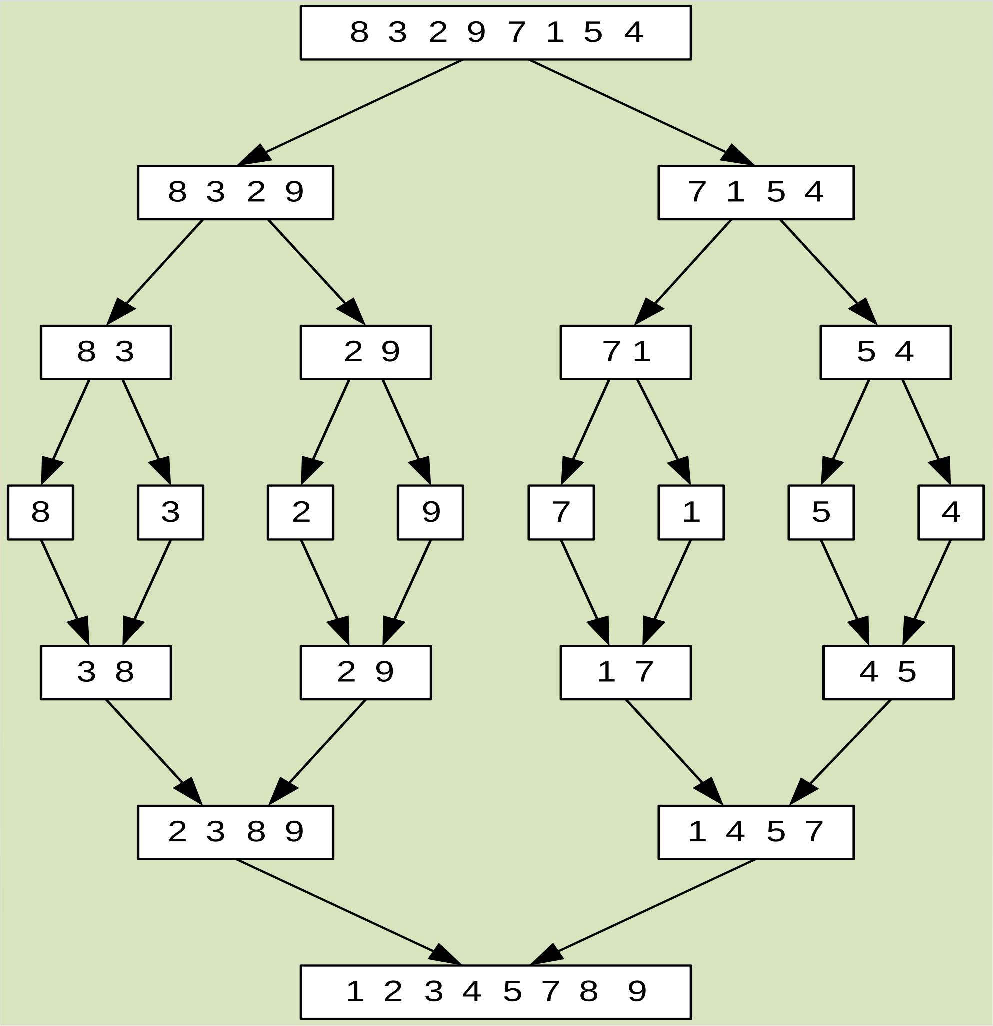
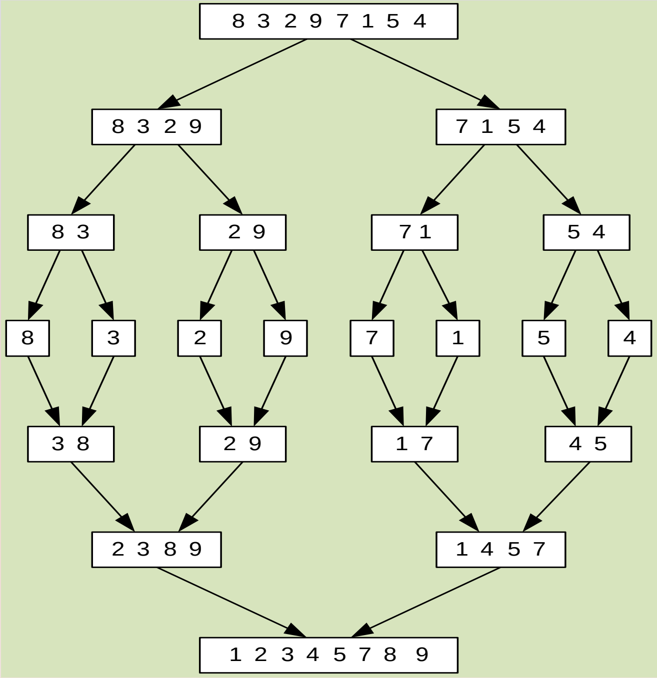
# Divide-and-Conquer

