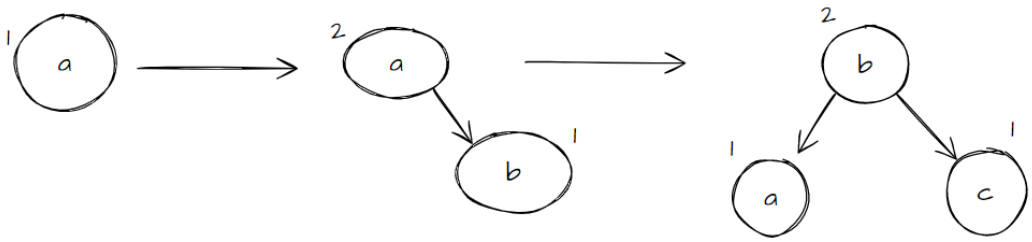
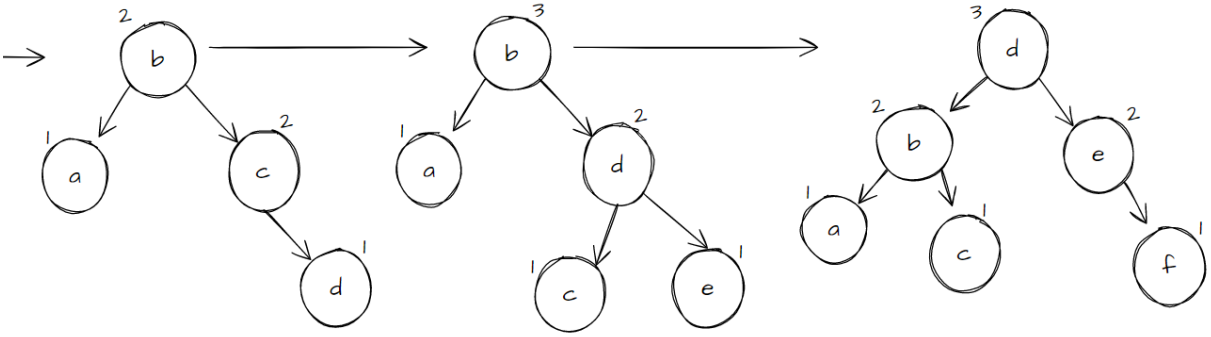
2383 Assignment

