## Question One

Lecture 21, slides 22-24:

$\Theta(n+k)$ per pass in counting sort
$d$ passes = # of digits

$\Theta(d(n+k))$ total $\rightarrow$ assume $k=O(n) \rightarrow \Theta(dn)$

$\Theta\left(\frac{b}{r}(n+2^r)\right) \rightarrow$ still linear

$k = 2^r - 1$, where $r \sim \log(n)$

$d = \frac{b}{r}$ $\quad \Theta(dn) = \Theta\left(\frac{b}{r}(n+2^r)\right)$

$= \Theta(d(n+2^r))$ $\quad 2^r - 1 \sim 2^r \sim k$

$= \Theta(d(n+2^r))$

$= \Theta(d(n+k))$ $\quad n \sim k$

$\Theta(dn) = \Theta(dn)$

Count-sort is $O(n)$ and linear and radix sort is count-sort done the number of times of the number of digits in each number in the input array $\therefore$ factor x linear is still linear since there are no comparison operations.

# Question Two

```
// make one max-heap and one min-heap from
// the values of the given set

minHeap = S.makeMinHeap; // uses O(nlogn) time
maxHeap = S.makeMaxHeap;

// use an ArrayList indices for heap (global variables)
ArrayList<Integer> min = new ArrayList<Integer>();
ArrayList<Integer> max = new ArrayList<Integer>();
HashSet<Integer> list1 = new HashSet<Integer>();
HashSet<Integer> list2 = new HashSet<Integer>();
 min.add(1);
for (int i: minHeap) {
     min.add(i); list1.add(i);
}
max.add(-1);
for (int i: maxHeap) {
    max.add(i); list2.add(i);
}
// populate each array starting from 1
// default position 0 as -1

public int min() {
   // find the min, store it
   int minimum = min.get(1);
   // remove min by putting last element in root
   // position and then downheap
```

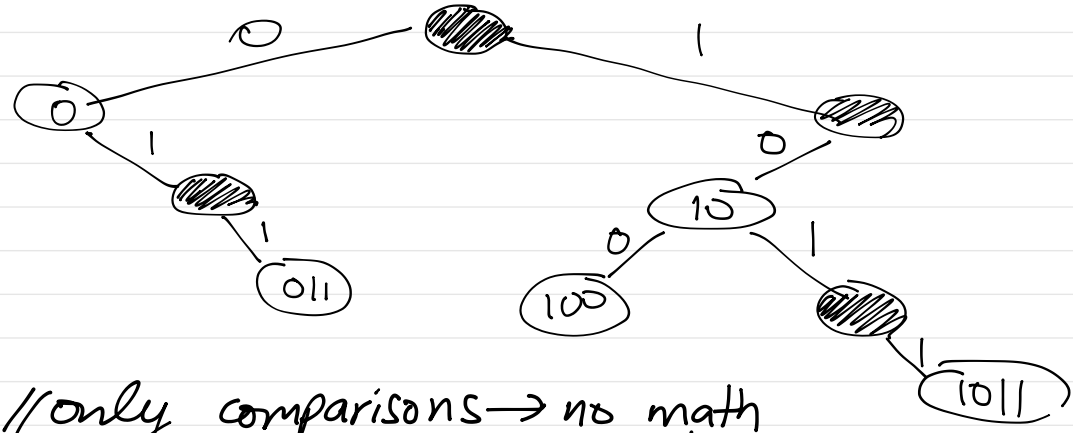
```
min.set(1, min.get(min.size()-1));
list1.remove(min.get(min.size()-1))
min.remove(min.get(min.size()-1));
int i=1;
while (min.get(2i) != null) {
  if (min.get(i) > min.get(2i) && min.get(i) > min.get(2i+1)){
      if (2i < 2i+1) {
          int temp= min.get(2i);
          min.set(2i, min.get(i));
          min.set(i, temp);
           i = 2i
      } else {
          int temp = min.get(2i+1);
          min.set(2i+1, min.get(i));
          min.set(i, temp);
           i = 2i+1;
      }
  } else if (min.get(i) > min.get(2i)) {
      int temp = min.get(2i);
      min.set(2i, min.get(i));
       min.set(i, temp);
        i = 2i;
  } else {
      int temp = min.get(2i+1);
      min.set(2i+1, min.get(i));
      min.set(i, temp);
       i = 2i+1;
  }
}
return minimum;
}
```

```
public int max() {
  // find the max, store it
  int maximum = max.get(1);
  // remove max by putting last element in root
  // position, and then dosinheap
  max.set(1, max.get(max.size()-1));
  list2.remove(max.get(max.size()-1));
  max.remove(max.get(max.size()-1));
  int i=1;
  while (max.get(2i) != null) {
    if (max.get(i) < max.get(2i) && max.get(i) < max.get(2i+1)){
        if (2i > 2i+1)
            int temp = max.get(2i);
            max.set(2i, max.get(i));
            max.set(i, temp);
             i = 2i
      } else {
            int temp = max.get(2i+1);
            max.set(2i+1, max.get(i));
            max.set(i, temp);
             i = 2i+1;
      }
    } else if (max.get(i) < max.get(2i)) {
        int temp = max.get(2i);
        max.set(2i, max.get(i));
        max.set(i, temp);
         i = 2i;
    } else {
        int temp = max.get(2i+1);
```

```
            max.set(2i+1, max.get(i));
            max.set(i, temp);
            i = 2i+1;
        }
    }
    return maximum;
}
public void insert(X){ // for minHeap
    // add the element to the last spot and then upheap
    min.add(X); //ArrayList auto-adds to last spot

    if (list1.contains(X) == false){
        while (i > 1) {
            if (min.get(i/2) > min.get(i)) {
                int temp = min.get(i);
                list2.add(temp);
                min.set(i, min.get(i/2));
                min.set(i/2, temp);
                i = i/2;
            }
        }
    }
    //change heap back to a Set
    Set S = new Set(minHeap);
}

public void insert(X){ // for maxHeap
    // add the element to the last spot and then upheap
    max.add(X); //ArrayList auto-adds to last spot
```

```
if (list2.contains(x) == false){
    while (i > 1) {
        if (max.get(i/2) > max.get(i)) {
            int temp = max.get(i);
            list2.add(temp);
            max.set(i, max.get(i/2));
            max.set(i/2, temp);
            i = i/2;
        }
    }
}
//change heap back to a Set
Set S = new Set(maxHeap);
}

//only swap comparisons = O(log n) operations
```

