Data Structures
COMP242

Ala' Hasheesh
ahashesh@birzeit.edu

Algorithm Analysis

BIRZEIT UNIVERSITY

Review

To select best Algorithm, we analyze two factors:

- **Time:** Function describing the amount of time it takes the given algorithm to give us the output

- **Space/Memory:** Function describing the total amount of memory needed to run our algorithm

Review

In this course we will focus on time complexity!

Review

Running time depends on:

- Algorithm design (Linear vs Logarithmic)
- Input size (n = 10 vs n = 10^6)
- Programming language (c vs JAVA)
- Compiler
- OS (windows vs Linux)
- Computer Hardware (CPU, RAM, etc…)

Review

- Asymptotic Notation

Asymptotic Notation is a formal notation for discussing and analyzing "classes of functions"

- "Big-O" notation : $O(N)$
- "Big-Omega of n": $\Omega(N)$
- "Theta of n": $\Theta(N)$

Review

"Big-O" notation : $\mathbf{O}(\boldsymbol{N})$

- $T(N) = \mathrm{O}(f(N))$ if there are positive constants c and n_0 such that $\boldsymbol{T}(\boldsymbol{N}) \leq \boldsymbol{cf}(\boldsymbol{N})$ **when** $\boldsymbol{N} \geq \boldsymbol{n_0}$

"Upper Bound"

"Big-Omega of n": $\boldsymbol{\Omega}(\boldsymbol{N})$

- $T(N) = \Omega(g(N))$ if there are positive constants c and n_0 such that $\boldsymbol{T}(\boldsymbol{N}) \geq \boldsymbol{cg}(\boldsymbol{N})$ **when** $\boldsymbol{N} \geq \boldsymbol{n_0}$

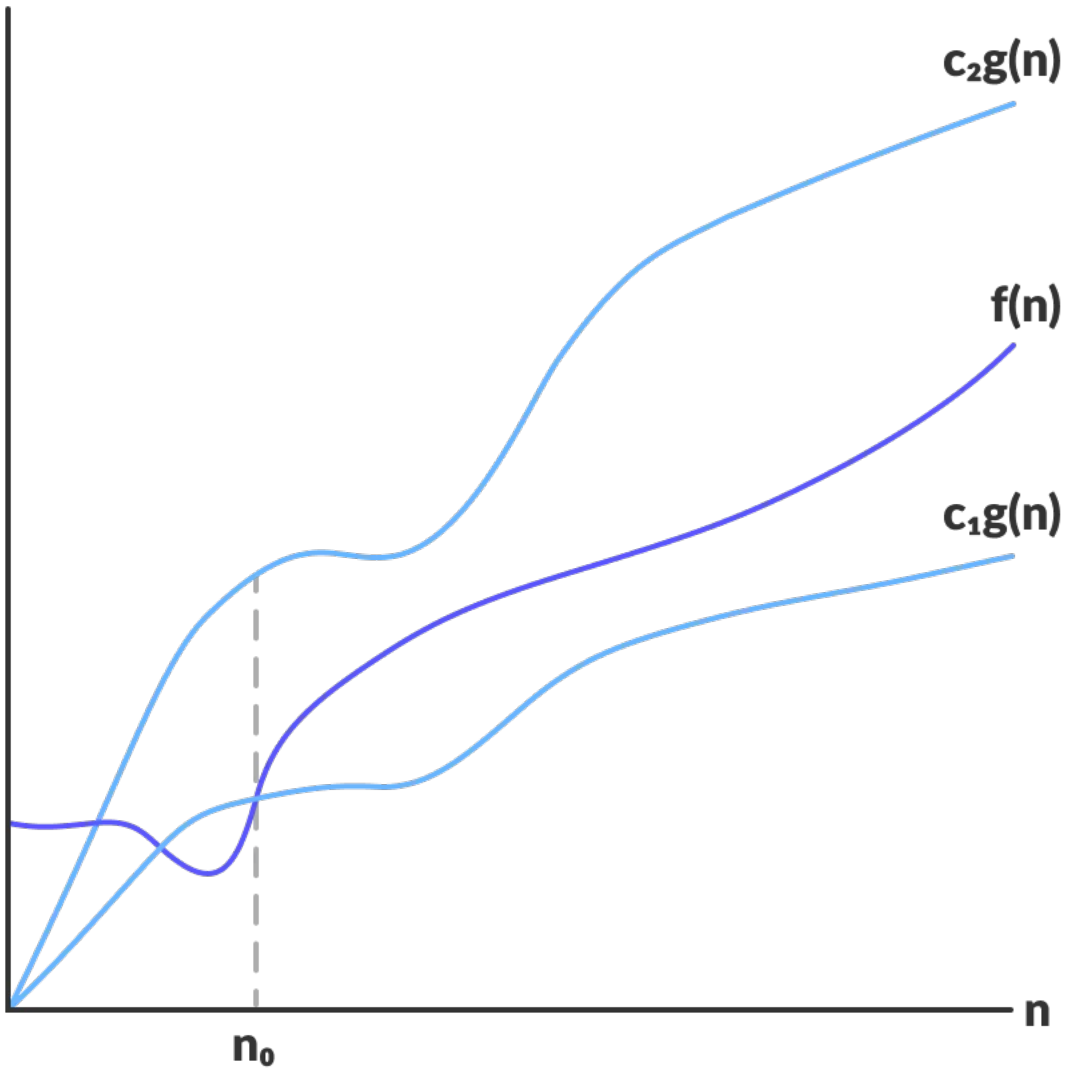
"Lower Bound"

