2	3	4	5	6	7	8	9
				10	13	17	25	55



1	2	3	4	5	6	7	8	9
			8	10	13	17	25	55



1	2	3	4	5	6	7	8	9
		7	8	10	13	17	25	55

1	2	3	4	5	6	7	8	9
	5	7	8	10	13	17	25	55

1	2	3	4	5	6	7	8	9
2	5	7	8	10	13	17	25	55

10. (10 points) Assume you have k sorted lists containing a total of n items. You want to merge them into 1 sorted list. Give a pseudo code or steps of your algorithm. Using that pseudo code comment on the complexity of the algorithm. Your algorithm should use a heap to solve the problem.

Steps:
1. Create a min-heap O(1)
2. Add the first element of every list to the heap O(logk)
3. Remove the root of the heap and add that value to the new merged result list O(1)
4. Add the next element from the list that the root was taken from to the heap O(logk)
5. Repeat steps 3-4 until the heap is empty O(n)

Pseudo code done using linked lists:

```
listMerge (Node[] lists, int K){
    Node results = new node();
    min-heap heap = new min-heap();
    int current;
    for (i=0 to k) {
        heap.min-heap-insert(list[i].head);
    }
    while (!heap.empty()) {
        node min = heap.min-heap-extract-min();
        heap.min-heap-insert(min.next);
        results.last.next=min;
        results.last = min;
    }
}
```

Complexity from the steps/pseudo code: O(nlogk)
$nlogk + logk + O(1) \in O(nlogk)$