# Question Three




//only comparisons → no math operations

↳ O(n)

```
class Node {
    String number;
    Node nextLeft;
    Node nextRight;

    public Node(String number)
      this.number = number;
    }
    public static ArrayList<String> sort(Set S) {
        String word = S.get(0).charAt(0);
        Node root = new Node("black");
        if(S.get(0).charAt(0) == 0) {
            if(S.contains(word) == true) {
                Node next = new Node(word);
                next = root.nextLeft;
            }else {
                Node next = new Node("black");
                next = root.nextLeft;
```

```
} else {
    if( S.contains(word)== true) {
      Node next = new Node(word);
      next = root.nextRight;
    }else {
      Node next = new Node("black");
      next = root.nextRight;
    }
}
for(int k=1, k< S.size(); k++){
    for (int j=1 ; j < S.get(k).length(); j++) {
      word= word + S.get(k).charAt(j);
      if (S.get(k).charAt(j)==0) {
        if(S.contains(word)== true) {
          next = new Node(word);
          next = next.nextLeft;
        }else {
          next = new Node("black");
          next = next.nextLeft;
        }
      } else {
        if( S.contains(word)== true) {
          next = new Node(word);
          next = next.nextRight;
        }else {
          next = new Node("black");
          next = next.nextRight;
        }
}}}
```

```
ArrayList<String> order = new ArrayList<String>();
String n = root.nextLeft.number;
order.add(n);

if(!(n.equals("black"))) {
    order.add(n);
    n = root.nextRight.number;
}
if(!(n.equals("black"))) {
    order.add(n);
    n = n.nextLeft.number;
}

while (n != null) {
    if(!(n.equals("black"))) {
        order.add(n);
        n=n.nextRight.number;
    }
    if(!(n.equals("black"))) {
        order.add(n);
        n= n.nextLeft.number;
    }
}

return order;
}
```

Since $a=2, b=2 \rightarrow$ Master Theorem Case #1.