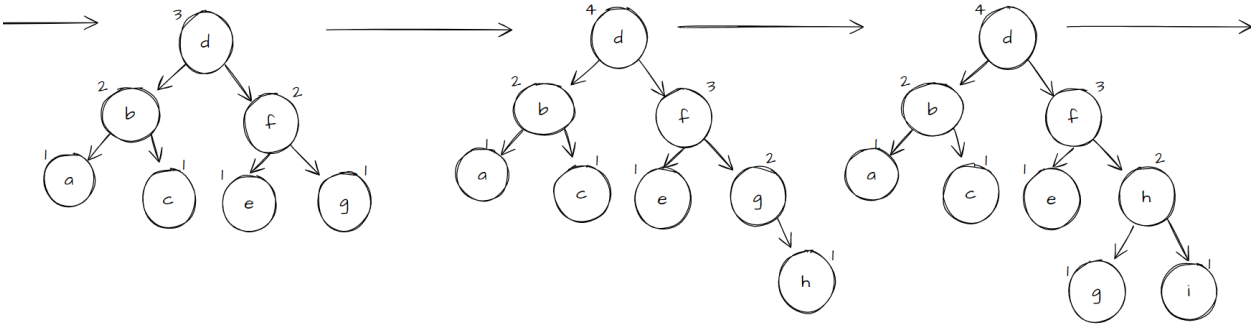
1- a,b,c,d,e,f,g,h,i,j,k,l



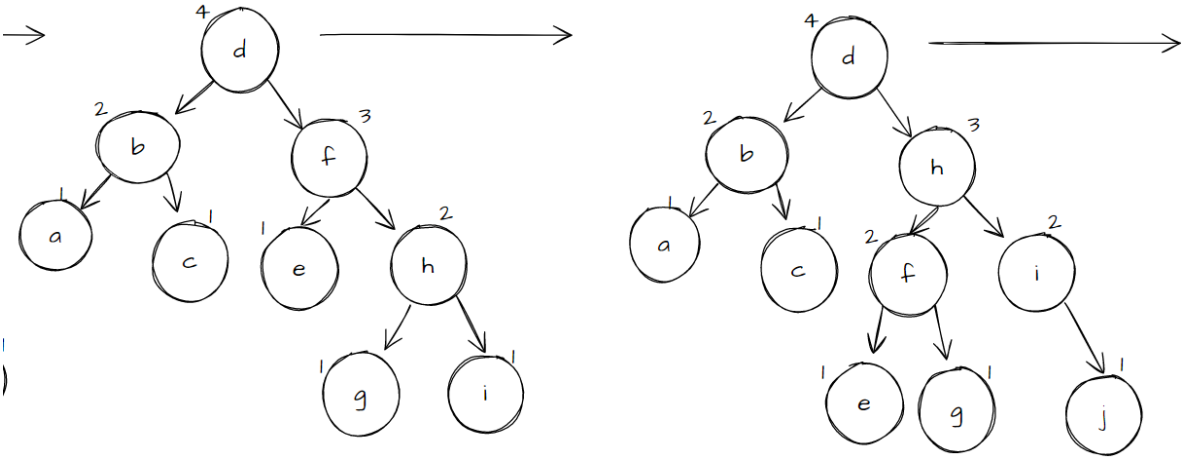
Then



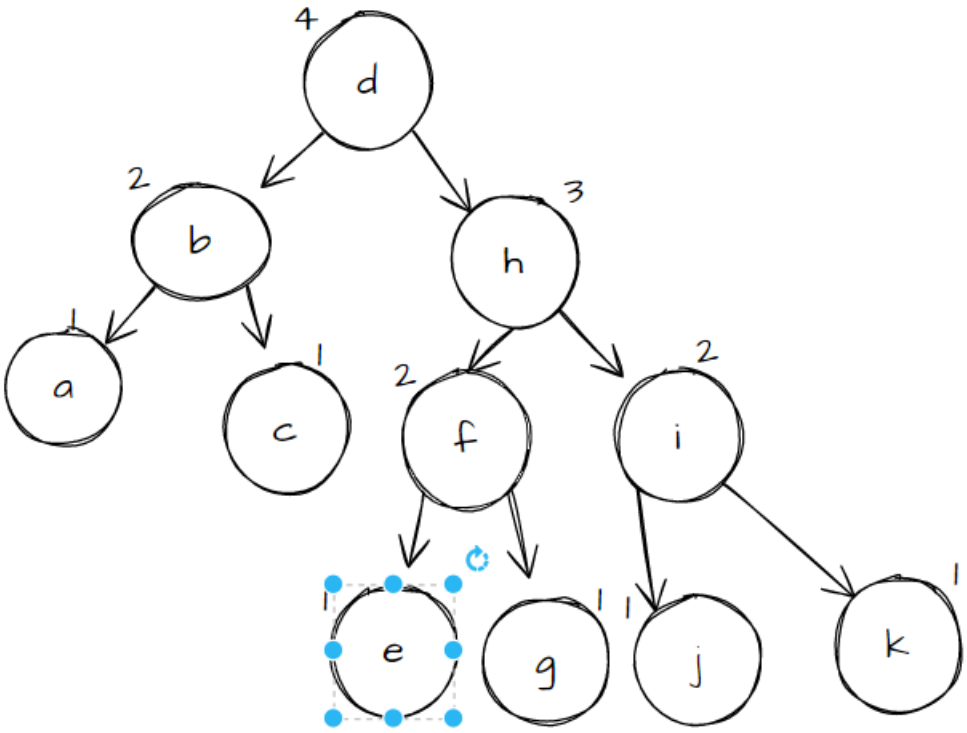
Then



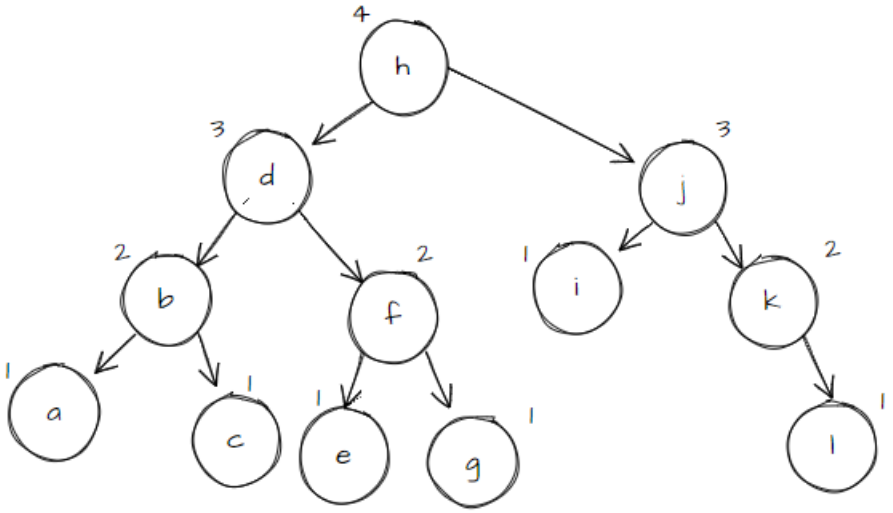
Then


4
d
2
b
3
f
1
a
1
c
1
e
2
h
1
g
1
i
4
d
2
b
3
h
1
a
1
c
2
f
2
i
1
e
1
g
1
j

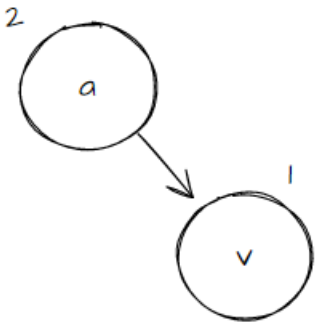
Then


4
