Seperate chaining

1)

0	7
1	
2	
3	3
4	
5	12 → 5
6	27

linear probing

2)

0	5
1	7
2	
3	3
4	
5	12
6	27

double hashing

3)

0	3
1	
2	7
3	5
4	
5	12
6	27

Scanned with CamScanner

Find time complexity using substitution

4) $F(1) = C_0$

$F(n) = F(n-1) + c_1 n + c_2,\ n > 0$ ①

$F(n-1) = F(n-2) + c_1(n-1) + c_2$

substitute into 1

$F(n) = [F(n-2) + c_1(n-1) + c_2] + c_1 n + c_2$ ②

sob into ② $F(n-2) = F(n-3) + c_1(n-2) + c_2$

$F(n) = [F(n-3) + c_1(n-2) + c_2] + c_1(n-1) + c_2 + c_1 n + c_2$

$= F(n-3) + c_1(n-2) + c_1(n-1) + c_1 n + 3(c_2)$

General Formula

$F(n) = F(n-k) + c_1(n-(k-1)) + c_1(n-(k-2)) \ldots c_1 n + kc_2$

$F(1) = C_0$

$n-k = 1$

$k = n-1$

sub n-1 for k in general formula

$F(n) = f(n-(n-1)) + c_1(n-(n-1-1)) \ldots c_1 n + (n-1)c_2$

$F(n) = f(1) + c_1(2) + c_1(3) \ldots + c_1(n) + (n-1)c_2$

$F(n) = C_0 + C_1(2+3+\ldots n) + (n-1)c_2$

*sub c_0 for $f(1)$

$F(n) = c_0 + c_1\left(\sum_{i=1}^{n} i - 1\right) + (n-1)c_2$

since the sum is from $2 \to n$ we can take the sum from $1 \to n$ minus 1

$F(n) = c_0 + c_1\left(\frac{n(n+1)}{2} - 1\right) + (n-1)c_2$

$= c_0 + c_1\left(\underbrace{\frac{n^2+n}{2}}_{} - 1\right) + c_2 n - c_2$

n^2 is the upper bound of this function

$\therefore F(n) \in O(n^2)$

Scanned with CamScanner

5)

```
1. totalLeaves(r, level):
2.     if r.isLeaf():
3.         return level
4.     count = 0
5.     for each child c of r do:
6.         count += totalLeaves(c, level+1)
7.     return count
```

line 2: we check if the current node is a leaf,
line 3: if it is we return the current level.
line 4: a variable to hold the sum of the levels of all the children variables
line 5: looping thorogh every child
line 6: adding the levels of all the leaves under the current node to the sum
line 7: returning the sum

5b) Time complexity:

let the if statement (line 2,3) be constant c_1
let everything inside the loop (line 5,6) be constant c_2
let everything else (line 4, 7) be constant c_3

if leaf node: c_1
otherwise: $c_2(\text{degree}(r)) + c_3$

$$f(n) = \#\text{ operations on leaves} + \#\text{ operations internal nod}$$
$$= \sum_{\text{leaves}} c_1 + \sum_{\text{internal node}} (c_3 + c_2(\text{degree}(n)))$$
$$= \text{leaves} \times c_1 + \text{internal} \times c_3 + \text{internal} \times \text{degree}(r) \times c_2$$
$$= \underbrace{\text{leaves} \times c_1 + \text{internal} \times c_3 + n \times c_2}_{n}$$

n is largest scale variable

$\therefore f(n) \in O(n)$

6) tracing algo(A, B, n)

	i	j
1	0	0
2	n	$n+1$
end		

- i and j are both 0 at the start

since $B[0] \leftarrow A[0]$, $A[0] = B[0]$ is true $\therefore$ $i \leftarrow n-0$ $\therefore$ $i \leftarrow n$

$j \leftarrow i+1$ $\therefore$ $j \leftarrow n+1$

the function terminates since $j \geq n$

time complexity: let everything outside the loop be c_1

let everything inside the loop be c_2

$F(n) = c_1 + c_2$ ← only one instance of c_2 because loop only runs 1 time

function is only constants, so it runs in constant time.

$\therefore F(n) \in O(1)$