Name:- E Lovelin Anand Edward Paul Section:- 2

CS575 Homework 2
Instructor: KD Kang

Upload a scanned soft copy in pdf format through Blackboard by 9am on Feb. 27.
Submit a hard copy at the beginning of the class on Feb. 27. Please be timely.
No late submission will be accepted for this Homework.

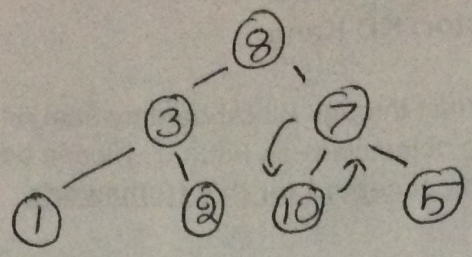
Please remember to write down Your Name, Section Number, and Sign it.

"I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment for my first offense and that I will receive a grade of "F" for the course for any additional offense."

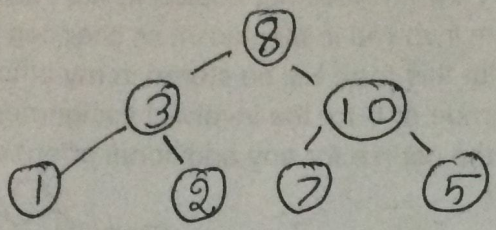
Sign:- E.Lovelin Anand

1. You are given an input array of integers: [8, 3, 7, 1, 2, 10, 5].
(a) [4 points] Build a max-heap for the input data. (No credit will be given for a min-heap.)
Draw figures to show the process.

(i)

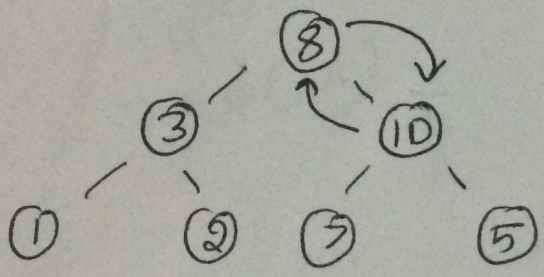


n = 3



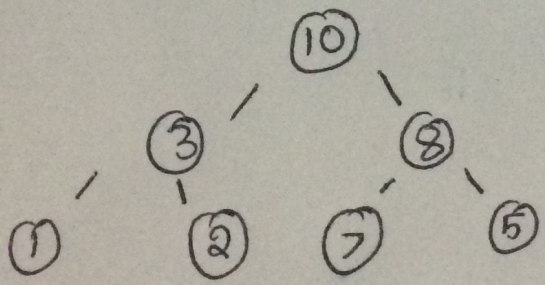
n = 2

(iii)



