


Here is the solution to the sample questions on time complexity. As an alternative to the computations here, you can also use the master theorem ([https://en.wikipedia.org/wiki/Master\\_theorem\\_\(analysis\\_of\\_algorithms\)](https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms))). I personally, don't like that approach because it gives little insight as to why the time complexity is what it is. But you may choose to use it over this kind of computation.


Mystery1: Final result :  $O(n^3)$

```
void mystery1(int n){
    for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
            cout << i << " " << j << endl;
    if (n>0)
        mystery1(n-1);
}
```

<p>The graph of complexity:</p> 	<p>Computations:</p> <p>In level i:          only one child          size of the child is <math>n-i</math>          the extra time we spend is <math>O((n-i)^2)</math></p> <p>number of levels = <math>n</math></p> <p>The sum of extra time over all the levels :  <math>n(n+1)(2n+1)/6 = O(n^3)</math></p>
--	--


Mystery2: Final result :  $O(n)$

```
void mystery2(int n){  
    for (int i=0; i<n; i++)  
        cout << i << endl;  
    if (n>1)  
        mystery2(n/2);  
}
```

<p>The graph of complexity:</p>  <pre>graph TD; n((n)) --- n2((n/2)); n2 --- n4((n/4)); n4 --- dots[⋮]; dots --- 1((1))</pre>	<p>Computations:</p> <p>In level i: only one child size of the child is <math>n/(2^i)</math> the extra time we spend is <math>C*n/(2^i) = O(n/(2^i))</math></p> <p>number of levels: <math>\lg(n) + 1</math> (assuming <math>n=2^k</math>)</p> <p>The sum of extra time over all the levels : <math>C* n* (1/1+1/2+...1/(2^i)+....1/2^{\lg(n)}) &lt;</math> <math>C* n* (1/1+1/2+....) \leq n* 2 = O(n)</math></p>
---	--

Mystery3: Final result :  $O(\lg(n))$

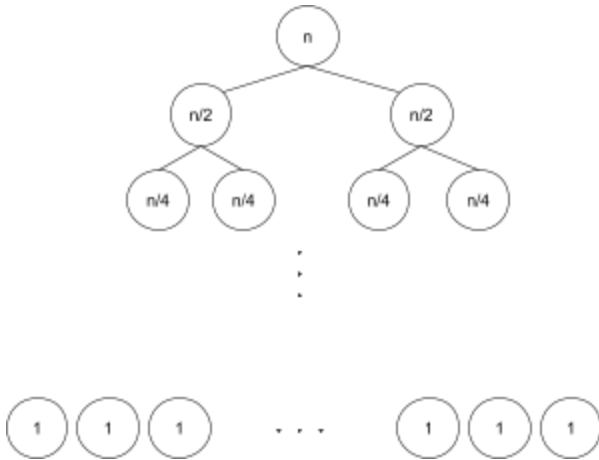
```
void mystery3(int n){  
    cout << n << endl;  
    if (n>1)  
        mystery3(n/2);  
}
```

<p>The graph of complexity:</p>  <pre>graph TD; n((n)) --- n2((n/2)); n2 --- n4((n/4)); n4 --- dots[⋮]; dots --- 1((1))</pre>	<p>Computations:</p> <p>In level i: only one child size of the child is <math>n/(2^i)</math> the extra time we spend is <math>C = O(1)</math></p> <p>number of levels: <math>\lg(n) + 1</math> (assuming <math>n=2^k</math>)</p> <p>The sum of extra time over all the levels : <math>C + C + C + \dots + C</math> (number of times: <math>\lg(n)+1</math>) <math>= C \lg(n) + C = O(\lg(n))</math></p>
---	---

Mystery4: Final result :  $O(n)$

```
void mystery4(int n){  
    cout << n << endl;  
    if (n>1){  
        mystery4(n/2);  
        mystery4(n/2);  
    }  
}
```

The graph of complexity:



Computations:

In level  $i$ :

$2^i$  children

size of the child is  $n/(2^i)$

the extra time we spend for each child  $C =$

$O(1)$

the total extra time we spend in level  $i$  is

$C \cdot 2^i$

number of levels =  $\lg(n)+1$  (assuming  $n=2^k$ )

