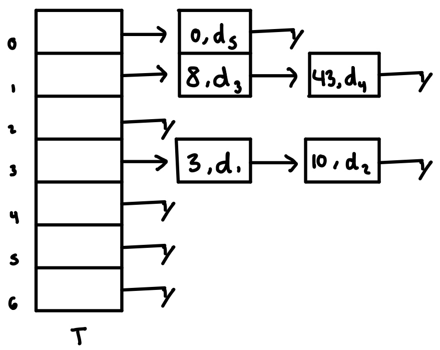
Mylan Nguyen - 251155416
Assignment 3

1. Size of the hash table as $N = 7$
Hash Function: $h(k) = k \bmod 7$



$h(3) = 3 \bmod 7$
$= 3$
$h(10) = 10 \bmod 7$
$= 3$
$h(8) = 8 \bmod 7$
$= 1$
$h(43) = 43 \bmod 7$
$= 1$
$h(0) = 0 \bmod 7$
$= 0$

$(3, d_1)$
$(10, d_2)$
$(8, d_3)$
$(43, d_4)$
$(0, d_5)$

2. Size of the hash table as $N = 7$
Hash Function: $h(k) = k \bmod 7$

	T
0	$(0, d_5)$
1	$(8, d_3)$
2	$(43, d_4)$
3	$(3, d_1)$
4	$(10, d_2)$
5	
6	

$h(3) = 3 \bmod 7$
$= 3$
$h(10) = 10 \bmod 7$
$= 3$ → go to next empty: $(3+1) \bmod 7$ $= 4$
$h(8) = 8 \bmod 7$
$= 1$
$h(43) = 43 \bmod 7$
$= 1$ → go to next empty: $(1+1) \bmod 7$ $= 2$
$h(0) = 0 \bmod 7$
$= 0$

$(3, d_1)$
$(10, d_2)$
$(8, d_3)$
$(43, d_4)$
$(0, d_5)$

3. Size of the hash table as $N = 7$
Hash Function: $h(k) = k \bmod 7$
$h'(x) = 5 - (x \bmod 5)$

	T
0	$(43, d_4)$
1	$(10, d_2)$
2	
3	$(3, d_1)$
4	
5	$(8, d_3)$
6	$(0, d_5)$

$h(3) = 3 \bmod 7$
$= 3$
$h(10) = 10 \bmod 7$
$= 3$ → $h'(10) = 5 - (10 \bmod 5)$
$= 5 - 0$
$= 5$
$h(8) = 8 \bmod 7$
$= 1$ → $h'(8) = 5 - (8 \bmod 5)$
$= 5 - 3$
$= 2$
$h(43) = 43 \bmod 7$
$= 1$ → $h'(43) = 5 - (43 \bmod 5)$
$= 5 - 3$
$= 2$

$(3, d_1)$
$(10, d_2)$
$(8, d_3)$
$(43, d_4)$
$(0, d_5)$

$$h(0) = 0 \bmod 7$$
$$= 0 \longrightarrow h'(0) = 5 - (0 \bmod 5)$$
$$= 5 - 0$$
$$= 5$$

4. $f(0) = C_0$

$f(n) = f(n-2) + C_1 n + C_2$, for $n \geq 0$

$$f(n-2) = f((n-2)-2) + C_1(n-2) + C_2$$
$$= f(n-4) + C_1(n-2) + C_2$$

$$f((n-2)-2) = f(((n-2)-2)-2) + C_1((n-2)-2) + C_2$$
$$= f(n-6) + C_1(n-4) + C_2$$

$$f(((n-2)-2)-2) = f((((n-2)-2)-2)-2) + C_1(((n-2)-2)-2) + C_2$$
$$= f(n-8) + C_1(n-6) + C_2$$