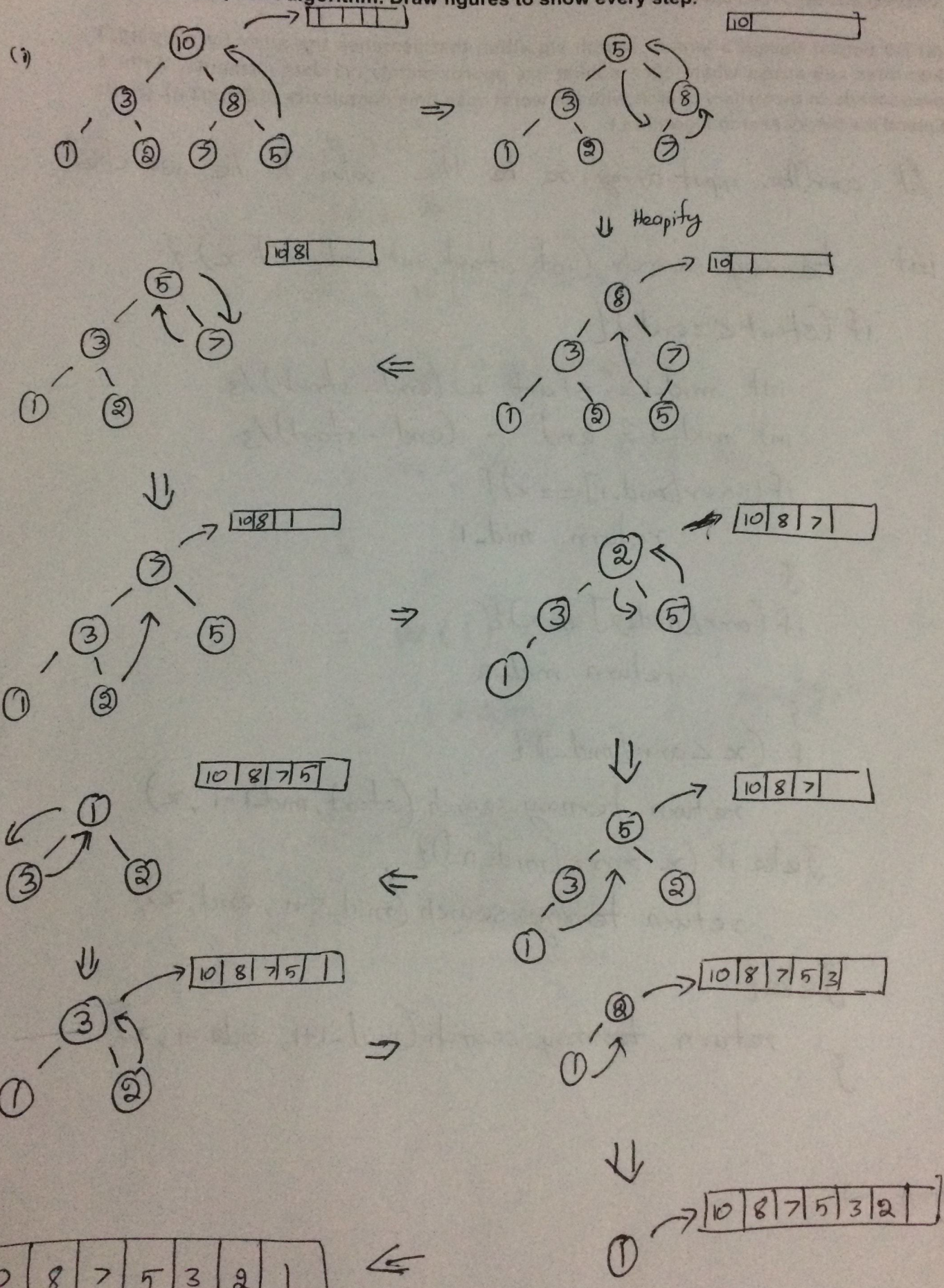
n = 1

(iv)



(b) [6 points] Using the max-heap built above, sort the input data in non-ascending order using the heap sort algorithm. Draw figures to show every step.

(i)



2. Assume that you are given an array of n elements sorted in non-descending order where n ≥ 1. Given the assumption, solve the following problem via divide and conquer.

(a) [10 points] Design a ternary search algorithm that searches the array by dividing it into three sub arrays where each sublist has approximately n/3 data elements. Write a pseudocode to do ternary search with the worst case time complexity of Θ(log3 n). (Hint: Extend the binary search algorithm.)

Let arr[] be input array, x be the value to be searched

```
int ternary_search (int start, int end, int x) {
    if (start<=end){
        int mid_1 = start + (end - start)/3
        int mid_2 = end - (end - start)/3
        if(arr[mid_1]==x){
            return mid_1
        }
        if(arr[mid_2]==x){
            return mid_2
        }
        if (x<arr[mid_1]){
            return ternary_search (start, mid_1-1, x)
        }else if (x>arr[mid_2]){
            return ternary_search (mid_2+1, end, x);
        }else{
            return ternary_search (mid_1+1, mid2-1, x);
        }
    }
return -1;
}
```

(b) [10 points] Prove that the worst case time complexity of your algorithm is Θ(log3 n). Write recursive equations and solve them iteratively. (Don't use the master theorem).