The sum of extra time over all the levels :

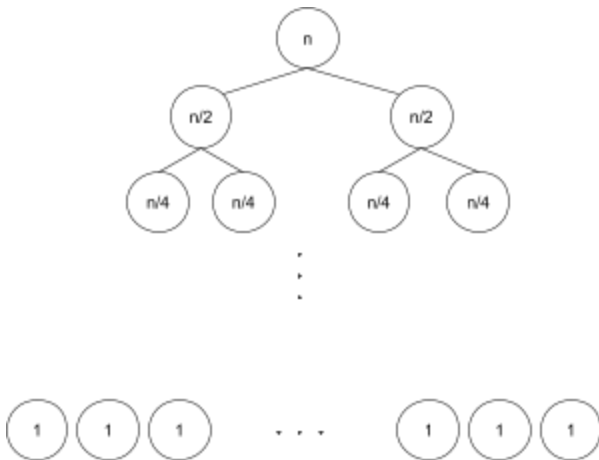
$2^0 + 2^1 + 2^2 + \dots + 2^i + \dots + 2^{\lg(n)+1} =$

$2^{\lg(n)+2} - 1 = 4n - 1 = O(n)$

Mystery5: Final result :  $O(n \lg(n))$

```
void mystery5(int n){  
    for (int i=0; i<n; i++)  
        cout << i << endl;  
    if (n>1){  
        mystery5(n/2);  
        mystery5(n/2);  
    }  
}
```

The graph of complexity:



Computations:

In level  $i$ :

$2^i$  children

size of the child is  $n/(2^i)$

the extra time we spend for each child

$C \cdot n / (2^i) = O(n / (2^i))$

the total extra time we spend in level  $i$  is

$(2^i) \cdot C \cdot (n / (2^i)) = C \cdot n$

number of levels =  $\lg(n)+1$  (assuming  $n=2^k$ )

The sum of extra time over all the levels :

$C \cdot n + C \cdot n + \dots + C \cdot n$  ( $\lg(n) + 1$  times)

$= C \cdot n (\lg(n)+1) = C \cdot n \cdot \lg(n) + C \cdot n = O(n \lg(n))$

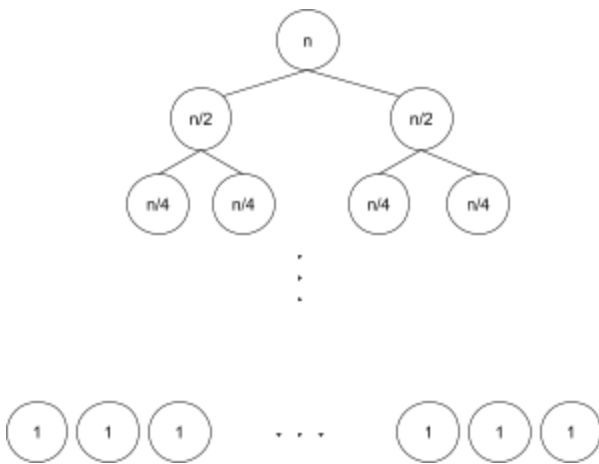
Mystery6: Final result :  $O(n^2)$

```

void mystery6(int n){
    for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
            cout << i << " " << j << endl;
    if (n>1){
        mystery6(n/2);
        mystery6(n/2);
    }
}

```

The graph of complexity:



Computations:

In level  $i$ :

$2^i$  children

size of the child is  $n/(2^i)$

the extra time we spend for each child is

$$C \cdot (n/(2^i))^2 = O((n/(2^i))^2)$$

the total extra time we spend in level  $i$  is

$$(2^i) \cdot C \cdot (n/(2^i))^2 = C \cdot (n^2)/(2^i)$$

number of levels =  $\lg(n)+1$  (assuming  $n=2^k$ )

The sum of extra time over all the levels :

$$C \cdot n^2 (1/1+1/2+\dots+1/(2^i)+\dots+1/2^{\lg(n)})$$

$$< C \cdot (n^2) (1/1+1/2+\dots+1/(2^i)+\dots)$$

$$\leq C \cdot (n^2) \cdot 2 = O(n^2)$$