"Theta of n": $\boldsymbol{\Theta}(\boldsymbol{N})$

- $T(N) = \Theta(h(N))$ if and only if $\boldsymbol{T}(\boldsymbol{N}) = \boldsymbol{O}(\boldsymbol{h}(\boldsymbol{N}))$ and $\boldsymbol{T}(\boldsymbol{N}) = \boldsymbol{\Omega}(\boldsymbol{h}(\boldsymbol{N}))$

"Tight Bound"

Review



Review

Big-O		Name
$O(1)$		Constant
$O(log\, n)$		Logarithmic
$O(\log^2 n)$		Log-squared time
$O(n)$		Linear
$O(nlog\, n)$		
$O(n^2)$		Quadratic
$O(n^3)$		Cubic
$O(n^i)$		Polynomial
$O(c^n)$	$c > 1$	Exponential

Time Complexity:
Computational complexity that measures or estimates the time taken for running an algorithm.

Complexity can be viewed as the maximum number of primitive operations that a program may execute.

Review

Function	N=10	N=100	N=1000	N=10^6
$O(n)$	10 ns	100 ns	1000 ns	1 ms
$O(n^2)$	100 ns	10000 ns	1 ms	17 min
$O(nlog\ n)$	35 ns	700 ns	10000 ns	20 ms
$O(2^n)$	1000 ns	4 x 10^{14} years!	Too long!	Too long!
$O(n!)$	4 ms	Too long!	Too long!	Too long!

Review

Rules

Assume $\boldsymbol{T_1(n) = O(f(n))}$ and $\boldsymbol{T_2(n) = O(g(n))}$ then:

1. $T_1(n) + T_2(n) = \max(O(f(n), O(g(n)))$

2. $T_1(n) * T_2(n) = O(f(n) * g(n))$

Review

```
private int add(int x, int y) {
    return x + y;   // 1 operation
}
```

$$T(n) = c$$

$$= \mathbf{O(1)}$$

```
private int add(int n) {
    int sum = 0;    // 1 operation
    for (int i = 0; i < n; i++) {   // 2n
        sum = sum + i; // n
    }

    return sum; // 1
}
```

