1 Priori Analysis and Posterori Testing

Priori Analysis	Posterori Analysis
Algorithm	Program
Independent of Language	Language dependent
Hardware Independent	Hardware Dependent
Time & Space Function	Exact Time & Bytes

2 Characteristics of Algorithm

1. Input
2. Output
3. Definiteness
4. Finiteness
5. Effectiveness

3 Frequency Count Method

3.0.1 example

8	3	9	7	2
0	1	2	3	4

```
sum(A,n) {                                  .
   s = 0;                                   1
   for( i=0 ; i<n ; i++ ) {                 n+1
      s += A[i];                            n
   }                                        .
   return s;                                1
}                                           .
```

Time Complexity :

$$\Rightarrow f(n) = 2n + 3 \Rightarrow O(n)$$

Space Complexity :

$$\left.\begin{array}{ll} A & \rightarrow n \\ n & \rightarrow 1 \\ s & \rightarrow 1 \\ i & \rightarrow 1 \end{array}\right\} \Rightarrow s(n) = n + 3 \Rightarrow O(n)$$

3.0.2 example

```
add(A,B,n) {                                             .
   for( i=0 ; i<n ; i++ ) {                              n+1
      for( j=0 ; j<n ; j++ ) {                           n * (n+1)
         C[i,j] = A[i,j] + B[i,j];                       n * n
      }                                                  .
   }                                                     .
}                                                        .
```

Time Complexity :

$$\Rightarrow f(n) = 2n^2 + 2n + 1 \Rightarrow O(n^2)$$

Space Complexity :

$$\left.\begin{array}{ll} A & \rightarrow n^2 \\ B & \rightarrow n^2 \\ C & \rightarrow n^2 \\ n & \rightarrow 1 \\ i & \rightarrow 1 \\ j & \rightarrow 1 \end{array}\right\} \Rightarrow s(n) = 3n^2 + 3 \Rightarrow O(n^2)$$

3.0.3 example

```
multiply(A,B,n) {                                  .
    for( i=0 ; i<n ; i++ ) {                       n+1
        for( j=0 ; j<n ; j++ ) {                   n * (n+1)
            C[i,j] = 0;                            n * n
            for( k=0 ; k<n ; k++ ) {               n * n * (n+1)
                C[i,j] += A[i,k] * B[k,j];         n * n * n
            }                                      .
        }                                          .
    }                                              .
}                                                  .
```

Time Complexity :

$$\Rightarrow f(n) = 2n^3 + 3n^2 + 2n + 1 \Rightarrow O(n^3)$$

Space Complexity :

$$\left.\begin{array}{ll} A & \rightarrow n^2 \\ B & \rightarrow n^2 \\ C & \rightarrow n^2 \\ n & \rightarrow 1 \\ i & \rightarrow 1 \\ j & \rightarrow 1 \\ k & \rightarrow 1 \end{array}\right\} \Rightarrow s(n) = 3n^2 + 4 \Rightarrow O(n^2)$$

4 Time Complexity 1th

4.0.1 example

```
for( i=0 ; i<n ; i++ ) {            n+1
    statement;                      n
}                                   .
```

Time Complexity : $O(n)$

4.0.2 example

```
for( i=n ; i>0 ; i-- ) {            n+1
    statement;                      n
}                                   .
```

Time Complexity : $O(n)$

4.0.3 example

```
for( i=0 ; i<n ; i+=2 ) {           (n+1)/2
    statement;                      n/2
}                                   .
```

Time Complexity : $O(n)$

4.0.4 example

```
for( i=0 ; i<n ; i+=20 ) {          (n+1)/20
   statement;                       n/20
}                                   .
```

Time Complexity : $O(n)$

4.0.5 example

```
for( i=0 ; i<n ; i++ ) {                        n+1
   for( j=0 ; j<n ; j++ ) {                     n * (n+1)
      statement ;                               n * n
   }                                            .
}                                               .
```

Time Complexity : $O(n^2)$

4.0.6 example

```
for( i=0 ; i<n ; i++ ) {                        n+1
   for( j=0 ; j<i ; j++ ) {                     n * (n+1)
      statement ;                               n * n
   }                                            .
}                                               .
```

Time Complexity :

$$\Rightarrow 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$$

4.0.7 example

```
for( i=0 ; i<n ; i*=2 ) {
    statement;
}
```

i
1
$1 \times 2 = 2$
$2 \times 2 = 2^2$
$2^2 \times 2 = 2^3$
$\vdots$
2^k

$$2^k < n$$
$$\Rightarrow \log_2^{2^k} < \log_2^n$$

$\Rightarrow$ Time Complexity : $O(\log_2^n)$

if n = 8 :

i
$1 \rightarrow 2 \rightarrow 4 \rightarrow 8$

$$\left\lceil \log_2^8 \right\rceil = 3$$

if n = 10 :

$$\begin{array}{c} \text{i} \\ \hline 1 \to 2 \to 4 \to 8 \to 16 \end{array}$$

$$\lceil \log_2^{10} \rceil = \lceil 3.2 \rceil = 4$$

4.0.8 example

```
for( i=0 ; i<n ; i/=2 ) {
    statement;
}
```

$$\begin{array}{c} \text{i = n} \\ \hline \frac{n}{2} \to \frac{n}{2^2} \to \frac{n}{2^3} \to \frac{n}{2^4} \to \cdots \to \frac{n}{2^n} \end{array}$$

$$\begin{aligned} & \frac{n}{2^k} < 1 \\ \Rightarrow \quad & n < 2^k \\ \Rightarrow \quad & \log_2^n < \log_2^{2^k} \\ \Rightarrow \quad & \log_2^n < k \\ \Rightarrow \quad & k > \log_2^n \end{aligned}$$

$\Rightarrow$ Time Complexity : $O(\log_2^n)$

4.0.9 example

```
for( i=0 ; i*i<n ; i++ ) {
    statement;
}
```

$$i \times i \geq n$$
$$i^2 \geq n$$
$$i \geq \sqrt{n}$$

$\Rightarrow$ Time Complexity : $O(\sqrt{n})$

4.0.10 example

```
for( i=0 ; i<n ; i++ ) {            n+1
    statement;                      n
}                                   .
                                    .
for( j=0 ; j<n ; j++ ) {            n+1
    statement;                      n
}                                   .
```

$\Rightarrow$ Time Complexity : $O(n)$

4.0.11 example

```
p = 0;                                  .
for( i=0 ; i<n ; i*=2 ) {               .
   p += 1;                              p = log(n)
}                                       .
                                        .
for( j=0 ; j<p ; j*=2 ) {               .
   statement;                           log(p)
}                                       .
```