d
2
b
3
h
1
a
1
c
2
f
2
i
1
e
1
g
1
j
1
k

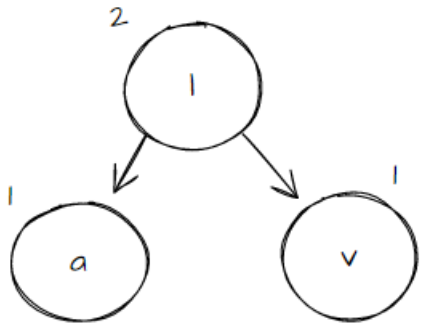
Finally

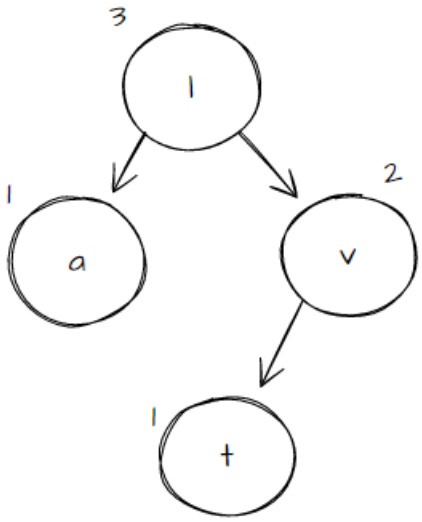


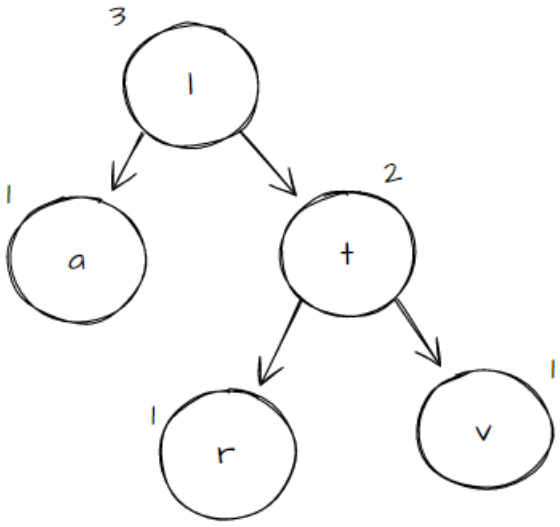
1- a,v,l,t,r,e,i,s,f,u,n

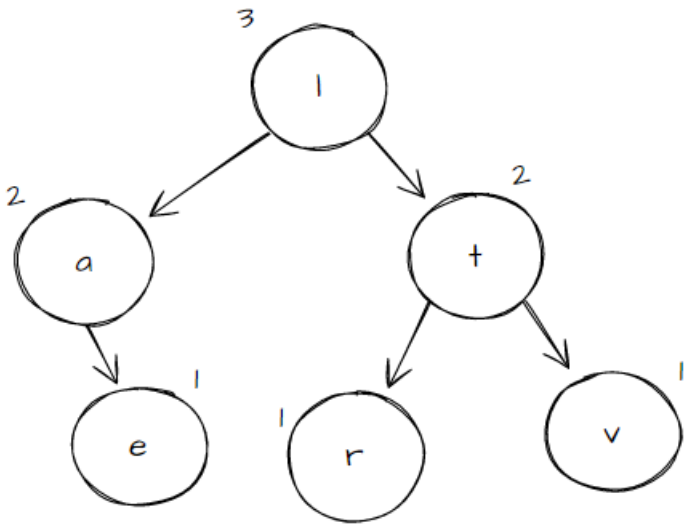


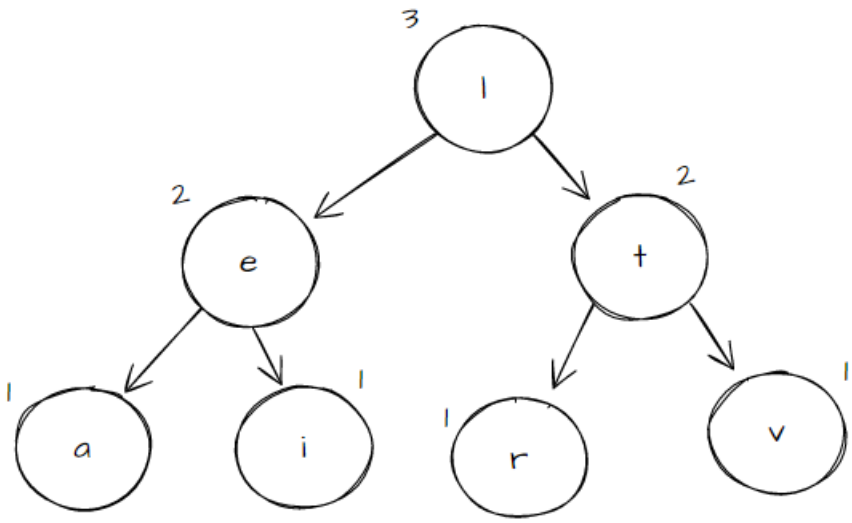
Then

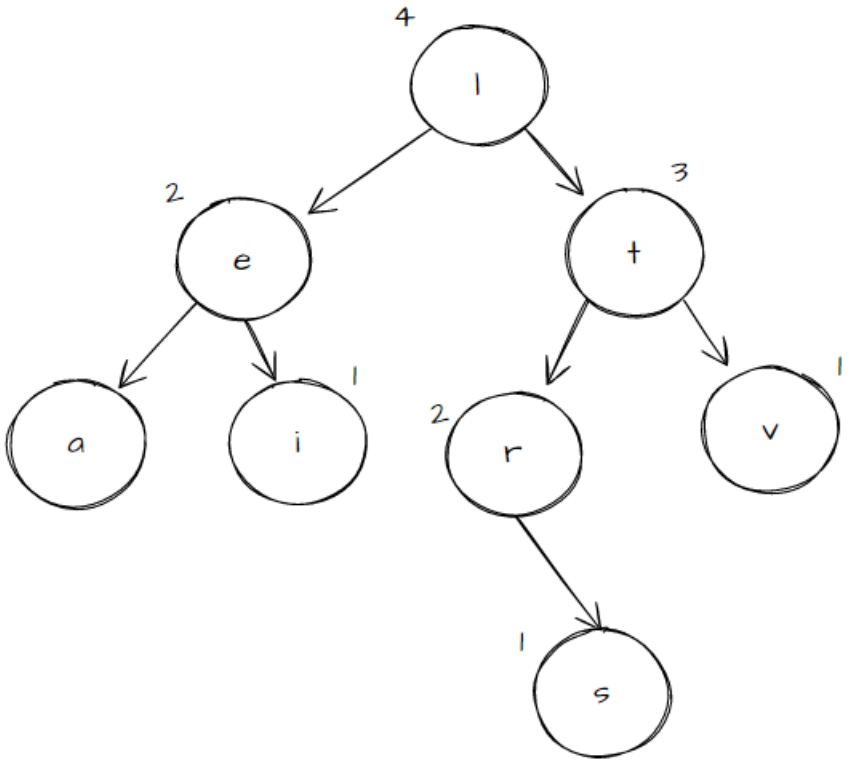

2
l
1
a