# Divide-and-Conquer Mergesort




A. Levitin "Introduction to the Design & Analysis of Algorithms," 2nd ed., Ch. 4

# Divide-and-Conquer Mergesort




$$T(n) = 2T(n/2) + \theta(n),$$
$$T(1) = \theta(1)$$

# Divide-and-Conquer
# Mergesort

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2[2T(n/4) + n/2] + n \\
&= 4T(n/4) + 2n.
\end{aligned}
$$

We can repeat this substitution again, and again, up to $\log n$ times:

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 4T(n/4) + 2n \\
&= 8T(n/8) + 3n \\
&\dots \\
&= 2^j T(n/2^j) + jn \\
&\dots \\
&= nT(1) + n \log n \\
&= \Theta(n \log n).
\end{aligned}
$$

# Example

**Find the maximum and minimum of a sequence**

If $n=1$, the number is itself min or max

If $n>1$, divide the numbers into two lists.

Decide the min & max in the first list.

Decide the min & max in the second list.

Choose the min & max of the entire list.

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

$\boldsymbol{T(n) = \Theta(n)}$ *(proof - later)*

Can you give another algorithm?

Let's solve the following recurrence in general:

$$T(n) = aT(n/b) + n$$

*where a > 0, b > 1, T(1) = 1*

do repeated substitutions:

$$\begin{aligned}
T(n) &= aT(n/b) + n \\
&= a[aT(n/b^2) + n/b] + n \\
&= a^2T(n/b^2) + (a/b)n + n \\
&\dots \\
&= a^jT(n/b^j) + n[(a/b)^{j-1} + \dots + (a/b)^2 + (a/b) + 1] \\
&\dots \\
&= a^{\log_b n}T(1) + n \cdot \sum_{i=0}^{\log_b n - 1} (a/b)^i \\
&= n^{\log_b a} + n \cdot \sum_{i=0}^{\log_b n - 1} (a/b)^i
\end{aligned}$$

check

$$n^{\log_b a} + n \cdot \sum_{i=0}^{\log_b n - 1} (a/b)^i$$

**Case 1:** $a = b$. The first term is $n$. In the summation, we have $\log_b n$ terms and they are all equal $a/b = 1$, so the second term is $n \log_b n$. Thus we get $T(n) = \Theta(n \log n)$.

**Case 2:** $a < b$. The second term is now a geometric series with the ratio smaller than 1, so $\sum_{i=0}^{\log_b n - 1} (a/b)^i = \Theta(1)$. The first term is $n^{\log_b a}$ with $\log_b a < 1$, so we get $T(n) = \Theta(n)$.

**Case 3:** $a > b$. Summing the geometric series in the second term, we get

$$\sum_{i=0}^{\log_b n - 1} (a/b)^i = \frac{(a/b)^{\log_b n} - 1}{(a/b) - 1} \overset{\text{check}}{=} \frac{b}{a-b}(a^{\log_b n}/b^{\log_b n} - 1) = \frac{b}{a-b}(n^{\log_b a}/n - 1)$$

So

$$T(n) = n^{\log_b a} + \frac{b}{a-b}(n^{\log_b a} - n) = \Theta(n^{\log_b a}).$$

For $r \neq 1$, the sum of the first $n$ terms of a geometric series is

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a\left(\frac{1-r^n}{1-r}\right)$$