$\Rightarrow$ Time Complexity : $O(\log(\log(n)))$

4.0.12 example

```
for( i=0 ; i<n ; i++ ) {                          n
   for( j=0 ; j<n ; j*=2) {                       n * log(n)
      statement ;                                 n * log(n)
   }                                              .
}                                                 .
```

$\Rightarrow$ Time Complexity : $O(n\log(n))$

5 Summary

```
for( i=0 ; i<n ; i++ )
```

$O(n)$

```
for( i=0 ; i<n ; i+=2 )
```

$O(n)$

```
for( i=n ; i>1 ; i-- )
```

$O(n)$

```
for( i=1 ; i<n ; i*=2 )
```

$O(\log_2^n)$

```
for( i=0 ; i<n ; i*=3 )
```

$O(\log_3^n)$

```
for( i=n ; i>1 ; i/=2 )
```

$O(\log_2^n)$

6 Analysis of while loop

6.0.1 example

```
for( i=0 ; i*i<n ; i++ ) {          n+1
    statement;                      n
}                                   .
```

$f(n) = 2n + 1 \Rightarrow$ Time Complexity : $O(n)$

```
i = 0;                              1
while(i < n) {                      n+1
    statement;                      n
    i++;                            n
}                                   .
```

$f(n) = 3n + 2 \Rightarrow$ Time Complexity : $O(n)$

6.0.2 example

```
i = 0;
while(i < n) {
    statement;
    i*=2;
}
```

i
1
$1 \times 2 = 2$
$2 \times 2 = 2^2$
$2^2 \times 2 = 2^3$
$\vdots$
2^k

$$
\begin{aligned}
& 2^k < n \\
\Rightarrow \quad & \log_2^{2^k} < \log_2^n \\
\Rightarrow \quad & k < \log_2^n
\end{aligned}
$$

$\Rightarrow$ Time Complexity : $O(\log_2^n)$

6.0.3 example

```
i = 1;
k = 1;
while(k < n) {
    statement;
    k+=i;
    i++;
}
```

i	k
1	1
2	1+1
3	$1 + 1 + 2$
4	$1 + 1 + 2 + 3$
$\vdots$	$\vdots$
m	$1 + 1 + 2 + 3 + \ldots +$ m $= \dfrac{m(m+1)}{2}$

$$
\begin{aligned}
& k < n \\
\Rightarrow \quad & \frac{m(m+1)}{2} < n \\
\Rightarrow \quad & m^2 < n \\
\Rightarrow \quad & m < \sqrt{n}
\end{aligned}
$$

$\Rightarrow$ Time Complexity : $O(\sqrt{n})$

7 Analysis of if statement

```
Test(n) {                            .
   if(n < 5) {                       .
      print(n);                      1
   } else {                          .
      for(i=0;i<n;i++) {             n+1
         print(n);                   n
      }                              .
   }                                 .
}                                    .
```

$$\text{Time Complexity} \Rightarrow \begin{cases} Worst & \rightarrow O(1) \\ Best & \rightarrow O(n) \end{cases}$$

8 Classes of Functions

$O(1)$	constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential
$O(3^n)$	
$O(n!)$	
$O(n^n)$	

9 Compare Classes of Functions

$$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < 2^n < 3^n < n! < n^n$$

10 Asymptotic Notation

O	big-oh	Upper Bound
Ω	big-Omega	Lower Bound
θ	theta	Average Bound

10.1 Big oh - O

the function $f(n) = O(g(n))$ if there exists c and n_0 such that

$$f(n) \leq c \times g(n) \quad \forall \quad n \geq n_0$$

10.1.1 example

$f(n) = 2n + 3$

if $c = 10$:

$$\underbrace{2n+3}_{f(n)} \leq \underbrace{10}_{c} \times \underbrace{n}_{g(n)}$$

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

10.2 Big omega - Ω

the function $f(n) = \Omega(g(n))$ if there exists c and n_0 such that

$$f(n) \geq c \times g(n) \quad \forall \quad n \geq n_0$$

10.2.1 example

$f(n) = 2n + 3$

if $c = 1$:

$$\underbrace{2n+3}_{f(n)} \geq \underbrace{1}_{c} \times \underbrace{n}_{g(n)}$$

$$\Rightarrow f(n) = \Omega(n)$$

10.3 theta - θ

the function $f(n) = \theta(g(n))$ if there exists c_1 , c_2 and n_0 such that

$$c_1 \times g(n) \leq f(n) \leq c_2 \times g(n) \quad \forall \quad n \geq n_0$$

10.3.1 example

$f(n) = 2n + 3$

if $c_1 = 1$ and $c_2 = 10$:

$$\underbrace{1}_{c_1} \times \underbrace{n}_{g(n)} \leq \underbrace{2n+3}_{f(n)} \leq \underbrace{10}_{c_2} \times \underbrace{n}_{g(n)}$$

$$\Rightarrow f(n) = \theta(n)$$

11 Asymptotic Notation - examples

11.0.1 example

$f(n) = 2n^2 + 3n + 4$

if $c = 9$ and $g(n) = n^2$:

$$\underbrace{2n^2 + 3n + 4}_{f(n)} \leq \underbrace{9}_{c} \times \underbrace{n^2}_{g(n)}$$

$$\Rightarrow f(n) = O(n^2)$$

if $c = 1$ and $g(n) = n^2$:

$$\underbrace{2n^2 + 3n + 4}_{f(n)} \geq \underbrace{1}_{c} \times \underbrace{n^2}_{g(n)}$$

$$\Rightarrow f(n) = \Omega(n^2)$$

if $c_1 = 1$ and $c_2 = 9$ and $g(n) = n^2$:

$$\underbrace{1}_{c_1} \times \underbrace{n^2}_{g(n)} \leq \underbrace{2n^2 + 3n + 4}_{f(n)} \leq \underbrace{9}_{c_2} \times \underbrace{n^2}_{g(n)}$$

$$\Rightarrow f(n) = \theta(n^2)$$

11.0.2 example

$f(n) = n^2 \log n + n$

if $c = 10$ and $g(n) = n^2 \log n$:

$$\underbrace{n^2 \log n + n}_{f(n)} \leq \underbrace{10}_{c} \times \underbrace{n^2 \log n}_{g(n)}$$

$$\Rightarrow f(n) = O(n^2 \log n)$$

if $c = 1$ and $g(n) = n^2 \log n$:

$$\underbrace{n^2 \log n + n}_{f(n)} \geq \underbrace{1}_{c} \times \underbrace{n^2 \log n}_{g(n)}$$

$$\Rightarrow f(n) = \Omega(n^2 \log n)$$

if $c_1 = 1$ and $c_2 = 10$:

$$\underbrace{1}_{c_1} \times \underbrace{n^2 \log n}_{g(n)} \leq \underbrace{n^2 \log n + n}_{f(n)} \leq \underbrace{10}_{c_2} \times \underbrace{n^2 \log n}_{g(n)}$$

$$\Rightarrow f(n) = \theta(n^2 \log n)$$

11.0.3 example

$$f(n) = n!$$

$$n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$$

$$1 \times 1 \times 1 \times \cdots \times 1 \leq 1 \times 2 \times 3 \times \cdots \times (n-1) \times n \leq n \times n \times n \times \cdots \times n$$

$$1 \leq n! \leq n^n$$

$$\Rightarrow \begin{cases} O(n^n) \\ \Omega(1) \end{cases}$$

11.0.4 example

$$f(n) = \log{(n!)}$$

$$\log{(1 \times 1 \times 1 \times \cdots \times 1)} \leq \log{(1 \times 2 \times 3 \times \cdots \times (n-1) \times n)} \leq \log{(n \times n \times n \times \cdots \times n)}$$

$$1 \leq \log{(n!)} \leq \log{(n^n)} \Rightarrow 1 \leq \log{(n!)} \leq n \log{(n)}$$

$$\Rightarrow \begin{cases} O(n \log{(n)}) \\ \Omega(1) \end{cases}$$

12 Properties of Asymptotic Notations

12.1 General Properties

if $f(n)$ is $O(g(n))$ then $a \times f(n)$ is $O(g(n))$

e.g :
$f(n) = 2n^2 + 5$ is $O(n^2)$
then $7 \times f(n) = 7 \times (2n^2 + 5)$ is $O(n^2)$

if $f(n)$ is $\Omega(g(n))$ then $a \times f(n)$ is $\Omega(g(n))$

e.g :
$f(n) = 2n^2 + 5$ is $\Omega(n^2)$
then $7 \times f(n) = 7 \times (2n^2 + 5)$ is $\Omega(n^2)$

12.2 Reflexive

if $f(n)$ is given then we have $O(f(n))$

e.g :
$f(n) = n^2 \Rightarrow f(n) = O(n^2)$

12.3 Transitive

if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n) = O(h(n))$

e.g : $f(n) = n \quad g(n) = n^2 \quad h(n) = n^3$

$$\left.\begin{array}{l} n \rightarrow O(n^2) \\ n^2 \rightarrow O(n^3) \end{array}\right\} \Rightarrow f(n) = O(n^3)$$

12.4 Transitive

if $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

e.g :
$f(n) = n^2 \quad g(n) = n^2$
$f(n) = \theta(n^2)$
$g(n) = \theta(n^2)$

12.5 Transpose Symmetric

if $f(n)$ is $O(g(n))$ then $g(n)$ is $\Omega(f(n))$

e.g :
$f(n) = n \quad \Rightarrow \quad f(n) = O(n^2)$
$g(n) = n^2 \quad \Rightarrow \quad g(n) = \Omega(n)$

12.6 point

if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ then we have :

$$g(n) \leq f(n) \leq g(n) \quad \Rightarrow \quad f(n) = \theta(g(n))$$

12.7 point

if $f(n) = O(g(n))$ and $d(n) = O(e(n))$ then we have :

$$f(n) + d(n) = O(max(g(n), e(n)))$$

e.g :
$f(n) = n^2 \rightarrow O(n^2)$
$d(n) = n \rightarrow O(n)$

$$f(n) + d(n) = n^2 + n = O(n^2)$$

12.8 point

if $f(n) = O(g(n))$ and $d(n) = O(e(n))$ then we have :

$$f(n) \times d(n) = O(g(n) \times e(n))$$

e.g :
$f(n) = n^2 \to O(n^2)$
$d(n) = n \to O(n)$

$$f(n) \times d(n) = n^2 \times n = n^3 = O(n^3)$$

13 Comparision of functions

$$f(n) = 3n^{\sqrt{n}} \qquad g(n) = 2^{\sqrt{n}\log_2^n}$$

$$a^{\log_c^b} = b^{\log_c^a}$$

$$\begin{aligned} g(n) &= 2^{\sqrt{n}\log_2^n} \\ &= 2^{\log_2^{n^{\sqrt{n}}}} \\ &= (n^{\sqrt{n}})^{\log_2^2} \\ &= n^{\sqrt{n}} \end{aligned}$$

$$\Rightarrow \quad 3n^{\sqrt{n}} > n^{\sqrt{n}}$$

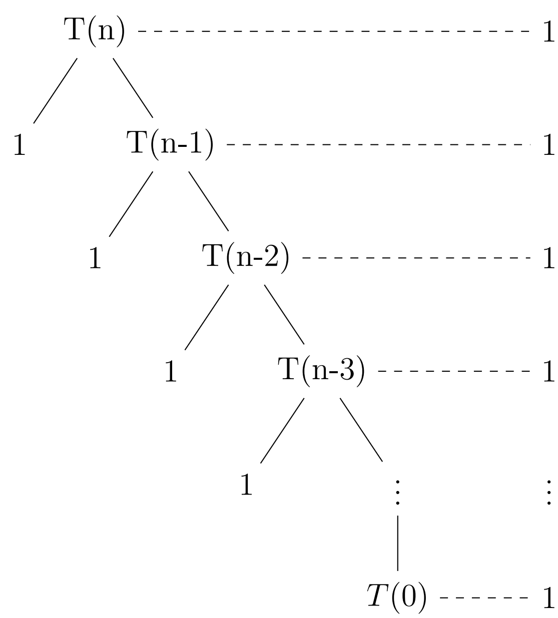
14 Recurrence Relation

14.0.1 example

```
Test(n) {                                          T(n)
   if(n > 0) {                                     .
      print(n);                                    1
      Test(n-1);                                   T(n-1)
   }                                               .
}                                                  .
```

$$T(n) = T(n-1) + 1$$

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + 1 & n > 0 \end{cases}$$



$$1 + 1 + 1 + \cdots + 1 = n \Rightarrow f(n) = O(n)$$

$$
\begin{aligned}
& T(n) = T(n-1) + 1 \\
& T(n-1) = T(n-2) + 1
\end{aligned}
$$

substitute

$$
\begin{aligned}
\Rightarrow \quad & T(n) = [T(n-2) + 1] + 1 \\
\Rightarrow \quad & T(n) = T(n-2) + 2 \\
& T(n-2) = T(n-3) + 1
\end{aligned}
$$

substitute

$$
\begin{aligned}
\Rightarrow \quad & T(n) = [T(n-3) + 1] + 2 \\
\Rightarrow \quad & T(n) = T(n-3) + 3
\end{aligned}
$$

continue for k times

$$
\Rightarrow \quad T(n) = T(n-k) + k
$$

$$
\begin{aligned}
& k = n \\
& \Rightarrow \quad T(n) = T(n-n) + n \\
& \Rightarrow \quad T(n) = T(0) + n \\
& \Rightarrow \quad T(n) = 1 + n
\end{aligned}
$$