1
v

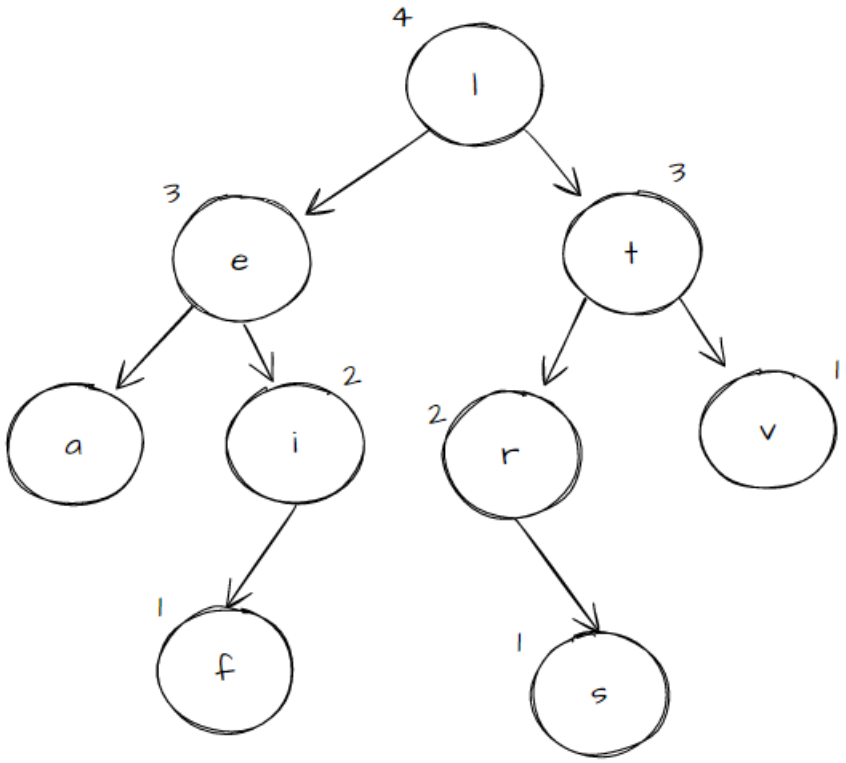

3
l
1
a
2
v
1
t

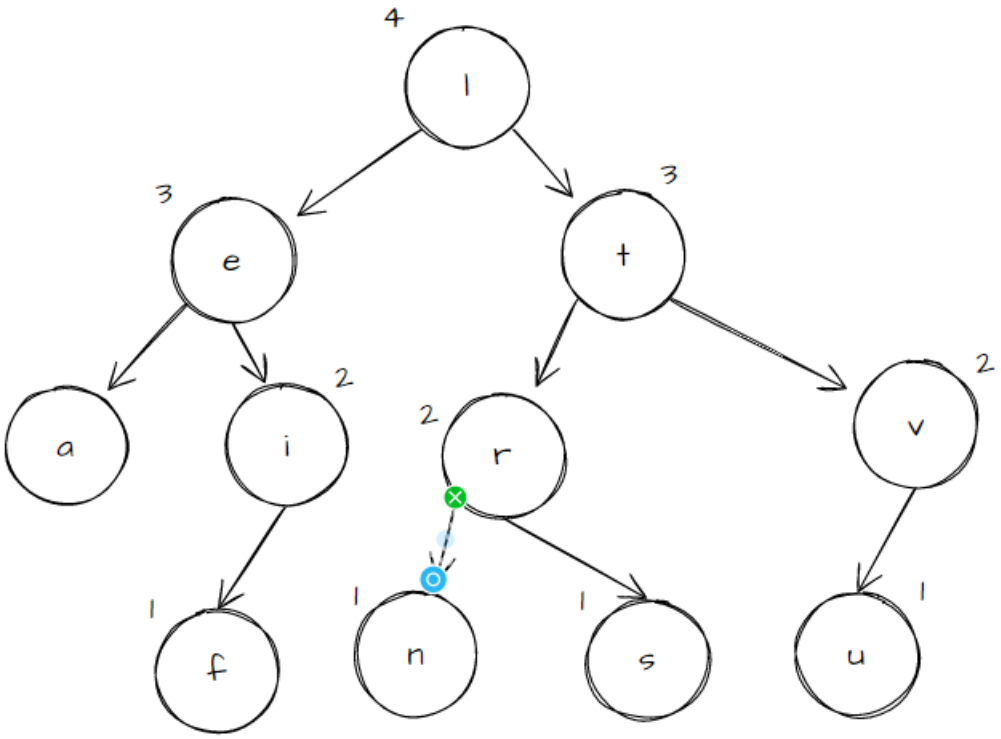

3
l
1
a
2
t
1
r
1
v

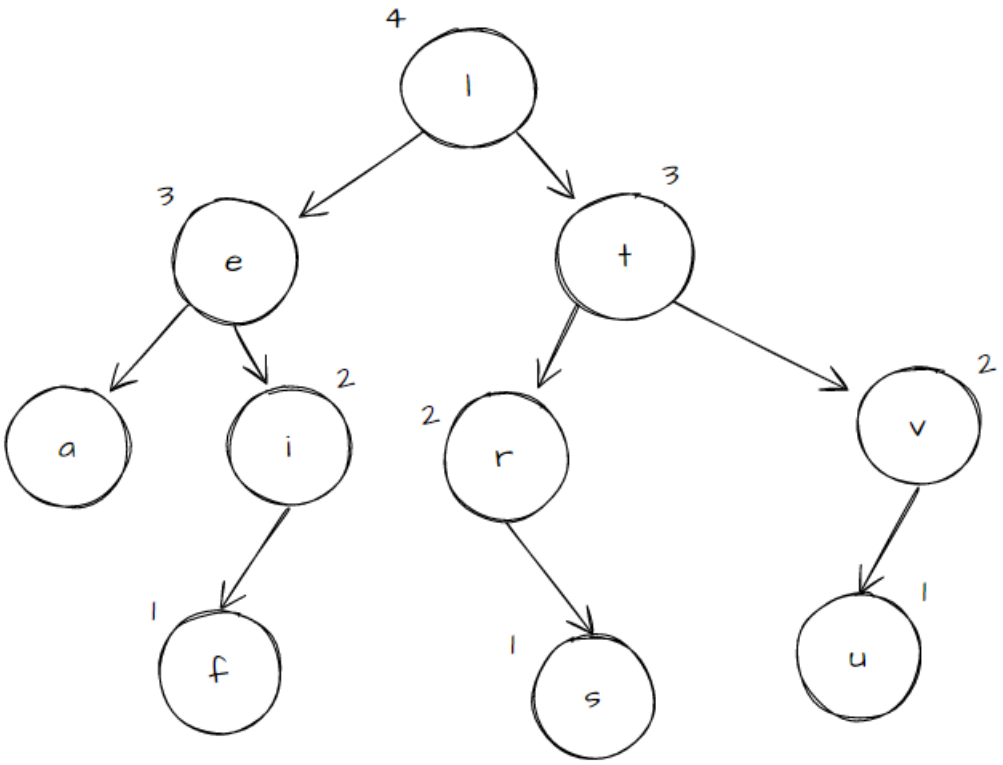

3
l
2
a
2
t
1
e
1
r
1
v

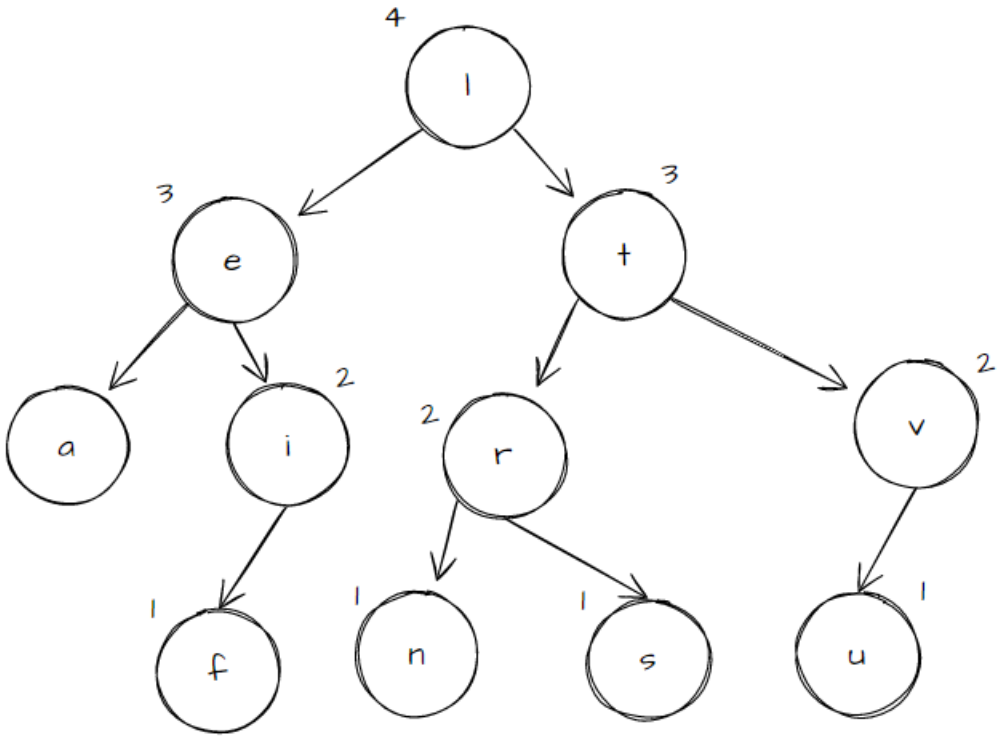

3
l
2
e
2
t
1
a
1
i
1
r
1
v


4
l
2
e
3
t
a
1
i
2
r
1
v
1
s


4
l
3
e
3
t
a
2
i
2
r
1
v
1
f
1
s


4
l
3
e