Let,

$$n = 3^k$$

$$w(1) = 1$$

$$w(n) = w(n/3) + 2$$

$$= w(n/9) + 2 + 2$$

$$= w(n/27) + 2 + 2 + 2$$

$$\vdots$$

$$= w(1) + k * 2$$

$$= 1 + 2\log_3 n \qquad [\because k = \log_3 n$$

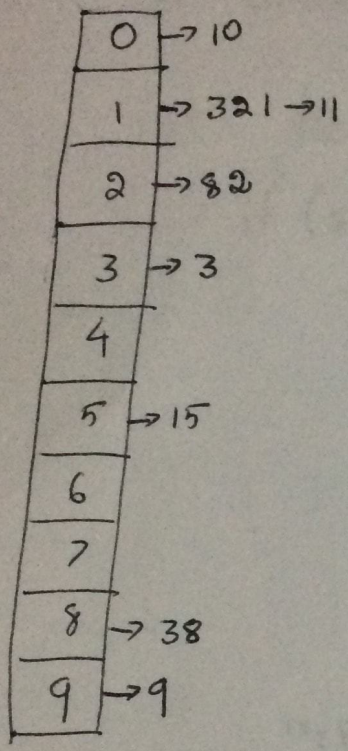
$\therefore$ ~~$\Theta(n) = \log_3$~~

$$\boxed{w(n) = \Theta(\log_3 n)}$$

3. Use the radix sort algorithm to sort the following list of numbers in non-descending order: 321, 38, 15, 3, 9, 82, 10, 11.

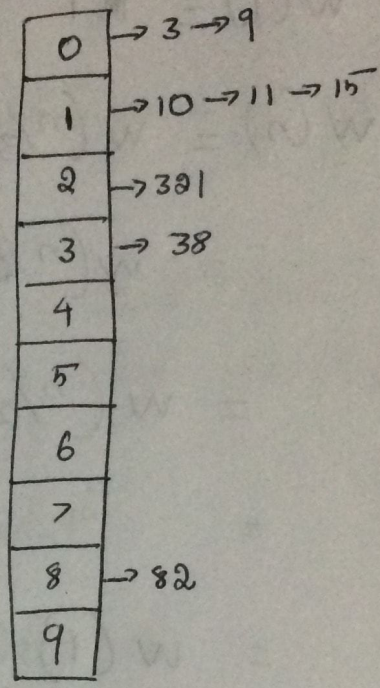
(a) Draw three figures (one for each digit) to show the process step by step [5 points].

first digit (LSD)