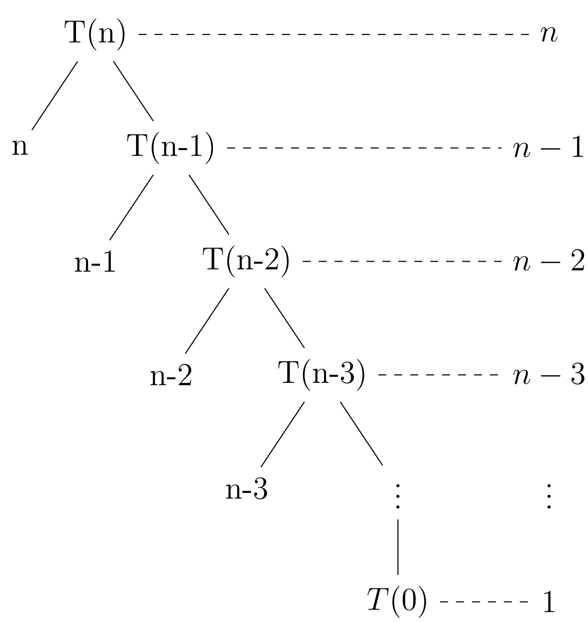
$$
\left.\begin{aligned}
f(n) = O(n) \\
f(n) = \Omega(n)
\end{aligned}\right\} \Rightarrow \quad f(n) = \theta(n)
$$

14.0.2 example

```
Test(n) {                                       T(n)
   if(n > 0) {                                  .
      for(i=0;i<n;i++) {                        n+1
         print(n);                              n
      }                                         .
      Test(n-1);                                T(n-1)
   }                                            .
}                                               .
```

$$T(n) = T(n-1) + n$$

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + n & n > 0 \end{cases}$$



$$0 + 1 + 2 + \cdots + (n-2) + (n-1) + n = \frac{n(n+1)}{2} \Rightarrow f(n) = O(n^2)$$

$$T(n) = T(n-1) + n$$

$$T(n-1) = T(n-2) + n - 1$$

substitute

$$\Rightarrow \quad T(n) = [T(n-2) + n - 1] + n$$

$$\Rightarrow \quad T(n) = T(n-2) + n - 1 + n$$

$$T(n-2) = T(n-3) + n - 2$$

substitute

$$\Rightarrow \quad T(n) = [T(n-3) + n - 2] + n - 1 + n$$

$$\Rightarrow \quad T(n) = T(n-3) + n - 2 + n - 1 + n$$

continue for k times

$$\Rightarrow \quad T(n) = T(n-k) + (n-(k-1)) + \cdots + n - 2 + n - 1 + n$$

$$k = n$$

$$\Rightarrow \quad T(n) = T(n-n) + n - (n-1) + \cdots + n - 2 + n - 1 + n$$

$$\Rightarrow \quad T(n) = T(0) + 1 + \cdots + n - 2 + n - 1 + n$$

$$\Rightarrow \quad T(n) = T(0) + 1 + 2 + 3 + \cdots + n - 2 + n - 1 + n$$

$$\Rightarrow \quad T(n) = 1 + \frac{n(n+1)}{2}$$

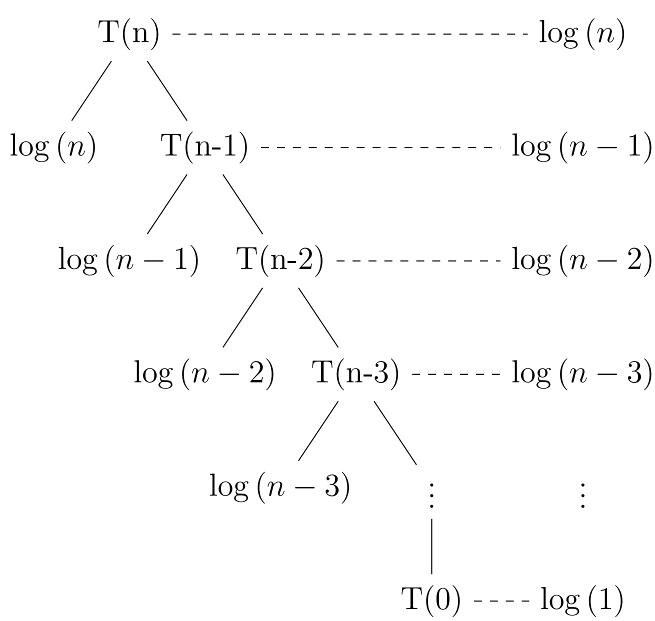
$$\left.\begin{aligned} f(n) &= O(n^2) \\ f(n) &= \Omega(n^2) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(n^2)$$

14.0.3 example

```
Test(n) {                                          T(n)
   if(n > 0) {                                     .
      for(i=0;i<n;i*=2) {                          .
         print(n);                                 log(n)
      }                                            .
      Test(n-1);                                   T(n-1)
   }                                               .
}                                                  .
```

$$T(n) = T(n-1) + \log(n)$$

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + \log(n) & n > 0 \end{cases}$$



$$\log(n) + \log(n-1) + \log(n-2) + \cdots + \log(2) + \log(1)$$

$$= \log(n) \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

$$= \log(n!)$$

$$n! < n^n$$

$$\underbrace{n \times (n-1) \times \cdots \times 2 \times 1}_{n!} < \underbrace{n \times n \times n \times \cdots \times n}_{n^n}$$

$$O(\log(n!)) < O(n \log(n))$$

$$\Rightarrow f(n) = O(n \log(n))$$

$$T(n) = T(n-1) + \log(n)$$

$$T(n-1) = T(n-2) + \log(n-1)$$

substitute

$$\Rightarrow \quad T(n) = [T(n-2) + \log(n-1)] + \log(n)$$

$$\Rightarrow \quad T(n) = T(n-2) + \log(n-1) + \log(n)$$

$$T(n-2) = T(n-3) + \log(n-2)$$

substitute

$$\Rightarrow \quad T(n) = [T(n-3) + \log(n-2)] + \log(n-1) + \log(n)$$

$$\Rightarrow \quad T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log(n)$$

continue for k times

$$\Rightarrow \quad T(n) = T(n-k) + \log(n-(k-1)) + \cdots + \log(n-2) + \log(n-1) + \log(n)$$

$$\Rightarrow \quad T(n) = T(n-k) + \log(n-(k-1)) + \cdots + \log(n-2) + \log(n-1) + \log(n)$$

