

## Q1

---

Let  $n$  be the number of edges and  $d$  be the sum of the degree.

Base case.

$n = 1$ , there is only 1 edge, thus the total degree is 2, thus  $d = 2n$ .

Suppose  $n = i$ ,  $d = 2i$

When  $n = i + 1$ , we can select an edge  $u - v$  from the undirected graph, removing this edge from the graph, now it has  $i$  edges and the remaining graph has  $2i$  total degree, now add the edge  $u - v$ , only the degree of  $u$  and  $v$  increases 1, other nodes's degree doesn't change, thus now  $d = 2i + 2 = 2(i + 1)$ .

From above, we have proved that  $d = 2n$ .

## Q2

---

2.a.i  $O(\log(N))$

2.a.ii  $O(N)$

2.a.iii  $O(|V|^2)$

2.a.iv  $O(N)$

2.b.i  $O(N^2)$

2.b.ii  $O(N^2)$

2.c.i  $O(N^2)$

2.c.ii  $O(N)$

## Q3

---

### 3.a

$$h(29) = 29 \% 11 = 7$$

$$h(7) = 7 \% 11 = 7 \quad 7 + 1^2 = 8$$

$$h(18) = 18 \% 11 = 7 \quad (7 + 2^2) \% 11 = 0$$

$$h(6) = 6 \% 11 = 6$$

$$h(19) = 19 \% 11 = 8 \quad 8 + 1^2 = 9$$

$$h(33) = 33 \% 11 = 0 \quad (0 + 1^2) = 1$$

$$h(55) = 55 \% 11 = 0 \quad (0 + 2^2) = 4$$

Index	0	1	2	3	4	5	6	7	8	9	10
Key	18	33			55		6	29	7	19	

### 3.b

33 would be no longer findable using the contains method.

Because location 0 set to NULL. When we want to find 33, we first lcoate in  $33 \% 11 = 0$ , but location 0 is null, thus return empty.

### Q4

```
int[] merge(int[] a, int[] b, int[] c) {
    int[] d = new int[a.length + b.length + c.length];

    int i = 0;

    int u = 0;
    int v = 0;
    int w = 0;

    int x;
    int y;
    int z;

    while (i < d.length) {
        if (u < a.length) {
            x = a[u];
        } else {

            x = Integer.MAX_VALUE;
        }

        if (v < b.length) {
            y = b[v];
        } else {

            y = Integer.MAX_VALUE;
        }
    }
}
```

```

    if (w < c.length) {
        z = c[w];

    } else {

        z = Integer.MAX_VALUE;
    }

    int minv;

    if (x <= y && x <= z) {
        u++;
        minv = x;
    } else if (y <= x && y <= z) {
        v++;
        minv = y;
    } else {

        w++;
        minv = z;
    }

    d[i] = minv;

    i++;
}
}

return d;
}