## Question Four:

Base case: $T_0 = T_1 = 1$

For $n \geq 4$:

$$T_{n+1} = \sum_{k=0}^{n} T_k T_{n-k}$$

try 6

$$T_6 = T_0 T_5 + T_1 T_4 + T_2 T_3 + T_3 T_2 + T_4 T_1 + T_5 T_0$$

apply base case:

$$T_4 = T_5 + T_4 + T_2 T_3 + T_3 T_2 + T_4 + T_5$$

$$T_6 = 2T_4 + 2T_5 + 2T_2 T_3$$

Try 6

$$T_7 = T_0 T_6 + T_1 T_5 + T_2 T_4 + T_3 T_3 + T_4 T_2 + T_5 T_1 + T_6 T_0$$

$$2T_6 + 2T_5 + 2T_2 T_4 + 1\, T_3 T_3$$

when $n$ = odd

$$2\left(T_{n-1} + T_{2n}\left(T_{n\ldots}\right)\right)$$

$$T_{n+1} = 2T_n + 2T_{n-1} + 2T_{n-2}T_{n-3} \ldots$$

when $n$ = even

$$T_{n+1} = 2T_n + 2T_{n-1} + 2T_{n-2}T_{n-3} \ldots + \left(T_{n/2}\right)^2$$

$T_n > 2^n$ $\qquad$ $T_5 > 2^5$

$T_{n+1} = \sum_{k=0}^{n} T_k T_{n-k}$ $\qquad$ $> 32$

$T_5 = T_{4+1}$

$T_{4+1} = T_0T_4 + T_1T_3 + T_2T_2 + T_3T_1 + T_4T_0$

$= 2T_0T_4 + 2T_1T_3 + (T_2)^2$

$= 2T_4 + 2T_3 + T_2^2$

$= 2(14) + 2(5) + (2)^2 = 28 + 10 + 4$

$T_4 = T_{3+1}$ $\qquad = 42$

$n=3$

$T_{3+1} = T_0T_3 + T_1T_2 + T_2T_1 + T_3T_0$

$T_{3+1} = 2T_0T_3 + 2T_1T_2$

$T_{3+1} = 2T_3 + 2T_2$ $\qquad 2(5) + 2(2) = 14$

$n=2$

$T_3 = T_{2+1}$

$T_{2+1} = T_0T_2 + T_1T_1 + T_2T_0$

$= 2T_0T_2 + (T_1)^2$

$= 2T_2 + 1$ $\qquad 2(2)+1 = 5$

$n=1$

$T_2 = T_{1+1}$

$T_{1+1} = T_0T_1 + T_1T_0$

$T_2 = 2$

$T_2 = 2$

$T_3 = 5$

$T_4 = 14$

$T_5 = 42$

$T_5 > 2^n$

$42 > 32$

for $n \geq 5$

or for $n \geq 4$

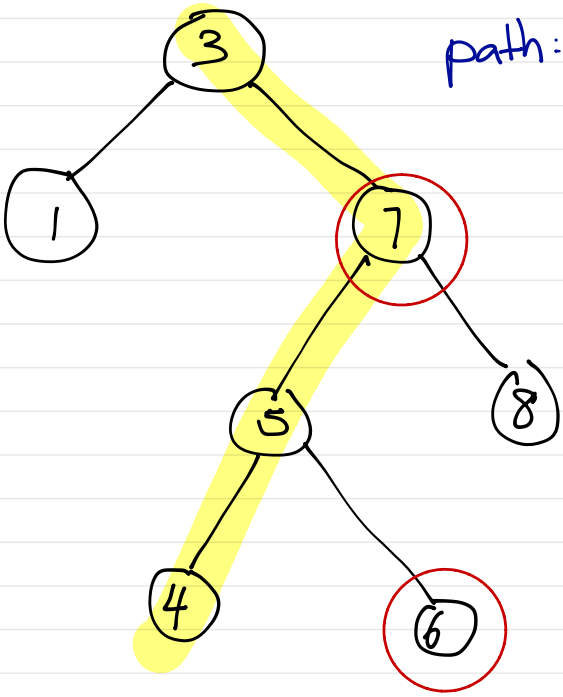
for $T_{n+1} = \sum_{k=0}^{n} T_k T_{n-k}$

$T_n = \Omega(2^n)$

$T_n > 2^n$

↳ lower limit

# Question Five




path: search for 4

| Set A | Set B | Set C |
|---|---|---|
| a = 1 | b = 3 | c = 6 |
| | b = 7 | c = 8 |
| | b = 5 | |
| | b = 4 | |

Possible combinations:

| | | | | | |
|---|---|---|---|---|---|
| a=1 | < | b=3 | < | c=6 | |
| a=1 | < | b=7 | > | c=6 | |
| a=1 | < | b=5 | < | c=6 | |
| a=1 | < | b=4 | < | c=6 | |

| | | | | | |
|---|---|---|---|---|---|
| a=1 | < | b=3 | < | c=8 | |
| a=1 | < | b=7 | < | c=8 | |
| a=1 | < | b=5 | < | c=8 | |
| a=1 | < | b=4 | < | c=8 | |