$$T(n) = \mathbf{3n + c}$$

Here **d** and **c** are constants

$$= \mathbf{O(n)}$$

Review

Rule1

- $T(n) = 1 + 3n + 5n^2$

Review

Rule1

- $T(n) = 1 + 3n + 5n^2$

$$\mathbf{O}(\mathbf{n}^2)$$

Review

Rule1

- $T(n) = 1 + 3n + 5n^2$

- $1 + 3n + 5n^2 \leq n^2 + 3n^2 + 5n^2$
- $1 + 3n + 5n^2 \leq 9n^2$

Review

Rule1

- $T(n) = 1 + 3n + 5n^2$

- $1 + 3n + 5n^2 \leq n^2 + 3n^2 + 5n^2$
- $1 + 3n + 5n^2 \leq 9n^2$

$$\mathbf{O(n^2)}$$

Review

Better

Worse

Big-O	
$O(1)$	
$O(log\, n)$	
$O(\log^2 n)$	
$O(n)$	
$O(nlog\, n)$	
$O(n^2)$	
$O(n^3)$	
$O(n^i)$	
$O(c^n)$	$c > 1$

For Example:

$O(log\, n)$ is better than $O(n)$

Review & Examples

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

    for (int j = 0; j < n * n; j++) {
        sum += 1;
    }

}
```

Review & Examples

```
for (int i = 0; i < n; i++) {                    // n

   for (int j = 0; j < n*n; j++) {               // n²
      sum += 1;
   }

}
```

Review & Examples

```
for (int i = 0; i < n; i++) {                   // n

    for (int j = 0; j < n*n; j++) {             // n²
        sum += 1;
    }

}

for (int i = 0; i < n; i++) {                   // n
    sum += 1;
}
```

$O(n^3)$

$O(n^3)$

$O(n)$

Review & Examples

```
for (int i = 0; i < n; i++) {      // n
    for (int j = n; j < n*n; j++) {    // n^2 - n
        sum += 1;
    }
}
```

```
for (int i = 0; i < n*n; i++) {   // n^2
    for (int j = i*i; j > 0; j--) {   // n^4
        for (int k = j; k < j * j; k++) {   // n^8 - n^4
            sum++;
        }
    }
}
```

Review & Examples

```
for (int i = 0; i < n; i++) {      // n
    for (int j = n; j < n*n; j++) {    // ≈ n^2
        sum += 1;
    }
}
```

$$T(n) = O(n^3)$$

```
for (int i = 0; i < n*n; i++) {   // n^2
    for (int j = i*i; j > 0; j--) {    // ≈ n^4
        for (int k = j; k < j * j; k++) {   // ≈ n^8
            sum++;
        }
    }
}
```

$$T(n) = O(n^{14})$$

If statement & switch

```
if (sum > 10) {
   // O(n)
} else {
   // O(n^2)
}
```

We usually analyze worse case running time!

$O(n^2)$

If statement & switch

```
switch (sum % 2) {
  case 1:
    // O(n)
    break;
  case 2:
    // O(n^10)
    break;
  default:
    // O(n^3)
}
```

$O(n^3)$

Review & Examples

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

    for (int j = n - 1; j < n; j++) {
        sum += 1;
    }

}
```

Review & Examples

```
for (int i = 0; i < n; i++) {                    // n

    for (int j = n - 1; j < n; j++) {            // 1
        sum += 1;
    }

}
```

Review & Examples

```
for (int i = 0; i < n; i++) {                    // n

    for (int j = n - 1; j < n; j++) {            // 1
        sum += 1;
    }

}
```

O(n)

Recursion (Factorial)

```
int fact(int n) {
   if (n == 0) {
      return 1;
   }

   return n * fact(n - 1);
}
```

Recursion

```
int fact(int n) {
   if (n == 0) {
      return 1;
   }

   return n * fact(n - 1);
}
```

$$fact(n) = \begin{cases} 1, & n = 0 \\ n * fact(n-1), & n > 0 \end{cases}$$

Recursion

```
int fact(int n) {
    if (n == 0) {
        return 1;
    }

    return n * fact(n - 1);
}
```

$$fact(n) = \begin{cases} 1, & n = 0 \\ n * fact(n-1), & n > 0 \end{cases}$$

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

Recursion

```
int fact(int n) {
    if (n == 0) {
        return 1;
    }

    return n * fact(n - 1);
}
```

$$fact(n) = \begin{cases} 1, & n = 0 \\ n * fact(n-1), & n > 0 \end{cases}$$

Multiplication is just one operation that takes constant time **c**

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

Recursion (Substitution Method)

