# Complexity Analysis of Recursive Algorithms

Solving Recurrences

Analyze Merge Sort by Recurrence

# Remember: Merge Sort

```
MergeSort(A, left, right) {
   if (left < right) {
       mid = floor((left + right) / 2);
       MergeSort(A, left, mid);
       MergeSort(A, mid+1, right);
       Merge(A, left, mid, right);
   }
}

// Merge() takes two sorted subarrays of A and
// merges them into a single sorted subarray of A
//     (how long should this take?)
```

# Analysis of Merge Sort

| Statement | Effort |
|---|---|
| `MergeSort(A, left, right) {` | T(n) |
| `if (left < right) {` | O(1) |
| `mid = floor((left + right) / 2);` | O(1) |
| `MergeSort(A, left, mid);` | T(n/2) |
| `MergeSort(A, mid+1, right);` | T(n/2) |
| `Merge(A, left, mid, right);` | O(n) |
| `}` | |
| `}` | |

- So $T(n) =$ $O(1)$ when $n = 1$, and
  $2T(n/2) + O(n)$ when $n > 1$
- So, what is $T(n)$? We shall solve this by **Recurrence**.

# Recurrences

- The expression:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + cn & n > 1 \end{cases}$$

is a *recurrence*.

  - **Recurrence:** an equation that describes a function by same but smaller functions

# Recurrence Examples

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

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

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

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

# Solving Recurrences

- There are three methods:
  1. Substitution method
  2. Iteration method
  3. Master method

- We shall see 2 and the 3.

# Solving Recurrences: Iterative Method

- Expand the recurrence
- Work some algebra to express as a summation
- Evaluate the summation

- We will show several examples

Example 1: $$s(n)=\begin{cases}0 & n=0\\ c+s(n-1) & n>0\end{cases}$$

- s(n) =

  c + s(n-1)

  c + c + s(n-2)

  2c + s(n-2)

  2c + c + s(n-3)

  3c + s(n-3)

  …

  kc + s(n-k) = ck + s(n-k)

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

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn

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

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn
- So

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

- Thus in general
  - s(n) = cn = O(n)

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

- s(n)

= n + s(n-1)

= n + n-1 + s(n-2)

= n + n-1 + n-2 + s(n-3)

= n + n-1 + n-2 + n-3 + s(n-4)

= …

= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)

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

- s(n)

= n + s(n-1)

= n + n-1 + s(n-2)

= n + n-1 + n-2 + s(n-3)

= n + n-1 + n-2 + n-3 + s(n-4)

= …

= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)

$$= \sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

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

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

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

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

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

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

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

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

- Thus in general

$$s(n) \quad = \quad n\frac{n+1}{2} = O(n^2)$$

Example 3:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + c & n > 1 \end{cases}$$

- T(n) =

  2T(n/2) + c

  2(2T(n/2/2) + c) + c

  $2^2T(n/2^2) + 2c + c$

  $2^2(2T(n/2^2/2) + c) + 3c$

  $2^3T(n/2^3) + 4c + 3c$

  $2^3T(n/2^3) + 7c$

  $2^3(2T(n/2^3/2) + c) + 7c$

  $2^4T(n/2^4) + 15c$

  …

  $2^kT(n/2^k) + (2^k - 1)c$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + c & n > 1 \end{cases}$$

- So far for $n > 2^k$ we have
  - $T(n) = 2^k T(n/2^k) + (2^k - 1)c$
- What if $k = \lg n$? (highest value of $n = 2^{\lg n}$)
  - $T(n) = 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c$

    $= n\, T(n/n) + (n - 1)c$

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

    $= nc + (n-1)c = (2n - 1)c = O(n)$

## Example 4: $T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$

- $T(n) =$

  $aT(n/b) + cn$

  $a(aT(n/b/b) + cn/b) + cn$

  $a^2T(n/b^2) + cna/b + cn$

  $a^2T(n/b^2) + cn(a/b + 1)$

  $a^2(aT(n/b^2/b) + cn/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2) + cn(a/b + 1)$

  $a^3T(n/b^3) + cn(a^2/b^2 + a/b + 1)$

  …

  $a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + \ldots + a^2/b^2 + a/b + 1)$

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$$

- So we have
  - $T(n) = a^k T(n/b^k) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
- For $k = \log_b n$
  - $n = b^k$ (because, the highest value of $n = b^k$)
  - $T(n) = a^k T(1) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= a^k c + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= ca^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cna^k /b^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$

    $= cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a = b$?
  - $T(n) = cn(k + 1)$

    $= cn(\log_b n + 1)$

    $= O(n \log n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\frac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with k = $\log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if a < b?
  - Recall that $(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 = \frac{(a/b)^{k+1}-1}{(a/b)-1} = \frac{1-(a/b)^{k+1}}{1-(a/b)} < \frac{1}{1-a/b}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

  - $T(n) = cn \cdot O(1) = O(n)$

Constant

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\frac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + ... + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if a > b?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad O\left((a/b)^k\right)$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 \quad = \quad \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \quad = \quad O\left((a/b)^k\right)$$

  - $T(n) = cn \cdot O(a^k / b^k)$

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 \quad = \quad \frac{(a/b)^{k+1}-1}{(a/b)-1} \quad = \quad O\left((a/b)^k\right)$$

  - $T(n) = cn \cdot O(a^k / b^k)$

$$= cn \cdot O(a^{\log_b n} / b^{\log_b n}) = cn \cdot O(a^{\log_b n} / n)$$

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 \quad = \quad \frac{(a/b)^{k+1}-1}{(a/b)-1} \quad = \quad O\left((a/b)^k\right)$$

