



4. When n is even:

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

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

$$\vdots$$

$$f(2) - f(0) = C_1 \cdot 2 + C_2$$

Add up all functions above,

we can have:

$$f(n) - f(0) = C_1 [n + (n-2) + \dots + 2] + \frac{n}{2} \cdot C_2$$

$$f(n) = \frac{(n+2) \cdot n}{4} \cdot C_1 + \frac{n}{2} \cdot C_2 + C_0$$

$$= \frac{C_1}{4} n^2 + \frac{(C_1 + C_2)}{2} n + C_0$$

So the complexity of $f(n)$ is $O(n^2)$.

$f(n)$

```
public class Odd {
    public int numOdd(Node r) {
        int count = 0;
        if(r.isLeaf()) {
            return 0;
        }
        else {
            if((r.numChildren()%2==1)) {
                count+=1;
            }
            Node[] children = r.getChildren();
            for(Node u: children) {
                count+=numOdd(u);
            }
        }
        return count;
    }
}
```

Worst case: all its nodes are looked through.

$$f(n) = (n+C_1) \cdot (n+C_1) = n^2 + 2C_1 n + C_1^2$$

The complexity is $O(n^2)$.

6. Since i is incremented to $0, 1, 2, \dots, n$, j would be incremented by $1, 2, 3, \dots, n$. $\sum_{i=1}^n i = \frac{n(n+1)}{2}$. So there're n times the while loop goes. For each while loop, there're three basic operation. So $f(n) = 2 + 3n$, complexity is $O(n)$.