# Divide-and-Conquer Master Theorem

Theorem **(Master Theorem)**

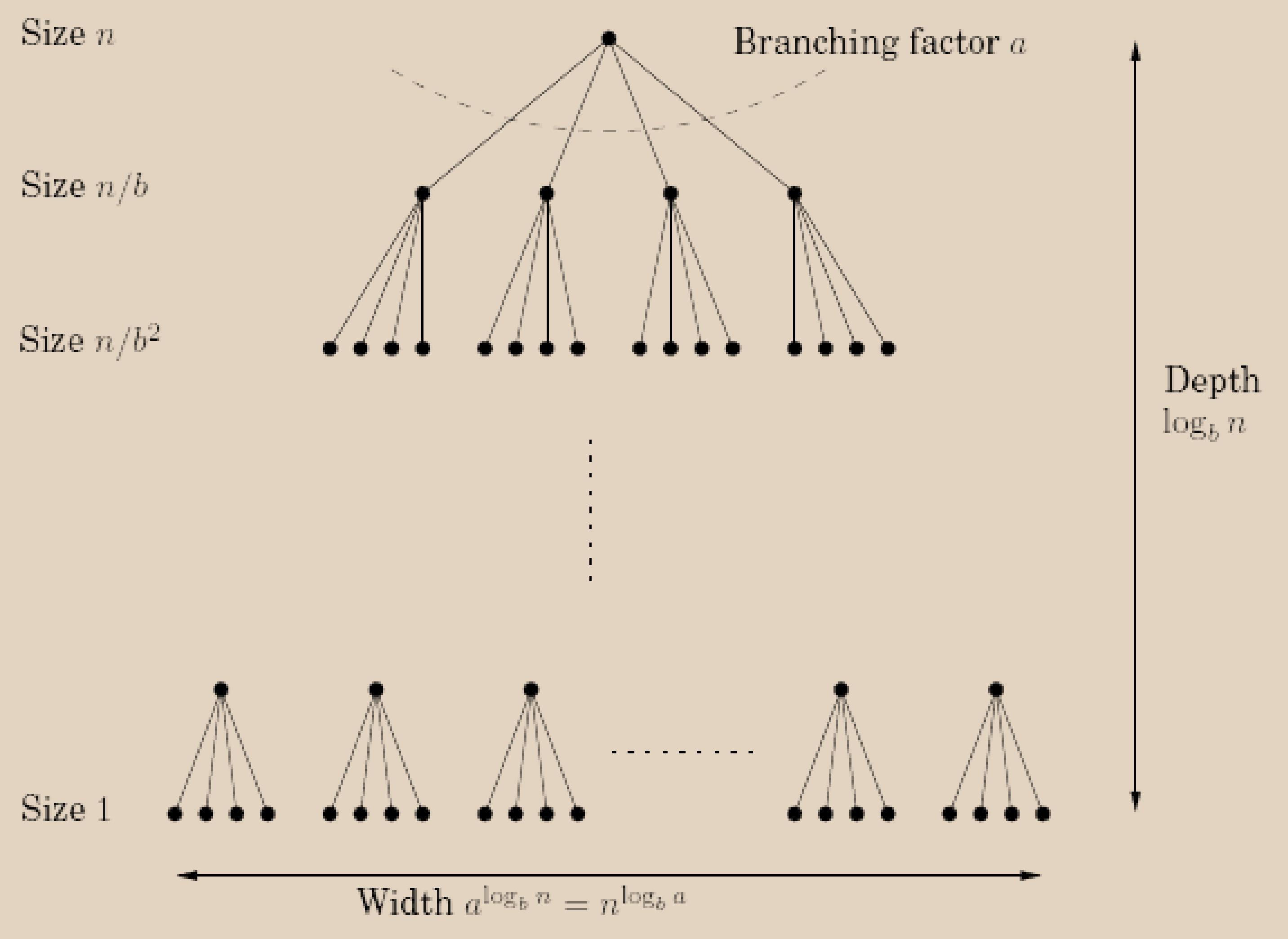
*Let* $a \geq 1$, $b > 1$, $c > 0$ *and* $d \geq 0$. *If* $T(n)$ *satisfies the recurrence*