```

## Q5

```

// Q5
int countInternal(TreeNode root)
{
    if (root == null)
    {
        return 0;
    }

    if (root.firstChild == null)
    {
        return countInternal(root.nextSibling);

    }else {

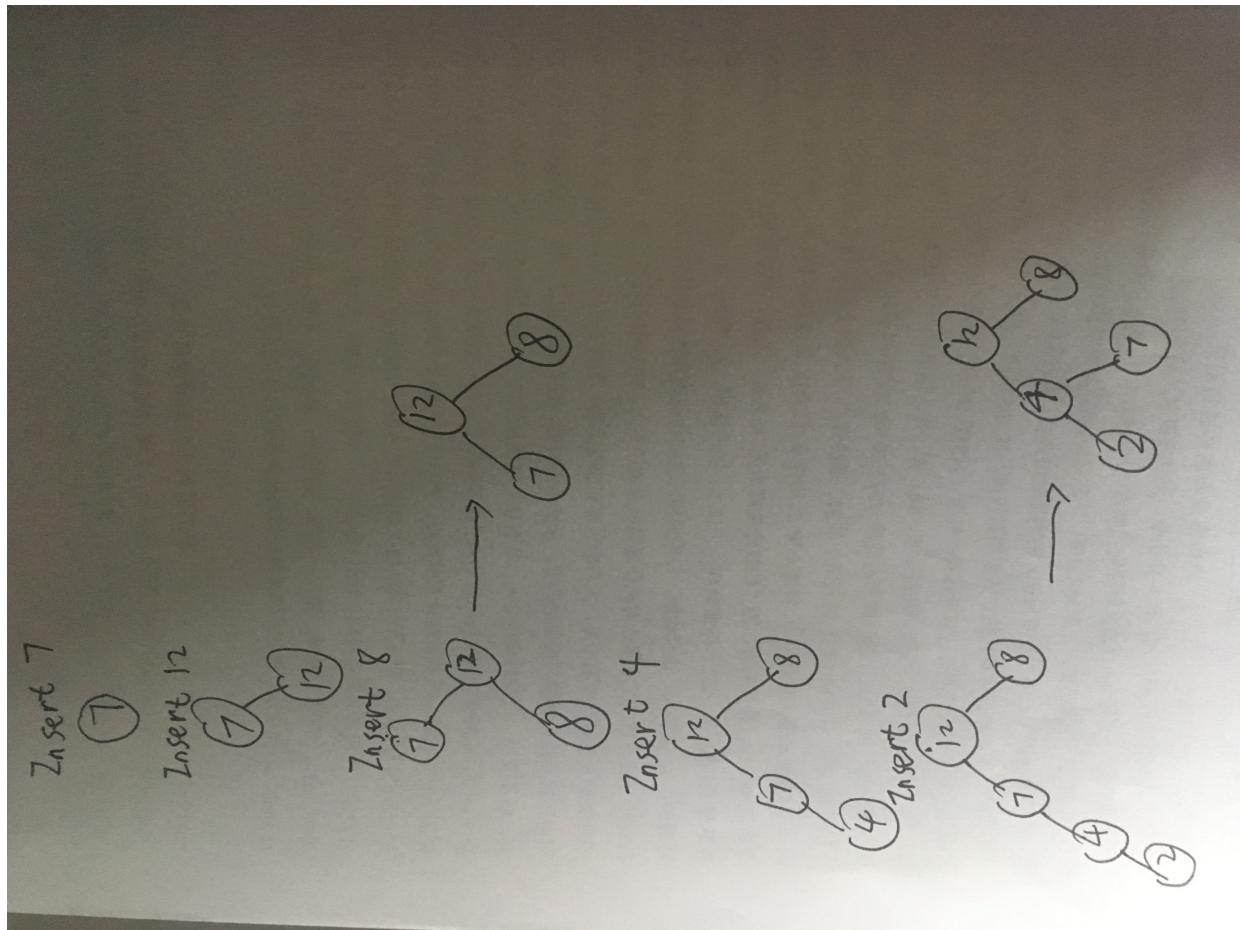
```