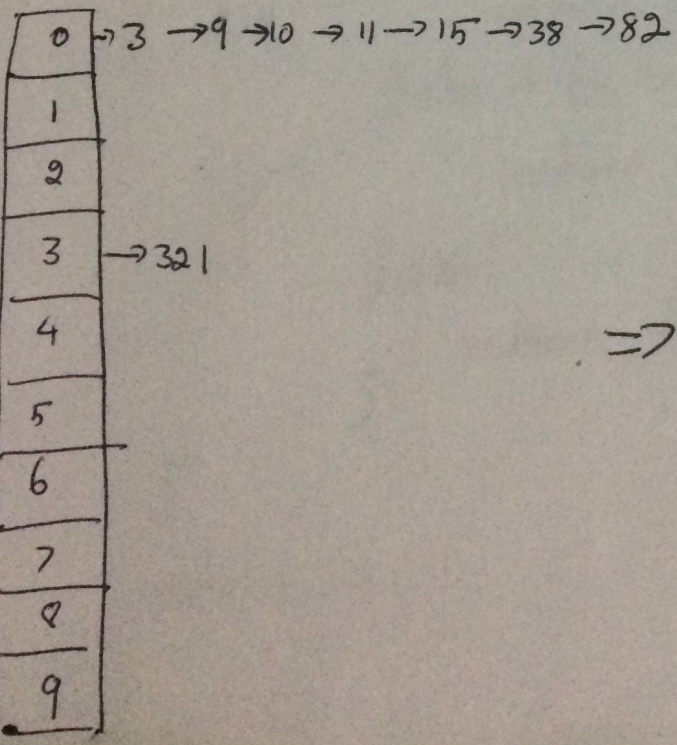


=>

tens digit



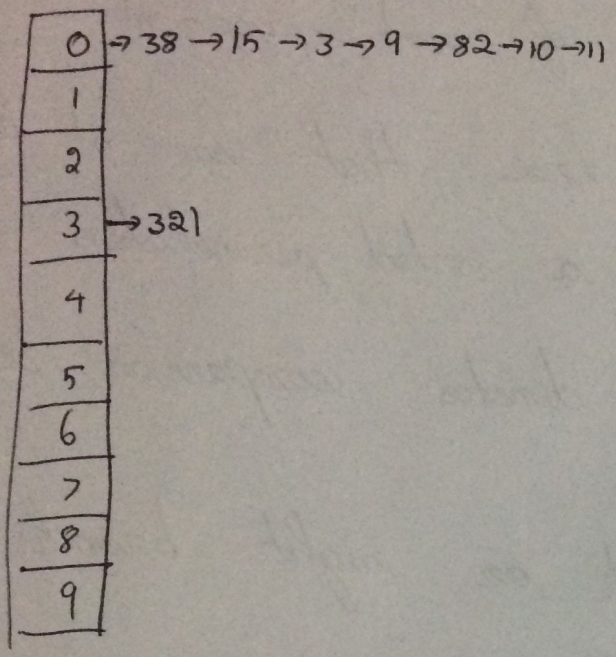
hundred digit (MSD)



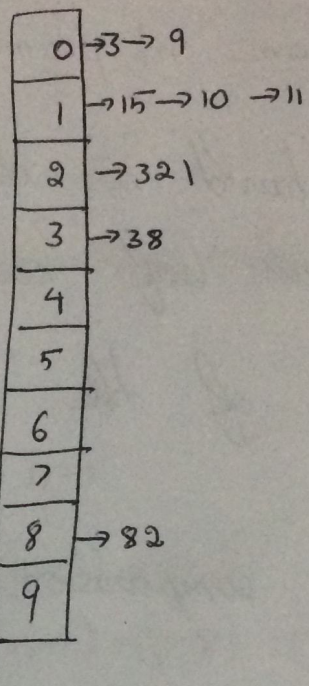
=> 3, 9, 10, 11, 15, 38, 82, 321

(b) Repeat part (a) by sorting the most significant digit first. Does it sort the numbers properly? Show the process. [5 points].

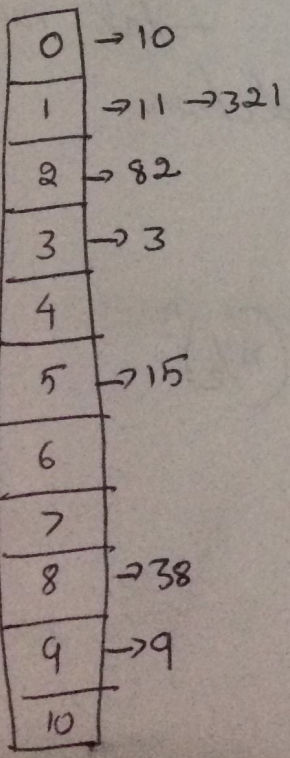
hundred digit (MSD)



tens digit



One digit (LSD)



=> 10, 11, 321, 82, 3, 15, 38, 9

which is NOT a sorted number.

So, it does not sort the numbers properly.

4. Prove that the lower bound of sorting based on comparions is Ω(nlgn) [10 points].

When a list of n integers is given as input, there are n! permatations of sort arrangement.

Let us build a decision tree that has n! leaf nodes where each leaf could be a sorted permutation.