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

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

Recursion (Substitution Method)

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

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

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

Recursion (Substitution Method)

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

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

$T(n) = \boldsymbol{T(n-1)} + c$

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

Recursion (Substitution Method)

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

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

$T(n) = \boldsymbol{T(n-1)} + c$

$T(n) = [\boldsymbol{T(n-2) + c}] + c$

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

Recursion (Substitution Method)

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

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

$T(n) = \boldsymbol{T(n-1)} + c$

$T(n) = [\boldsymbol{T(n-2) + c}] + c$

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

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

Recursion (Substitution Method)

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

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

Recursion (Substitution Method)

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

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

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

Recursion (Substitution Method)

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

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

$$T(n) = \boldsymbol{T(n-2)} + c + c$$

$$T(n) = [\boldsymbol{T(n-3) + c}] + c + c$$

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

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

Recursion (Substitution Method)

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

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

Recursion (Substitution Method)

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

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

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

Recursion (Substitution Method)

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

When do we stop???

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

$$T(n) = T(n-k) + kc$$

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

Recursion (Substitution Method)

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

When do we stop???

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

When $n = 0$ or when we reach $T(0)$

$T(n) = T(n-k) + kc$

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

Set $\boldsymbol{k = n}$

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

Set $\boldsymbol{k = n}$

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

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

Set $\boldsymbol{k = n}$

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

$T(n) = T(0) + nc$

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

Set $\boldsymbol{k = n}$

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

$T(n) = T(0) + nc$

$T(n) = d + nc$

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

Recursion (Substitution Method)

$T(n) = T(n-k) + kc$

We should stop when we reach $\boldsymbol{T(0)}$

Set $\boldsymbol{k = n}$

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

$T(n) = T(0) + nc$

$T(n) = d + nc$

$\mathbf{O(n)}$

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

Recursion (To Binary)

```
String toBinary(int n) {
    if (n <= 1) {
        return n + "";
    }

    return toBinary(n / 2) + (n % 2);
}
```

Recursion (To Binary)

```
String toBinary(int n) {
    if (n <= 1) {
        return n + "";
    }

    return toBinary(n / 2) + (n % 2);
}
```

$$f(n) = \begin{cases} n + \text{""}, & n \leq 1 \\ f\left(\frac{n}{2}\right) + (n\%2), & n > 1 \end{cases}$$

Recursion (To Binary)

```
String toBinary(int n) {
    if (n <= 1) {
        return n + "";
    }

    return toBinary(n / 2) + (n % 2);
}
```

$$f(n) = \begin{cases} n + \text{""}, & n \le 1 \\ f\left(\frac{n}{2}\right) + (n\%2), & n > 1 \end{cases}$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \le 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

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

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

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

$$T(n) = T\left(n \ * \frac{1}{2}\right) + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

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

$$T(n) = T\left(n\ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2}\ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2 * 2}\right) + c$$

$$\mathrm{T}(n) = \begin{cases} d,\ n \leq 1 \\ T\left(\frac{n}{2}\right) + c,\ n > 1 \end{cases}$$

Recursion (To Binary)

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

$$T(n) = T\left(n \ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2} \ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2 * 2}\right) + c$$

$$T(n) = \boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right) + c$$

$$T(n) = \left[\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{2} * \mathbf{2}}\right) + \boldsymbol{c}\right] + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

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

$$T(n) = T\left(n\ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2} * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{2*2}\right) + c$$

$$T(n) = \boldsymbol{T\left(\frac{n}{2}\right)} + c$$

$$T(n) = [\boldsymbol{T\left(\frac{n}{2*2}\right) + c}] + c$$

$$T(n) = \boldsymbol{T\left(\frac{n}{2*2}\right) + c} + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \le 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2*2}\right) + c + c$$

$$T(n) = T\left(\frac{n}{4}\right) + c + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2 * 2}\right) + c + c$$

$$T(n) = T\left(\frac{n}{4}\right) + c + c$$

$$T(n) = T\left(n \ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{4} \ * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{4 * 2}\right) + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2*2}\right) + c + c$$