3
t
a
2
i
2
r
2
v
1
f
1
n
1
s
1
u


4
l
3
e
3
t
a
2
i
2
r
2
v
1
f
1
s
1
u


4
l
3
e
3
t
a
2
i
2
r
2
v
1
f
1
n
1
s
1
u

2-

```
Algorithm: findAllInRange (n, k1, k2)
Input: Node n, int k1, int k2
findAllInRange(n,k1,k2){
        count ← 0;
        if(n = null){
                return;
        }
        if (k1< n.data)
        {
                findAllInRange(n.left, k1, k2);
        }
        if (k1 <= n.data AND k2 >= n.data) {
                print(n.data + " ");
                count++;
        }
        if (k2 > n.data) {
                findAllInRange(n.right, k1, k2);
        }
}
findAllInRange(int k1, int k2)
{
        findAllInRange(root, k1, k2);
}
```

Time complexity is O(n).

3-

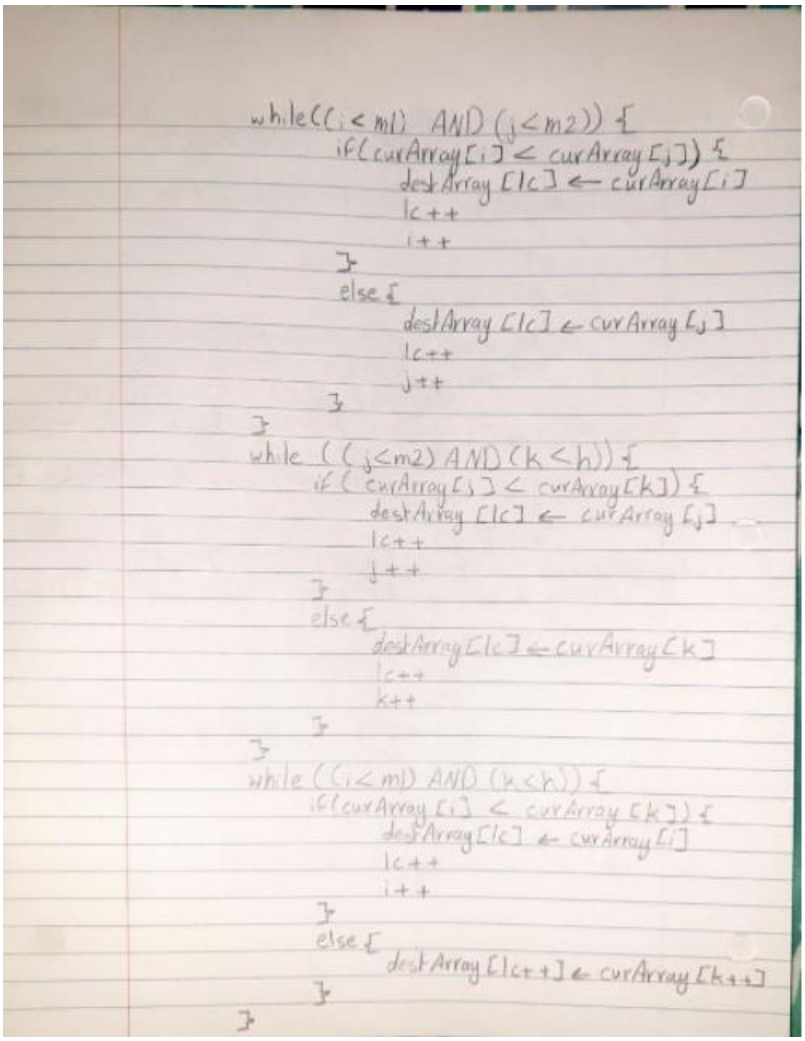
k	(2i+5)mod11	7 – (i mod 7)
12	7	
44	5	
13	9	
88	5	3->(5+3) mod 11 = 8
23	7	5-> 7+5mod11 =1
94	6	
11	5	3->8 Then (5+2*3) mod 11 = 0
39	6	3->9->1->4
20	1	1->2
16	4	5->9->3
5	4	2->6->8->10

Hash Table	
H[0]	11
H[1]	23
H[2]	20
H[3]	16
H[4]	39
H[5]	44
H[6]	94
H[7]	12
H[8]	88
H[9]	13
H[10]	5