$\vdots$

$$f(n-(2\cdot i)) = \underbrace{f(n-2(i+1))}_{=0} + C_1(n-(2\cdot i)) + C_2$$

Let $(n-2(i+1)) = 0$, so by performing repeated substitutions we get:

$$f(n) = C_0 + C_1(n-(2\cdot i)) + C_2$$

$$f(n) = C_0 + C_1 \sum_{j=0}^{i} (n-2i) + C_2$$

$$f(n) = \cancel{C_0} + \cancel{C_1}\, \frac{n-2i((n-2i)+\cancel{1})}{\cancel{2}} + \cancel{C_2}$$

$$= C_0 + C_1(n^2) + C_2$$

$\therefore O(n^2)$ is the time complexity

$$(n-2(i+1)) = 0$$
$$n = 2(i+1)$$
$$\frac{n}{2} = i+1$$
$$\frac{n}{2} - 1 = i$$

5.ii) The worst case time complexity is when all the nodes of the tree have an odd degree (no even degree node) so the function must traverse through all the nodes of the tree.

By first analyzing the algorithm, ignoring recursive calls; C_1 operations in base case, $C_3 + C_2 \times$ degree (r) in recursive case.

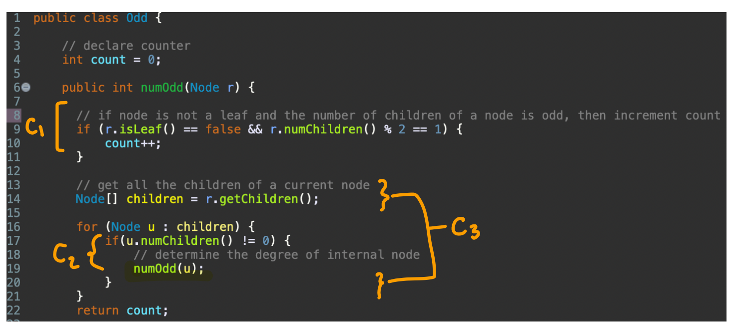
The total number of calls in the recursive case is one call per odd node.

$$\sum_{\text{leaves}} C_1 + \sum_{\substack{\text{internal} \\ \text{odd nodes } (u)}} (C_3 + C_2 \times \text{degree}(u))$$

$$f(n) = \cancel{C}_1 + \cancel{C}_3 + \cancel{C}_2(n - \cancel{1})$$

$\llcorner$ # of edges

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



```
public class Odd {

    // declare counter
    int count = 0;

    public int numOdd(Node r) {

        // if node is not a leaf and the number of children of a node is odd, then increment count
        if (r.isLeaf() == false && r.numChildren() % 2 == 1) {
            count++;
        }

        // get all the children of a current node
        Node[] children = r.getChildren();

        for (Node u : children) {
            if(u.numChildren() != 0) {
                // determine the degree of internal node
                numOdd(u);
            }
        }
        return count;
```

6.

Algorithm algo(A, B, n)
In: Arrays A and B of size $\frac{n(n+1)}{2}$

$$\left.\begin{array}{l} i \leftarrow 0 \\ j \leftarrow 0 \end{array}\right\} c_1$$

$$\left.\begin{array}{l} \textbf{while } j < \frac{n(n+1)}{2} \textbf{ do } \{ \\ \quad B[i] \leftarrow A[j] \\ \quad i \leftarrow i+1 \\ \quad j \leftarrow j+i \end{array}\right\} c_2$$
}

$\therefore$ Time complexity is $O(n)$

Note that $\sum_{i=1}^{n} i = j$

So, the last term of the sequence must be less than $\frac{n(n+1)}{2}$. Since j is increasing at the same amount of as the function $\frac{n(n+1)}{2}$

Ex: if $n=3$,

$$\left.\begin{array}{l} \text{Iteration } 1: \; i=1,\ j=1 \\ \text{Iteration } 2: \; i=2,\ j=3 \\ \text{Iteration } 3: \; i=3,\ j=6 \end{array}\right\} \begin{array}{l} n=3 \\ \text{and there are} \\ 3 \text{ iterations} \end{array}$$

So the loop iterates n times

So $f(n) = \cancel{C}_1 + n\cancel{C}_2$

$\therefore$ the time complexity is $O(n)$