$$T(n) = T\left(\frac{n}{4}\right) + c + c$$

$$T(n) = T\left(n \; * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{4} * \frac{1}{2}\right) + c$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{4*2}\right) + c$$

$$T(n) = \boldsymbol{T\left(\frac{n}{4}\right)} + c + c$$

$$T(n) = \boldsymbol{\left[T\left(\frac{n}{4*2}\right) + c\right]} + c + c$$

$$T(n) = \boldsymbol{T\left(\frac{n}{4*2}\right) + c} + c + c$$

$$\mathrm{T}(n) = \begin{cases} d, \; n \le 1 \\ T\left(\frac{n}{2}\right) + c, \; n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{4*2}\right) + c + c + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{4 * 2}\right) + c + c + c$$

$$T(n) = T\left(\frac{n}{2 * 2 * 2}\right) + c + c + c$$

$$\mathrm{T}(n) = \begin{cases} d, \ n \leq 1 \\ T\left(\frac{n}{2}\right) + c, \ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{4*2}\right) + c + c + c$$

$$T(n) = T\left(\frac{n}{2*2*2}\right) + c + c + c$$

$$T(n) = T\left(\frac{n}{2^3}\right) + 3c$$

$$\mathrm{T}(n) = \begin{cases} d,\ n \le 1 \\ T\left(\frac{n}{2}\right) + c,\ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{4*2}\right) + c + c + c$$

$$T(n) = T\left(\frac{n}{2*2*2}\right) + c + c + c$$

When do we stop???

$$T(n) = T\left(\frac{n}{2^3}\right) + 3c$$

When $n \leq 1$ or when we reach $T(1)$

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

$$\mathrm{T}(n) = \begin{cases} d, & n \leq 1 \\ T\left(\frac{n}{2}\right) + c, & n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

Recursion (To Binary)

$T(n) = T\left(\frac{n}{2^k}\right) + kc$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

Recursion (To Binary)

$T(n) = T\left(\frac{n}{2^k}\right) + kc$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

Recursion (To Binary)

$T(n) = T\left(\frac{n}{2^k}\right) + kc$

$$\boldsymbol{log_2 n = log_2 2^k}$$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

Recursion (To Binary)

$T(n) = T\left(\frac{n}{2^k}\right) + kc$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log_2 n = \log_2 2^k}$$

$$\boldsymbol{\log_2 n = k \log_2 2}$$

Rules

1. $\boldsymbol{\log_x a^b = b \log_x a}$

Recursion (To Binary)

$T(n) = T\left(\frac{n}{2^k}\right) + kc$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\boldsymbol{\frac{n}{2^k} = 1}$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log_2 n = \log_2 2^k}$$

$$\boldsymbol{\log_2 n = k \log_2 2}$$

$$\boldsymbol{\log_2 n = k}$$

Rules

1. $\boldsymbol{\log_x a^b = b \log_x a}$
2. $\boldsymbol{\log_x x = 1}$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

We want $T\left(\frac{n}{2^k}\right) = T(1)$

We want $\frac{\boldsymbol{n}}{\boldsymbol{2^k}} = \mathbf{1}$

$$\frac{\boldsymbol{n}}{\boldsymbol{2^k}} = \mathbf{1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log_2 n = \log_2 2^k}$$

$$\boldsymbol{\log_2 n = k \log_2 2}$$

$$\boldsymbol{\log_2 n = k}$$

When dealing with logs we usually use $\boldsymbol{log}$

$$\boldsymbol{\log\ n = k}$$

Rules

1. $\boldsymbol{\log_x a^b = b \log_x a}$
2. $\boldsymbol{\log_x x = 1}$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

$$\mathbf{log}\ \boldsymbol{n} = \boldsymbol{k}$$

$$\mathrm{T}(n) = \begin{cases} d,\ n \le 1 \\ T\left(\frac{n}{2}\right) + c,\ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{\boldsymbol{n}}{\boldsymbol{2^k}}\right) + \boldsymbol{k}c$$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log\ n = k}$$