4-

4) Algorithm merge(curArray[], l, m1, m2, h, destArray[])
input: int curArray[], l, m1, m2, h, destArray[].

```
merge(curArray[], l, m1, m2, h, destArray[]) {
    i ← l
    j ← m1
    k ← m2
    lc ← l
    while ((i < m1) AND (j < m2) AND (k < h)) {
        if (curArray[i] < curArray[j]) {
            if (curArray[i] >= curArray[k])
                destArray[lc++] ← curArray[k++]
            else
                destArray[lc++] ← curArray[i++]
        }
        else {
            if (curArray[j] < curArray[k]) {
                destArray[lc] ← curArray[j]
                lc++
                j++
            }
            else {
                destArray[lc] ← curArray[k]
                lc++
                k++
            }
        }
    }
```



```
while((i < m1) AND (j < m2)) {
    if(curArray[i] < curArray[j]) {
        destArray[lc] ← curArray[i]
        lc++
        i++
    }
    else {
        destArray[lc] ← curArray[j]
        lc++
        j++
    }
}
while ((j < m2) AND (k < h)) {
    if (curArray[j] < curArray[k]) {
        destArray[lc] ← curArray[j]
        lc++
        j++
    }
    else {
        destArray[lc] ← curArray[k]
        lc++
        k++
    }
}
while ((i < m1) AND (k < h)) {
    if(curArray[i] < curArray[k]) {
        destArray[lc] ← curArray[i]
        lc++
        i++
    }
    else {
        destArray[lc++] ← curArray[k++]
    }
}
```

```
while (i<m1) {
    destArray[lc] ← curArray[i]
    lc++
    i++
}
while (j<m2) {
    destArray[lc] ← curArray[j]
    lc++
    j++
}
while (k<h) {
    destArray[lc] ← curArray[k]
    lc++
    k++
}
}
```

Algorithm: MS3WR (curArray[], l, h, destArray[])
input: int curArray[], l, h, destArray[]

```
MS3WR (curArray[], l, h, destArray[]) {
    if (h-l<2) THEN
        return
    m1 ← l + ((h-l)/3)
    m2 ← l + 2*((h-l)/3)+1
    MS3WR (destArray, l, m1, curArray)
    MS3WR (destArray, m1, m2, curArray)
    MS3WR (destArray, m2, h, curArray)
    merge (destArray, l, m1, m2, h, curArray)
}
```

Algorithm: MS3Way (curArray[], n)
input: int curArray[], n

```
MS3Way (curArray[], n) {
    if (n=0)
        return
    for (i←0; i<n; i++)
        fArray[i] ← curArray[i]
    MS3WR (fArray, 0, n, curArray)

    for (i←0; i<n; i++)
        curArray[i] ← fArray[i]
}
```

$$T(n) = 3T\left(\frac{n}{3}\right) + O(n)$$

$$= 3\left(3T(n/3) + n/3\right) + n$$

$$= 3^2 T(n/3^2) + 2n$$

$$= 3^2\left(3T(n/3^2) + n/9\right) + 2n$$

$$\vdots$$

$$3^k T(n/3^k) + kn$$

$$\frac{n}{3^k} = 1 \rightarrow n = 3k \rightarrow \log n = k \log 3$$

$T(n)$ is $O(n \log n)$

5-

```
Algorithm: exchange(array, ind1, ind2)
Input: int[] array, int ind1, ind2
exchange(array, ind1, ind2){
        temp ← array[ind1]
        array[ind1] ← array[ind2]
        array[ind2] ← temp;
}

Algorithm: gRI(min, max)
Input: int min, int max
gRI(min, max){
        r ← new Random()
        lim ← (max-min)+1
        rni ← r.nextInt(lim)+min
        return rni
}

Algorithm: fKL(array, k)
Input: int[] array, int k
fKL(array, k){
        if array.length = 1 then
                return array[0]
        return sR(array, 0, array.length-1, k)
}
```

```
Algorithm: sR(array, start, end, k)
Input: int[] array, inst start, int end, int k
sR(array, start, end, k){
        ind ← rP(array, start, end)
        if (ind = (k-1))
                return array[ind]
        else if(ind  >= k)
                return sR(array, start, ind-1, k)
        else
                return sR(array, ind+1, end, k)
}
Algorithm: rP(array, start, end)
Input: int[] array, int start, int end
rP(array, start, end){
        rI ← gRi (start, end)
        exchange(array, rI)
        piv ← array[end]
        p ← start-1
        for (i ← start to end){
                if (array[i]>=piv){
                        p++
                        exchange(array, i, p)
                }
        }
        exchange(array, p+1, end)
        return p++
}
```

Time complexity is O(n)