$k = n$

$$\Rightarrow \quad T(n) = T(n-n) + \log(n-(n-1)) + \cdots + \log(n-2) + \log(n-1) + \log(n)$$

$$\Rightarrow \quad T(n) = T(n-n) + \log(1) + \cdots + \log(n-2) + \log(n-1) + \log(n)$$

$$\Rightarrow \quad T(n) = T(0) + \log(1) + \log(2) + \cdots + \log(n-2) + \log(n-1) + \log(n)$$

$$\Rightarrow \quad T(n) = 1 + \log(n!)$$

$$\left.\begin{aligned} f(n) &= O(n\log(n)) \\ f(n) &= \Omega(n\log(n)) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(n\log(n))$$

15 Summary

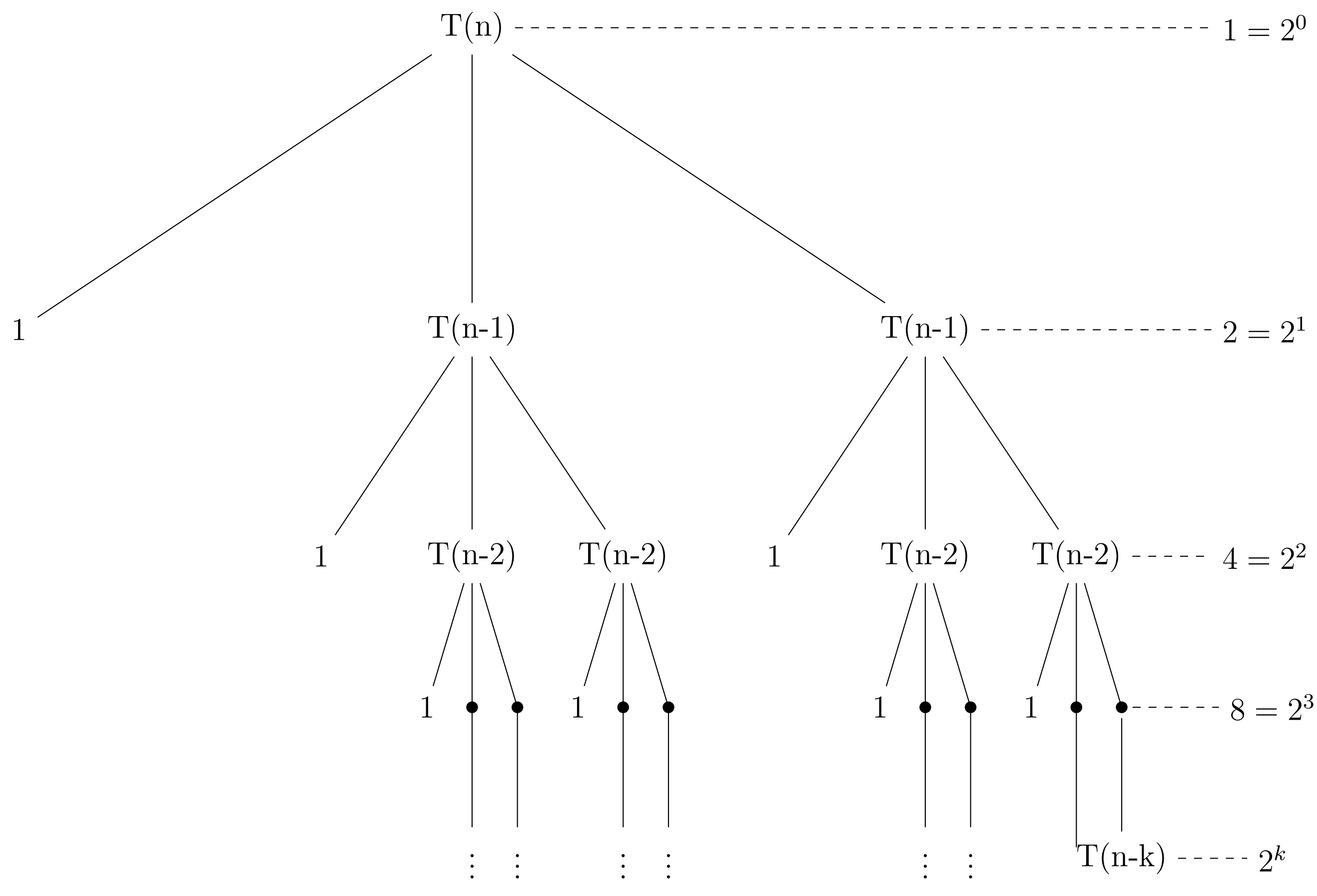
$T(n) = T(n-1) + 1$	$O(n)$
$T(n) = T(n-1) + n$	$O(n^2)$
$T(n) = T(n-1) + \log(n)$	$O(n\log(n))$
$T(n) = T(n-1) + n^2$	$O(n^3)$
$T(n) = T(n-1) + 1$	$O(n)$
$T(n) = T(n-2) + 1$	$O(\frac{n}{2}) = O(n)$
$T(n) = T(n-100) + n$	$O(n^2)$