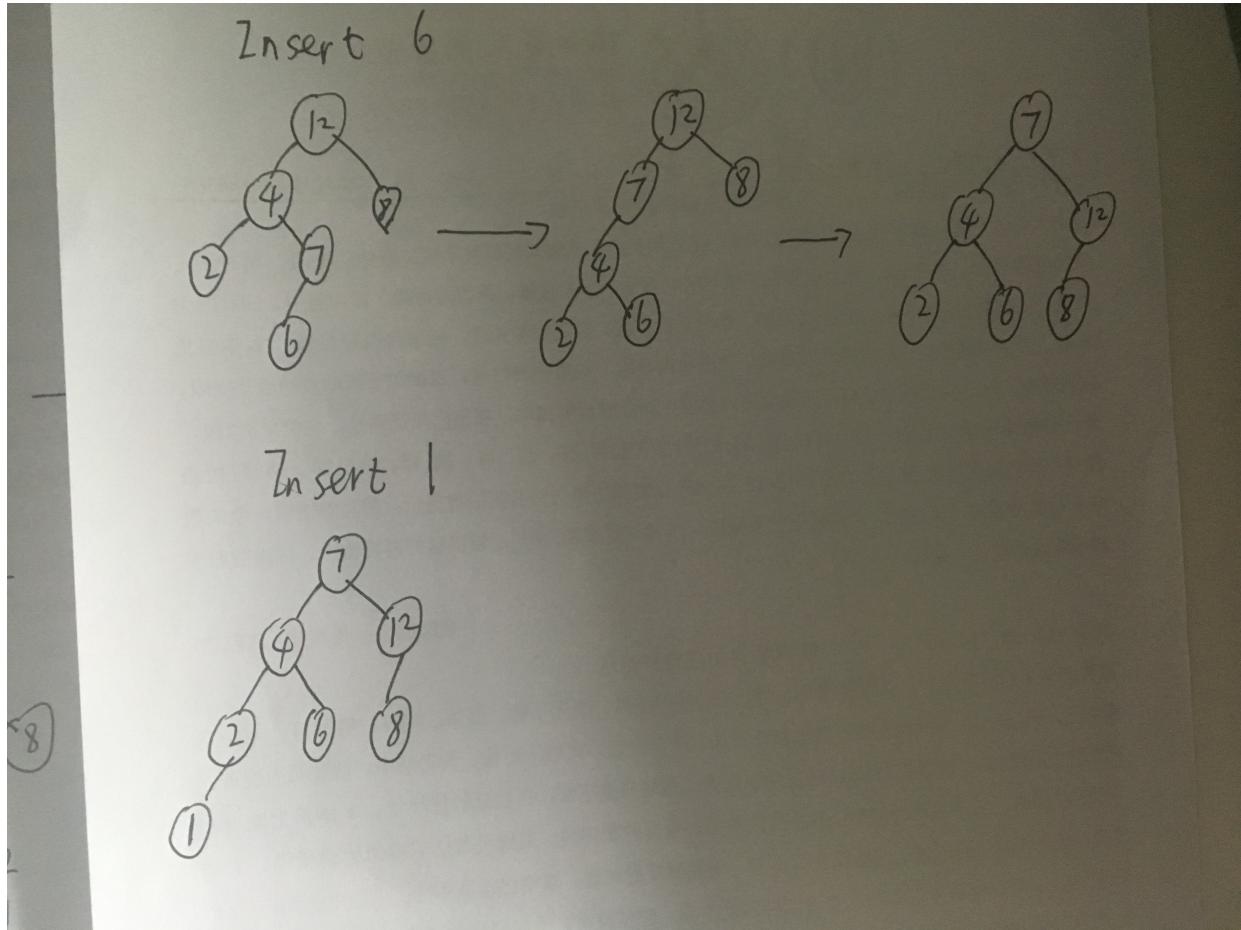
```

        return 1 + countInternal(root.firstChild) +
countInternal(root.nextSibling);
    }
}

```

## Q6





## Q7

```
// Q7
boolean testMinHeap(int A[]) {

    int N = A.length - 1;

    for(int i=1; i<=N; i++)
    {

        if(2*i <= N)
        {
            if(A[i] > A[2*i])
            {
                return false;
            }
        }

        if(2*i + 1 <= N)
        {
            if(A[i] > A[2*i + 1])
            {
                return false;
            }
        }
    }
}
```

```
    }

    return true;
}
```