$$T(n) \quad = \quad aT(n/b) + cn^d,$$

then

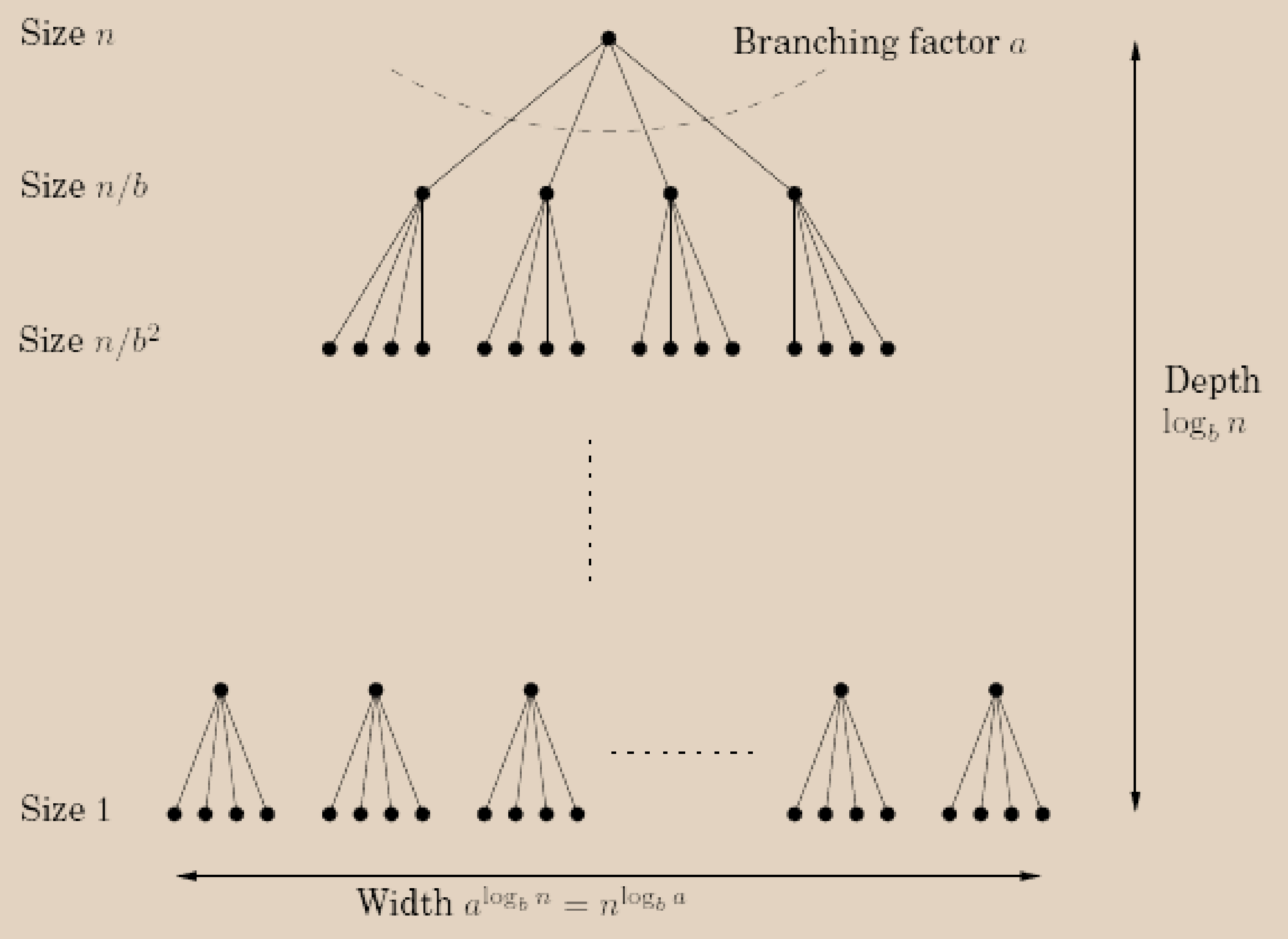
$$T(n) \quad = \quad \begin{cases} \Theta(n^{\log_b a}) & \text{for } a > b^d \\ \Theta(n^d \log n) & \text{for } a = b^d \\ \Theta(n^d) & \text{for } a < b^d \end{cases}$$

# Divide-and-Conquer
# Master Theorem






Algorithms, S. Dasgupta




$$T(n) = aT(n/b) + cn^d$$

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{for } a > b^d \\ \Theta(n^d \log n) & \text{for } a = b^d \\ \Theta(n^d) & \text{for } a < b^d \end{cases}$$

# Divide-and-Conquer

(a) **Algorithm** PrintXs ($n$ : **integer**)

    **if** $n < 3$

        print("X")

    **else**

        PrintXs($\lceil n/3 \rceil$)

        PrintXs($\lceil n/3 \rceil$)

        PrintXs($\lceil n/3 \rceil$)

        **for** $i \leftarrow 1$ **to** $2n$ **do** print("X")

# Divide-and-Conquer

(a) **Algorithm** PRINTXS ($n$ : **integer**)

  **if** $n < 3$

    print("X")

  **else**

    PRINTXS($\lceil n/3 \rceil$)

    PRINTXS($\lceil n/3 \rceil$)

    PRINTXS($\lceil n/3 \rceil$)

    **for** $i \leftarrow 1$ **to** $2n$ **do** print("X")

There are 3 recursive calls, each with parameter $\lceil n/3 \rceil$. Since we are looking for an asymptotic solution, we can ignore rounding. Then the number of letters printed can be expressed by the recurrence:

$$X(n) = 3X(n/3) + 2n.$$

We apply the Master Theorem with $a = 3$, $b = 3$, $c = 2$, $d = 1$. Here, we have $a = b^d$, so the solution is $\Theta(n \log n)$.