15.0.1 example

```
Test(n) {                                   T(n)
   if(n > 0) {                              .
      print(n);                             1
      Test(n-1);                            T(n-1)
      Test(n-1);                            T(n-1)
   }                                        .
}                                           .
```

$$T(n) = 2T(n-1) + 1$$

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1) + 1 & n > 0 \end{cases}$$



$$a + ar + ar^2 + ar^3 + \cdots + ar^k = \frac{a(r^{k+1} - 1)}{r - 1}$$

$$\left.\begin{aligned} a &= 1 \\ r &= 2 \end{aligned}\right\} \Rightarrow 1 + 2 + 2^2 + 2^3 + \cdots + 2^k = \frac{1 \times (2^{k+1} - 1)}{2 - 1} = 2^{k+1} - 1$$

$$\begin{aligned} &n = k \\ \Rightarrow &f(n) = 2^{n+1} - 1 \\ \Rightarrow &f(n) = O(2^n) \end{aligned}$$

$$\begin{aligned}
&T(n) = 2T(n-1) + 1 \\
&T(n-1) = 2T(n-2) + 1 \\
\text{substitute} \\
\Rightarrow \quad &T(n) = 2[2T(n-2) + 1] + 1 \\
\Rightarrow \quad &T(n) = 2^2T(n-2) + 2 + 1 \\
&T(n-2) = 2T(n-3) + 1 \\
\text{substitute} \\
\Rightarrow \quad &T(n) = 2^2[2T(n-3) + 1] + 2 + 1 \\
\Rightarrow \quad &T(n) = 2^3T(n-3) + 2^2 + 2 + 1 \\
\text{continue for k times} \\
\Rightarrow \quad &T(n) = 2^kT(n-k) + 2^{k-1} + \cdots + 2^3 + 2^2 + 2 + 1
\end{aligned}$$

$$\Rightarrow \quad T(n) = 2^k T(n-k) + 2^{k-1} + \cdots + 2^3 + 2^2 + 2 + 1$$

$$k = n$$

$$\Rightarrow \quad T(n) = 2^n T(n-n) + 2^{n-1} + \cdots + 2^3 + 2^2 + 2 + 1$$

$$\Rightarrow \quad T(n) = 2^n T(n-n) + 2^{n-1} + \cdots + 2^3 + 2^2 + 2 + 1$$

$$\Rightarrow \quad T(n) = 2^n \underbrace{T(0)}_{1} + 2^n - 1$$

$$\Rightarrow \quad T(n) = 2^n + 2^n - 1$$

$$\Rightarrow \quad T(n) = 2 \times 2^n - 1$$

$$\Rightarrow \quad T(n) = 2^{n+1} - 1$$

$$\left.\begin{array}{l} f(n) = O(2^n) \\ f(n) = \Omega(2^n) \end{array}\right\} \Rightarrow \quad f(n) = \theta(2^n)$$

16 Summary

$T(n) = T(n-1) + 1$	$O(n)$
$T(n) = T(n-1) + n$	$O(n^2)$
$T(n) = T(n-1) + \log(n)$	$O(n \log(n))$
$T(n) = 2T(n-1) + 1$	$O(2^n)$
$T(n) = 3T(n-1) + 1$	$O(3^n)$
$T(n) = 2T(n-1) + n$	$O(n2^n)$

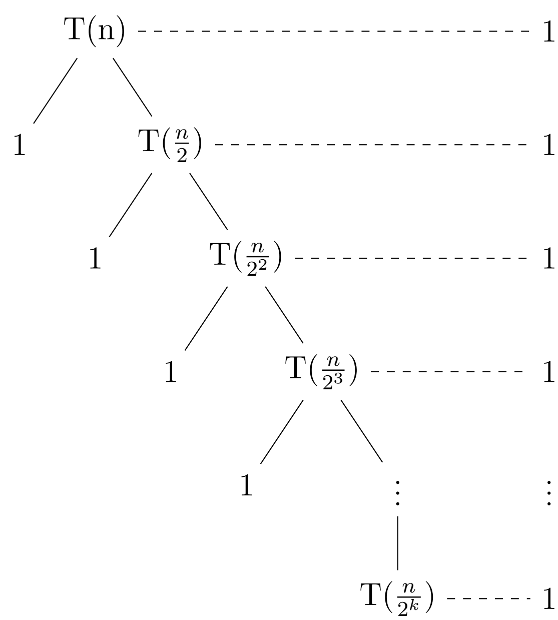
$$\left.\begin{array}{l} T(n) = aT(n-b) + f(n) \\ a > 0 \ \& \ b > 0 \ and \ f(n) = O(n^k) \ where \ k \geq 0 \end{array}\right\} \Rightarrow \begin{cases} a < 1 \rightarrow O(n^k) = O(f(n)) \\ a = 1 \rightarrow O(n^{k+1}) = O(n \times f(n)) \\ a > 1 \rightarrow O(n^k a^{n/b}) = O(f(n) \times a^{n/b}) \end{cases}$$

16.0.1 example

```
Test(n) {                                                    T(n)
   if(n > 1) {                                               .
      print(n);                                              1
      Test(n/2);                                             T(n/2)
   }                                                         .
}                                                            .
```

$$T(n) = T(\frac{n}{2}) + 1$$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$



k steps

$$\frac{n}{2^k} = 1$$
$$n = 2^k$$
$$\log_2^n = \log_2^{2^k}$$
$$k = \log_2^n$$
$$k = \log n$$
$$\Rightarrow f(n) = O(\log(n))$$

$$T(n) = T(\frac{n}{2}) + 1$$
$$T(\frac{n}{2}) = T(\frac{n}{2^2}) + 1$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^2}) + 1] + 1$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^2}) + 2$$
$$T(\frac{n}{2^2}) = T(\frac{n}{2^3}) + 1$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^3}) + 1] + 2$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^3}) + 3$$

continue for k times

$$\Rightarrow \quad T(n) = T(\frac{n}{2^k}) + k$$

$$\frac{n}{2^k} = 1 \to n = 2^k \to k = \log(n)$$

$$\Rightarrow \quad T(n) = T(1) + \log(n)$$

$$\Rightarrow \quad T(n) = 1 + \log(n)$$

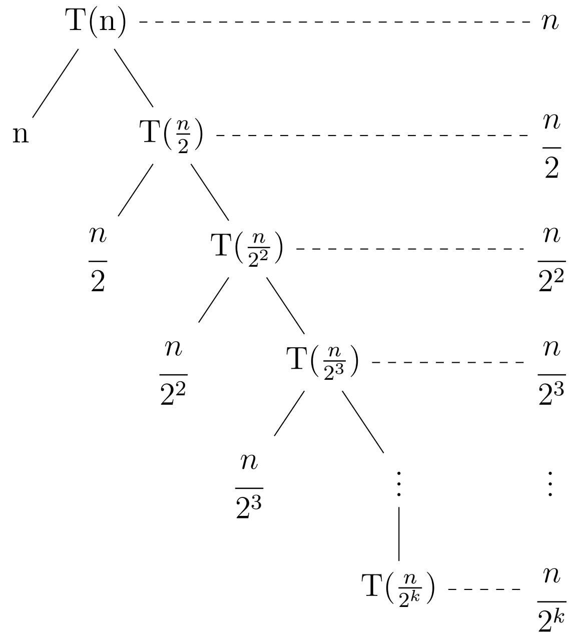
$$\left.\begin{aligned} f(n) &= O(\log(n)) \\ f(n) &= \Omega(\log(n)) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(\log(n))$$