Each node of the trees denotes comparision between specific numbers.

Based on comparision, left or right branch can be taken.

So to reach the sorted ~~no~~ leaf node, we need to traverse through ~~left~~ o each node.

The depth of decision tree indicates total number of comparisions needed to reach a sorted permutation.

Depth of tree : $\lg(n!)$

$\lg(n!) \simeq n \lg(n/k)$ $\left[n! \overset{\text{in}}{\approx} (n/e)^n\right.$

∴ sorting by comparisions is $\Omega(n \lg n)$

5. The worst case time complexity of quick sort is $O(n^2)$. Regarding this, answer the following questions.

(a) Briefly describe when the time complexity of quick sort becomes $O(n^2)$ [2 points].

The time complexity of quick sort becomes $O(n^2)$ if the pivot divides the list into two unbalanced partition as 0 and $(n-1)$ recursively. This happens if the list is already sorted (or) all the elements are equal

(b) Write recursive equations for the worst case and solve them to prove that the time complexity is $O(n^2)$ [6 points].

$$w(1) = 1$$

$$w(n) = w(n-1) + n-1$$

$$= w(n-2) + (n-2) + (n-1))$$

$$= w(n-3) + (n-3) + (n-2) + (n-1)$$

$$\vdots$$

$$= w(1) + \cdots + (n-3) + (n-2) + n-1)$$

$$= 1 + (n-1) + (n-2) + (n-3) + \cdots$$

$$= 1 + \frac{n(n-1)}{2}$$

$$\underline{\in} O(n^2)$$

(c) Briefly describe how to avoid the worst case in quick sort [2 points].

The worst case in quick sort can be avoided if the pivot is chosen randomly instead of same first or last pivot.

6. Is it a good idea to apply the divide and conquer method to compute a Fobonacci number?

(a) Simply say yes or no [1 point]. Ans -> No.

(b) Justify your yes/no answer [9 points]

```
fib(n)
    if n < 2
        return n
    else return (fib(n-1) + fib(n-2))
```

This is divide and conquer method to compute a fibonacci number. The time complexity of this method is

~~$T(n) \in O(n)$~~ $T(n) \in O(2^n)$

It is not a good idea to apply divide and conquer because there are ~~so many computations~~ repeated computation of same ~~fun~~ value. For Example: Fib(5)

Fib(5) = Fib(4) + Fib(3)

Fib(4) = Fib(3) + Fib(2)

Fib(3) = Fib(2) + Fib(1)

Fib(2) = Fib(1) + Fib(0)

∴ For Fib(5), there are 3 repeative computation.

So, it is not a good idea to apply divide and conquer.

7. Prove the following.

(a) [10 points] Prove $n^k = o(2^n)$ where k is a positive real constant.

$$g(n) = n^k$$

$$f(n) = 2^n$$

$$\lim_{n \to \infty} \frac{n^k}{2^n}$$

By L'Hopital Rule, $= \lim_{n \to \infty} \frac{k n^{k-1}}{\ln 2 \cdot 2^n}$

$$= \lim_{n \to \infty} \frac{(k)(k-1)(k-2) \cdots 1}{(\ln 2)^k \, 2^n}$$

Since numerator is constant

$$= 0$$

$$\therefore \quad n^k = o(2^n)$$

(b) [10 points] lgn = o(n).

$$\lim_{n \to \infty} \frac{\lg n}{n}$$

By L'Hopital Rule, $\lim_{n \to \infty} \frac{1/n}{1}$

$$= \lim_{n \to \infty} \frac{1}{n}$$

$$= 0$$

$$\therefore \quad \lg n = o(n)$$

(c) [10 points] $n = o(n^3)$.

$$\lim_{n \to \infty} \frac{n}{n^3}$$

By L'Hopital Rule,

$$= \lim_{n \to \infty} \frac{1}{3n^2}$$

$$= \frac{1}{\infty}$$

$$= 0$$

$\therefore \quad n = o(n^3)$