# Divide-and-Conquer

(b) **Algorithm** $\textsc{PrintYs}$ ($n$ : **integer**)

**if** $n < 2$

print("Y")

**else**

**for** $j \leftarrow 1$ **to** 16 **do** $\textsc{PrintYs}(\lfloor n/2 \rfloor)$

**for** $i \leftarrow 1$ **to** $n^3$ **do** print("Y")

# Divide-and-Conquer

(b) **Algorithm** PRINTYS ($n$ : **integer**)

```
if n < 2
    print("Y")
else
    for j ← 1 to 16 do PrintYs(⌊n/2⌋)
    for i ← 1 to n^3 do print("Y")
```

(b)
There are 16 recursive calls, each with parameter $\lfloor n/2 \rfloor$. Since we are looking for an asymptotic solution, we can ignore rounding. Then the number of letters printed can be expressed by the recurrence:

$$X(n) = 16X(n/2) + n^3.$$

We apply the Master Theorem with $a = 16$, $b = 2$, $c = 1$, $d = 3$. Here, we have $a > b^d$, so the solution is $\Theta(n^{\log_2 16})$.

# Divide-and-Conquer

(c) **Algorithm** PrintZs ($n$ : **integer**)
    **if** $n < 3$
        print("Z")
    **else**
        PrintZs($\lceil n/3 \rceil$)
        PrintZs($\lceil n/3 \rceil$)
        **for** $i \leftarrow 1$ **to** $7n$ **do** print("Z")

# Divide-and-Conquer

(c) **Algorithm** PRINTZS ($n$ : **integer**)

  **if** $n < 3$

    print("Z")

  **else**

    PRINTZS($\lceil n/3 \rceil$)

    PRINTZS($\lceil n/3 \rceil$)

    **for** $i \leftarrow 1$ **to** $7n$ **do** print("Z")

(c)

There are 2 recursive calls, each with parameter $\lceil n/3 \rceil$. Since we are looking for an asymptotic solution, we can ignore rounding. Then the number of letters printed can be expressed by the recurrence:

$$X(n) = 2X(n/3) + 7n.$$

We apply the Master Theorem with $a = 2$, $b = 3$, $c = 7$, $d = 1$. Here, we have $a < b^d$, so the solution is $\Theta(n)$.

# Divide-and-Conquer

(d) **Algorithm** PrintUs ($n$ : **integer**)

```
if n < 4
    print("U")
else
    PrintUs(⌈n/4⌉)
    PrintUs(⌊n/4⌋)
    for i ← 1 to 11 do print("U")
```

# Divide-and-Conquer

(d) **Algorithm** PrintUs ($n$ : **integer**)
    **if** $n < 4$
        print("U")
    **else**
        PrintUs($\lceil n/4 \rceil$)
        PrintUs($\lfloor n/4 \rfloor$)
        **for** $i \leftarrow 1$ **to** 11 **do** print("U")

(d)
There are 2 recursive calls, each with parameter $\lceil n/4 \rceil$. Since we are looking for an asymptotic solution, we can ignore rounding. Then the number of letters printed can be expressed by the recurrence:

$$X(n) \quad = \quad 2X(n/4) + 11.$$

We apply the Master Theorem with $a = 2$, $b = 4$, $c = 11$, $d = 0$. Here, we have $a > b^d$, so the solution is $\Theta(n^{\log_4 2})$.

# Divide-and-Conquer

(e) **Algorithm** $\textsc{PrintVs}$ ($n$ : **integer**)

    **if** $n < 3$

        print("V")

    **else**

        **for** $j \leftarrow 1$ **to** 9 **do** $\textsc{PrintVs}(\lfloor n/3 \rfloor)$

        **for** $i \leftarrow 1$ **to** $2n^3$ **do** print("V")

# Divide-and-Conquer

(e) **Algorithm** PRINTVS ($n$ : **integer**)

  **if** $n < 3$

    print("V")

  **else**

    **for** $j \leftarrow 1$ **to** 9 **do** PRINTVS($\lfloor n/3 \rfloor$)

    **for** $i \leftarrow 1$ **to** $2n^3$ **do** print("V")

(e)
There are 9 recursive calls, each with parameter $\lfloor n/3 \rfloor$. Since we are looking for an asymptotic solution, we can ignore rounding. Then the number of letters printed can be expressed by the recurrence:

$$X(n) = 9X(n/3) + 2n^3.$$

We apply the Master Theorem with $a = 9$, $b = 3$, $c = 2$, $d = 3$. Here, we have $a < b^d$, so the solution is $\Theta(n^3)$.