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

$$\mathrm{T}(n) = \begin{cases} d,\ n \leq 1 \\ T\left(\frac{n}{2}\right) + c,\ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

$$\boldsymbol{\log\ n = k}$$

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

$$T(n) = d + c\log\ n$$

$$\mathrm{T}(n) = \begin{cases} d,\ n \le 1 \\ T\left(\frac{n}{2}\right) + c,\ n > 1 \end{cases}$$

Recursion (To Binary)

$$T(n) = T\left(\frac{n}{2^k}\right) + kc$$

$$\boldsymbol{\log n = k}$$

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

$$T(n) = d + c \log n$$

$$\mathbf{O}(\boldsymbol{\log n})$$

$$\mathrm{T}(n) = \begin{cases} d, & n \leq 1 \\ T\left(\frac{n}{2}\right) + c, & n > 1 \end{cases}$$

Recursion (Merge Sort)

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

$$\mathrm{T}(n) = \begin{cases} d,\ n = 1 \\ 2T\left(\frac{n}{2}\right) + n,\ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

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

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

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

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

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

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

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

$$T(n) = 2\left[\boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{2}} * \frac{\boldsymbol{1}}{\boldsymbol{2}}\right) + \frac{\boldsymbol{n}}{\boldsymbol{2}}\right] + n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

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

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

$$T(n) = 2\left[\boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{2}} * \frac{\boldsymbol{1}}{\boldsymbol{2}}\right) + \frac{\boldsymbol{n}}{\boldsymbol{2}}\right] + n$$

$$T(n) = \left[\boldsymbol{4T}\left(\frac{\boldsymbol{n}}{\boldsymbol{4}}\right) + \boldsymbol{n}\right] + n$$

$$\mathrm{T}(n) = \begin{cases} d, \; n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \; n > 1 \end{cases}$$

Recursion (Merge Sort)

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

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

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

$$T(n) = 2\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right) + n$$

$$T(n) = 2\left[\mathbf{2}\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{2}} * \frac{\mathbf{1}}{\mathbf{2}}\right) + \frac{\boldsymbol{n}}{\mathbf{2}}\right] + n$$

$$T(n) = \left[\mathbf{4}\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{4}}\right) + \boldsymbol{n}\right] + n$$

$$T(n) = 4T\left(\frac{n}{4}\right) + n + n$$

$$\mathrm{T}(n) = \begin{cases} d, & n = 1 \\ 2T\left(\frac{n}{2}\right) + n, & n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + n + n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

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

$$\mathrm{T}(n) = \begin{cases} d,\ n = 1 \\ 2T\left(\frac{n}{2}\right) + n,\ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4} * \frac{1}{2}\right) + \frac{n}{4}$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

$$T(n) = 4\boldsymbol{T\left(\frac{n}{4}\right)} + 2n$$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4} * \frac{1}{2}\right) + \frac{n}{4}$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4} * \frac{1}{2}\right) + \frac{n}{4}$$

$$T(n) = 4\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\boldsymbol{4}}\right) + 2n$$

$$T(n) = 4\left[\boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{8}}\right) + \frac{\boldsymbol{n}}{\boldsymbol{4}}\right] + 2n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4} * \frac{1}{2}\right) + \frac{n}{4}$$

$$T(n) = 4\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\boldsymbol{4}}\right) + 2n$$

$$T(n) = 4[\boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{8}}\right) + \frac{\boldsymbol{n}}{\boldsymbol{4}}] + 2n$$

$$T(n) = [\boldsymbol{8T}\left(\frac{\boldsymbol{n}}{\boldsymbol{8}}\right) + \boldsymbol{n}] + 2n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 4T\left(\frac{n}{4}\right) + 2n$$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4} * \frac{1}{2}\right) + \frac{n}{4}$$

$$T(n) = 4\boldsymbol{T}\left(\frac{\boldsymbol{n}}{\boldsymbol{4}}\right) + 2n$$

$$T(n) = 4\left[\boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{8}}\right) + \frac{\boldsymbol{n}}{\boldsymbol{4}}\right] + 2n$$

