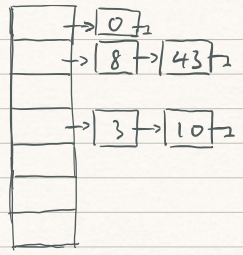
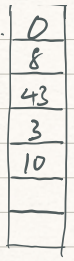
1. 

2. 

3. 

4. When n is even:

$f(n) - f(n-2) = c_1 n + c_2$

$f(n-2) - f(n-4) = c_1 (n-2) + c_2$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

$f(2) - f(0) = c_1 \times 2 + c_2$.

Add up all functions above, we can have:

$f(n) - f(0) = c_1 [n + (n-2) + \cdots + 2] + \frac{n}{2} \cdot c_2$

$f(n) = \frac{(n+2)\cdot n}{4} \cdot c_1 + \frac{n}{2} \cdot c_2 + c_0$

$= \frac{c_1}{4} n^2 + \frac{(c_1+c_2)}{2} n + c_0$.

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

5. (i)

```
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;
  }
}
```

} n

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 $0, 1, 2 \cdots, j$ would be incremented by $1, 2, 3 \cdots, \sum_{1}^{n} n = \frac{(n+1)n}{2}$. 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 $O(n)$.