16.0.2 example

```
Test(n) {                                   T(n)
   if(n > 1) {                              .
      for(i=0;i<n;i++) {                    n+1
         print(n);                          n
      }                                     .
      Test(n/2);                            T(n/2)
   }                                        .
}                                           .
```

$$T(n) = T(\frac{n}{2}) + n$$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + n & n > 1 \end{cases}$$



$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots + \frac{n}{2^k}$$

$$T(n) = n\left[\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^k}\right]$$

$$T(n) = n\left[\sum_{i=0}^{k} \frac{1}{2^i}\right]$$

$$\sum_{i=0}^{k} \frac{1}{2^i} = 1$$

$$T(n) = n \times 1$$

$$T(n) = n$$

$$\left.\begin{aligned} f(n) &= O(n) \\ f(n) &= \Omega(n) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(n)$$

$$T(n) = T(\frac{n}{2}) + n$$
$$T(\frac{n}{2}) = T(\frac{n}{2^2}) + \frac{n}{2}$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^2}) + \frac{n}{2}] + n$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^2}) + \frac{n}{2} + n$$
$$T(\frac{n}{2^2}) = T(\frac{n}{2^3}) + \frac{n}{2^2}$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^3}) + \frac{n}{2^2}] + \frac{n}{2} + n$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^3}) + \frac{n}{2^2} + \frac{n}{2} + n$$

continue for k times

$$\Rightarrow \quad T(n) = T(\frac{n}{2^k}) + \frac{n}{2^{k-1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n$$

$$\frac{n}{2^k} = 1 \to n = 2^k \to k = \log{(n)}$$
$$\Rightarrow \quad T(n) = T(1) + \frac{n}{2^{k-1}} + \cdots + \frac{n}{2^2} + \frac{n}{2} + n$$
$$\Rightarrow \quad T(n) = 1 + n\left[\frac{1}{2^{k-1}} + \cdots + \frac{1}{2^2} + \frac{1}{2} + 1\right]$$

$$\frac{1}{2^{k-1}} + \cdots + \frac{1}{2^2} + \frac{1}{2} = 1$$

$$\Rightarrow \quad T(n) = 1 + n\,[1 + 1]$$
$$\Rightarrow \quad T(n) = 1 + 2n$$

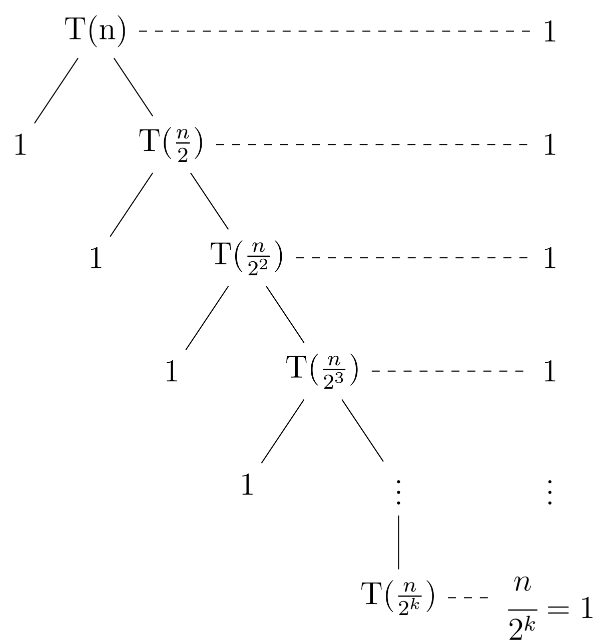
$$\left.\begin{array}{l} f(n) = O(n) \\ f(n) = \Omega(n) \end{array}\right\} \Rightarrow \quad f(n) = \theta(n)$$

16.0.3 example

```
Test(n) {                                T(n)
   if(n > 1) {                           .
      print(n);                          1
      Test(n/2);                         T(n/2)
   }                                     .
}                                        .
```

$$T(n) = T(\frac{n}{2}) + 1$$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$



k steps

$$\frac{n}{2^k} = 1$$
$$n = 2^k$$
$$\log_2^n = \log_2^{2^k}$$
$$k = \log_2^n$$
$$k = \log n$$
$$\Rightarrow f(n) = O(\log(n))$$

$$T(n) = T(\frac{n}{2}) + 1$$
$$T(\frac{n}{2}) = T(\frac{n}{2^2}) + 1$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^2}) + 1] + 1$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^2}) + \frac{n}{2} + n$$
$$T(\frac{n}{2^2}) = T(\frac{n}{2^3}) + 1$$

substitute

$$\Rightarrow \quad T(n) = [T(\frac{n}{2^3}) + 1] + 1 + 1$$

continue for k times

$$\Rightarrow \quad T(n) = T(\frac{n}{2^k}) + \underbrace{1 + \cdots + 1 + 1 + 1}_{\text{k times}}$$
$$\Rightarrow \quad T(n) = T(\frac{n}{2^k}) + k$$