## Q8

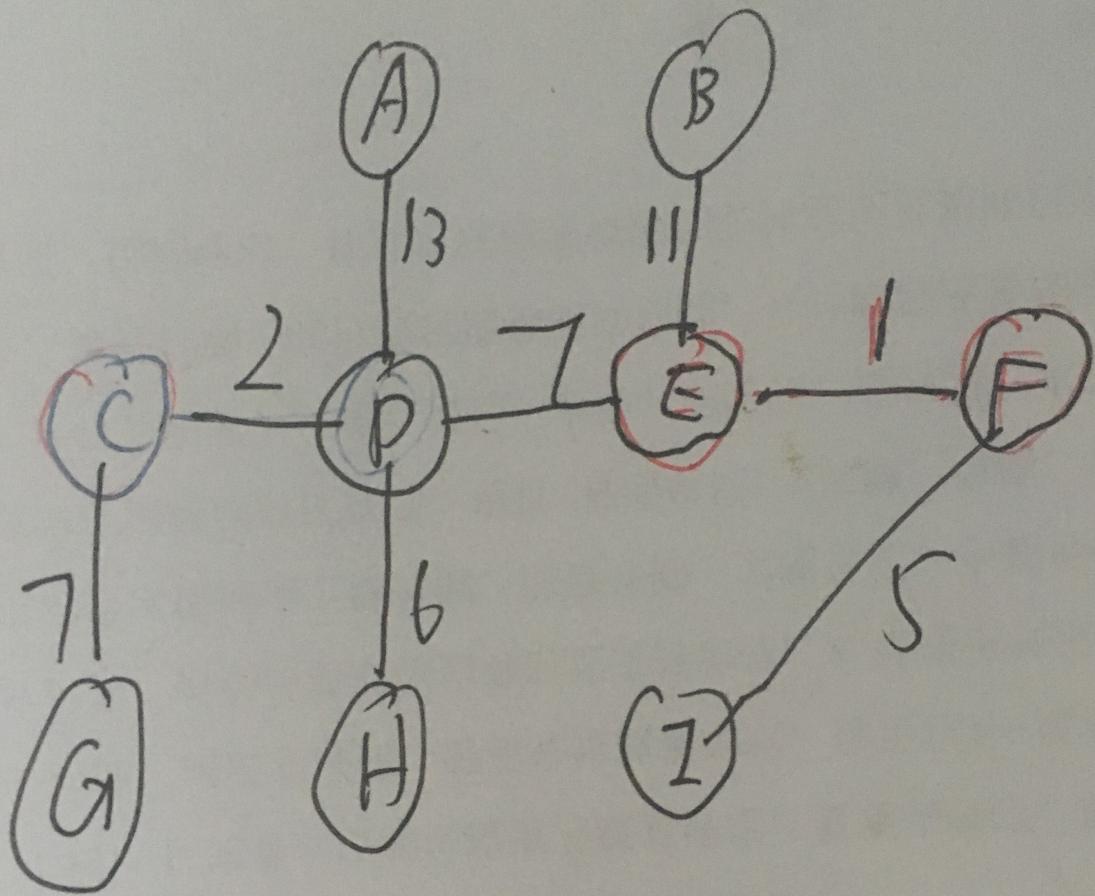
---

### 8.a

No it is not strongly connected. Because there is no path from node F to node C.

### 8.b

edges examined	accept or rejected
(E F) 1	accept
(C D) 2	accept
(F I) 5	accept
(D H) 6	accept
(C G) 7	accept
(D E) 7	accept
(E I) 8	rejected
(G H) 10	rejected
(B E) 11	accept
(H I) 12	rejected
(A D) 13	accept



Q9

Step	Vertex Chosen	A	B	C	D	E	F	G
0	--	Infinity	Infinity	0	Infinity	Infinity	Infinity	Infinity
1	C	Infinity	Infinity	0	7	7	Infinity	Infinity
2	D	9	Infinity	0	7	7	13	Infinity
3	E	9	Infinity	0	7	7	8	Infinity
4	F	9	Infinity	0	7	7	8	Infinity
5	A	9	14	0	7	7	8	Infinity
6	B	9	14	0	7	7	8	15

## Q10

---

If we can solve in  $O(n^4)$  time, then it means it is solved in polynomial time, and this problem is in  $P$ . Because the bounded travelling salesperson problem is a NP-complete problem. Now a NP-complete problem is in  $P$ , because every problem in  $NP$  can be reduced to NP-complete problem. Thus, every problem in  $NP$  can be now solved in polynomial time. Thus, we have  $P = NP$ .

## Q11

---

It is a directed graph. It is not weakly connected. It is not strongly-connected. It has 5 nodes and 2 edges.