  - $T(n) = cn \cdot O(a^k / b^k)$

$$= cn \cdot O(a^{\log n} / b^{\log n}) = cn \cdot O(a^{\log n} / n)$$

*recall:* $a^{\log n} = n^{\log a}$ ( how? Take $\log_b$ in both sides)

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 \quad = \quad \frac{(a/b)^{k+1}-1}{(a/b)-1} \quad = \quad O\left((a/b)^k\right)$$

  - $T(n) = cn \cdot O(a^k / b^k)$

$= cn \cdot O(a^{\log n} / b^{\log n}) = cn \cdot O(a^{\log n} / n)$

*recall:* $a^{\log n} = n^{\log a}$ ( how? Take $\log_b$ in both sides)

$= cn \cdot O(n^{\log a} / n) = O(cn \cdot n^{\log a} / n)$

$$T(n)=\begin{cases} c & n=1 \\ aT\left(\dfrac{n}{b}\right)+cn & n>1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k}+\frac{a^{k-1}}{b^{k-1}}+\cdots+\frac{a}{b}+1 \quad = \quad \frac{(a/b)^{k+1}-1}{(a/b)-1} \quad = \quad O\left((a/b)^k\right)$$

  - $T(n) = cn \cdot O(a^k / b^k)$

$= cn \cdot O(a^{\log n} / b^{\log n}) = cn \cdot O(a^{\log n} / n)$

*recall:* $a^{\log n} = n^{\log a}$ ( how? Take $\log_b$ in both sides)

$= cn \cdot O(n^{\log a} / n) = O(cn \cdot n^{\log a} / n)$

$= O(n^{\log a})$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\dfrac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So…

$$T(n) = \begin{cases} O(n) & a < b \\ O(n \log_b n) & a = b \\ O\left(n^{\log_b a}\right) & a > b \end{cases}$$

# Analyze Merge Sort by Recurrence

Now we know that,

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases} = T(n) = \begin{cases} O(n) & a < b \\ O(n \log_b n) & a = b \\ O\left(n^{\log_b a}\right) & a > b \end{cases}$$

- So, what for Merge Sort?

  T(n) = O(1) when n = 1, and

  = 2T(n/2) + O(n) when n > 1
- a = b = 2, O(1) = c, and O(n) = cn.
- So, we get T(n) = O(n $\log_2$ n)
- So, MergeSort is O(n $\log_2$ n).

# The Master Theorem

- In general, a recurrence is like this: T(n) = aT(n/b) + f(n)
- The solution my Master Method is:

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & if \quad f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & if \quad f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & if \quad f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & \quad if\,(n/b) < cf(n) \text{ for large } n \end{cases} \left.\vphantom{\begin{matrix}1\\1\\1\\1\end{matrix}}\right\} \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

# Example: Master Method

- $T(n) = 9T(n/3) + n$
  - $a=9$, $b=3$, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Since $f(n) = n = O(n) = O(n^{\log_3 9 - \varepsilon}) = O(n^{\log_3 3^2 - \varepsilon}) = O(n^{2\log_3 3 - \varepsilon}) = O(n^{2*1 - \varepsilon}) = O(n^{2 - 1}) = O(n)$, where $\varepsilon=1$
  - So, case 1 of Master Method:
    $$\boldsymbol{T}(\boldsymbol{n}) = \Theta\left(\boldsymbol{n}^{\log_{\boldsymbol{b}} \boldsymbol{a}}\right) \text{when } \boldsymbol{f}(\boldsymbol{n}) = \boldsymbol{O}\left(\boldsymbol{n}^{\log_{\boldsymbol{b}} \boldsymbol{a} - \varepsilon}\right)$$
  - So solution is $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 9}) = \Theta(n^2)$

# Analysis of Merge Sort by Master Method

## Remember: Merge Sort:

| Statement | Effort |
|---|---|
| `MergeSort(A, left, right) {` | `T(n)` |
| `   if (left < right) {` | `O(1)` |
| `      mid = floor((left + right) / 2);` | `O(1)` |
| `      MergeSort(A, left, mid);` | `T(n/2)` |
| `      MergeSort(A, mid+1, right);` | `T(n/2)` |
| `      Merge(A, left, mid, right);` | `O(n)` |
| `   }` | |
| `}` | |

- These are actually Θ. Because, for example, you must do cn work here. So, it is <=cn and >=cn. For <=cn, we get O(n). For >=cn, we get Ω(n). Together we get: Θ(n).

# Analysis of Merge Sort by Master Method

## Remember: Merge Sort:

| Statement | Effort |
|---|---|
| `MergeSort(A, left, right) {` | `T(n)` |
| `    if (left < right) {` | `Θ(1)` |
| `        mid = floor((left + right) / 2);` | `Θ(1)` |
| `        MergeSort(A, left, mid);` | `T(n/2)` |
| `        MergeSort(A, mid+1, right);` | `T(n/2)` |
| `        Merge(A, left, mid, right);` | `Θ(n)` |
| `    }` | |
| `}` | |

- These are actually Θ. Because, for example, you must do cn work here. So, it is <=cn and >=cn. For <=cn, we get O(n). For >=cn, we get Ω(n). Together we get: Θ(n).

# Example: Analyze Merge Sort by Master Method

- So, the recurrence for Merge Sort is:

  $T(n) = \Theta(1)$ when $n = 1$, and

  $= 2T(n/2) + \Theta(n)$ when $n > 1$
- $a = b = 2$, $f(n) = \Theta(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n)$
- So, case 2 of Master Method
- So, $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^{\log_2 2} \log n) = \Theta(n \log n)$.