$$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow k = \log(n)$$

$$\Rightarrow \quad T(n) = T(1) + \log(n)$$

$$\Rightarrow \quad T(n) = 1 + \log(n)$$

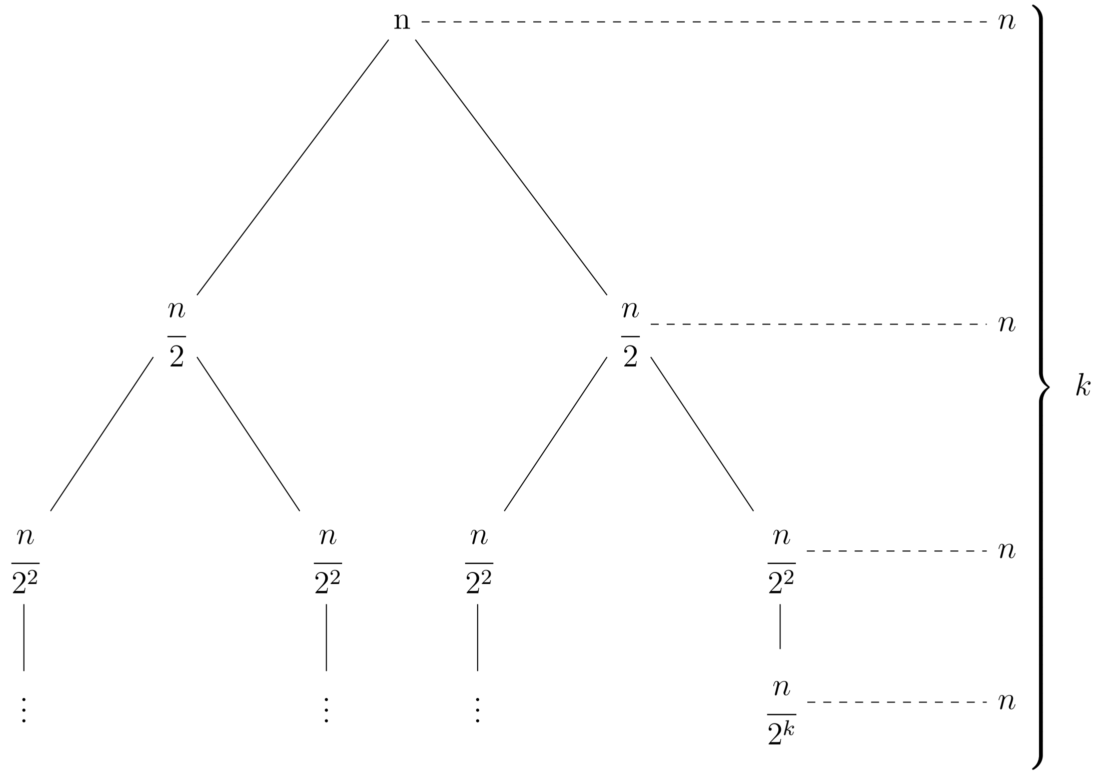
$$\left.\begin{aligned} f(n) &= O(\log(n)) \\ f(n) &= \Omega(\log(n)) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(\log(n))$$

16.0.4 example

```
Test(n) {                                   T(n)
  if(n > 1) {                               .
    for(i=0;i<n;i++) {                      n+1
      print(n);                             n
    }                                       .
    Test(n/2);                              T(n/2)
    Test(n/2);                              T(n/2)
  }                                         .
}                                           .
```

$$T(n) = 2\,T(\frac{n}{2}) + n$$

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\frac{n}{2}) + n & n > 1 \end{cases}$$



$$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow k = \log(n)$$

$$T(n) = 2T(\frac{n}{2}) + n$$
$$T(\frac{n}{2}) = 2T(\frac{n}{2^2}) + \frac{n}{2}$$

substitute

$$\Rightarrow \quad T(n) = 2[2T(\frac{n}{2^2}) + \frac{n}{2}] + n$$
$$\Rightarrow \quad T(n) = 2^2 T(\frac{n}{2^2}) + n + n$$
$$T(\frac{n}{2^2}) = 2T(\frac{n}{2^3}) + \frac{n}{2^3}$$

substitute

$$\Rightarrow \quad T(n) = 2^2[2T(\frac{n}{2^3}) + \frac{n}{2^2}] + n + n$$
$$\Rightarrow \quad T(n) = 2^3 T(\frac{n}{2^3}) + n + n + n$$

continue for k times

$$\Rightarrow \quad T(n) = T(\frac{n}{2^k}) + \underbrace{n + \cdots + n + n + n}_{\text{k times}}$$
$$\Rightarrow \quad T(n) = 2^k\, T(\frac{n}{2^k}) + k \times n$$

$$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow k = \log(n)$$
$$\Rightarrow \quad T(n) = 2^k\, T(1) + k \times n$$
$$\Rightarrow \quad T(n) = n \times 1 + n\log(n)$$

$$\left.\begin{aligned} f(n) &= O(n\log(n)) \\ f(n) &= \Omega(n\log(n)) \end{aligned}\right\} \Rightarrow \quad f(n) = \theta(n\log(n))$$

17 Summary

$T(n) = 2T(\frac{n}{2}) + 1$	$O(n)$
$T(n) = 4T(\frac{n}{2}) + 1$	$O(n^2)$
$T(n) = 4T(\frac{n}{2}) + 1$	$O(n^2)$
$T(n) = 8T(\frac{n}{2}) + n^2$	$O(n^3)$
$T(n) = 16T(\frac{n}{2}) + n^2$	$O(n^4)$

$T(n) = T(\frac{n}{2}) + 1$	$O(\log(n))$
$T(n) = 2T(\frac{n}{2}) + n$	$O(n\log(n))$
$T(n) = 4T(\frac{n}{2}) + n^2$	$O(n^2\log(n))$

18 Binary Search

l ↓							mid ↓							h ↓
3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

key $= 42$

l	h	mid $= \left[\frac{l+h}{2} \right]$
1	15	$\frac{1+15}{2} = 8$

								l ↓			mid ↓			h ↓
3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

l	h	mid $= \left[\frac{l+h}{2} \right]$
8+1 $= 9$	15	$\frac{9+15}{2} = \frac{24}{2} = 12$

								l ↓	mid ↓	h ↓				
3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

l	h	$\text{mid} = \left[\dfrac{l+h}{2}\right]$
9	12-1 = 11	$\dfrac{9+11}{2} = \dfrac{20}{2} = 10$

										mid ↓				
										l h ↓ ↓				
3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

l	h	$\text{mid} = \left[\dfrac{l+h}{2}\right]$
10+1 = 11	11	$\dfrac{11+11}{2} = 11$

19 Binary Search Iterative Method

```
int BinarySearch(A,n,key) {
   l = 1;
   h = n;
   while(l <= h) {
      mid = (l+h)/2;
      if(key == A[mid]) {
         return mid;
      }
      if(key < A[mid]) {
         h = mid - 1;
      } else {
         l = mid + 1;
      }
   }
   return -1;
}
```

20 Binary Search Recursive Method

```
int BinarySearch(l,h,key) {
   if( l == h ) {
      if(A[l] == key) {
         return l;
      } else {
         return -1;
      }
   } else {
      mid = (l+h)/2;
      if( key == A[mid] ) {
         return mid;
      } else if( key < A[mid] ) {
         return BinarySearch(l,mid-1,key);
      } else if( key > A[mid] ) {
         return BinarySearch(mid+1,h,key)
      }
   }
}
```

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

$$\Rightarrow f(n) = O(\log(n))$$