$$T(n) = \left[\boldsymbol{8T}\left(\frac{\boldsymbol{n}}{\boldsymbol{8}}\right) + \boldsymbol{n}\right] + 2n$$

$$T(n) = 8T\left(\frac{n}{8}\right) + n + 2n$$

$$\mathrm{T}(n) = \begin{cases} d, & n = 1 \\ 2T\left(\frac{n}{2}\right) + n, & n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 8T\left(\frac{n}{8}\right) + n + 2n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 8T\left(\frac{n}{8}\right) + n + 2n$$

$$T(n) = (2 * 2 * 2)T\left(\frac{n}{2 * 2 * 2}\right) + 3n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 8T\left(\frac{n}{8}\right) + n + 2n$$

$$T(n) = (2 * 2 * 2)T\left(\frac{n}{2 * 2 * 2}\right) + 3n$$

$$T(n) = 2^3T\left(\frac{n}{2^3}\right) + 3n$$

$$\mathrm{T}(n) = \begin{cases} d,\ n = 1 \\ 2T\left(\frac{n}{2}\right) + n,\ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 8T\left(\frac{n}{8}\right) + n + 2n$$

$$T(n) = (2 * 2 * 2)T\left(\frac{n}{2 * 2 * 2}\right) + 3n$$

$$T(n) = 2^3T\left(\frac{n}{2^3}\right) + 3n$$

$$T(n) = 2^kT\left(\frac{n}{2^k}\right) + kn$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

When do we stop?

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

When do we stop?

When $\boldsymbol{T\left(\frac{n}{2^k}\right) = T(1)}$

When $\boldsymbol{\frac{n}{2^k} = 1}$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

When do we stop?

When $\boldsymbol{T\left(\frac{n}{2^k}\right) = T(1)}$

When $\boldsymbol{\frac{n}{2^k} = 1}$

When $\boldsymbol{\log_2 n = k}$ **(From previous Example)**

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\boldsymbol{\frac{n}{2^k} = 1}$$
$$\boldsymbol{n = 2^k}$$
$$\boldsymbol{\log\ n = k}$$

$$\mathrm{T}(n) = \begin{cases} d,\ n = 1 \\ 2T\left(\frac{n}{2}\right) + n,\ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\frac{\boldsymbol{n}}{\boldsymbol{2^k}} = \boldsymbol{1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log\ n = k}$$

$$T(n) = \mathbf{2^k} T\left(\frac{\boldsymbol{n}}{\boldsymbol{2^k}}\right) + \boldsymbol{k}n$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log\ n = k}$$

$$T(n) = \mathbf{2^k} T\left(\boldsymbol{\frac{n}{2^k}}\right) + \boldsymbol{k}n$$

$$T(n) = \boldsymbol{n}T(\mathbf{1}) + n\log n$$

$$\mathrm{T}(n) = \begin{cases} d, \ \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log\ n = k}$$

$$T(n) = \mathbf{2^k} T\left(\boldsymbol{\frac{n}{2^k}}\right) + \boldsymbol{k}n$$

$$T(n) = \boldsymbol{n} T(\mathbf{1}) + n\log n$$

$$T(n) = dn + nlog\, n$$

$$\mathrm{T}(n) = \begin{cases} d, \ \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ \ n > 1 \end{cases}$$

Recursion (Merge Sort)

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\boldsymbol{\frac{n}{2^k} = 1}$$

$$\boldsymbol{n = 2^k}$$

$$\boldsymbol{\log\ n = k}$$

$$T(n) = \mathbf{2}^{\boldsymbol{k}} T\left(\frac{\boldsymbol{n}}{\mathbf{2}^{\boldsymbol{k}}}\right) + \boldsymbol{k}n$$

$$T(n) = \boldsymbol{n}T(\mathbf{1}) + n\log n$$

$$T(n) = dn + nlog\, n$$

$$\boldsymbol{O(nlog\ n)}$$

$$\mathrm{T}(n) = \begin{cases} d, \ n = 1 \\ 2T\left(\frac{n}{2}\right) + n, \ n > 1 \end{cases}$$