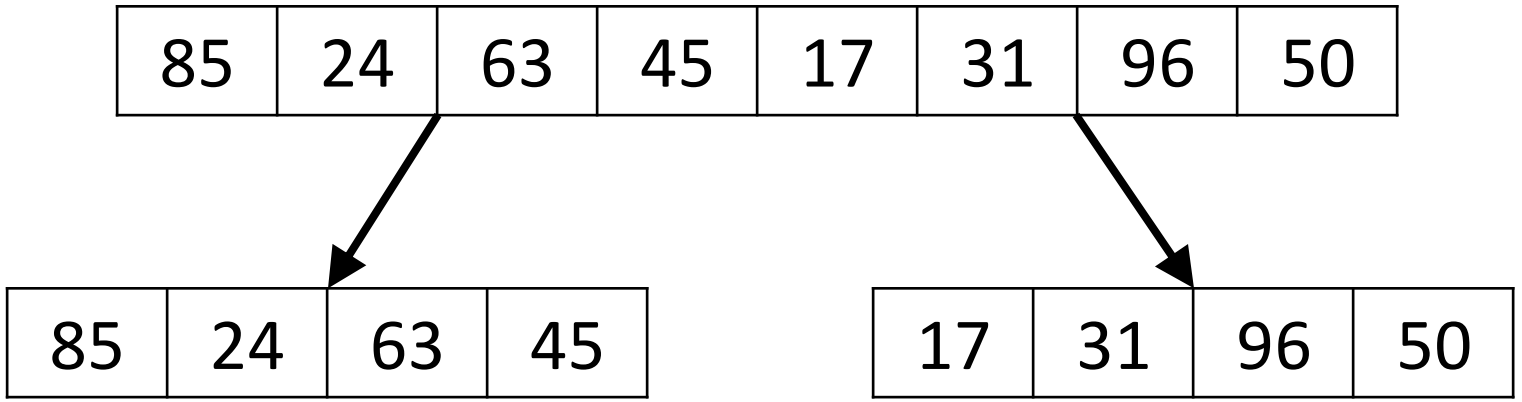
Recurrence Relations

CSCI 432

Divide and Conquer - Merge Sort

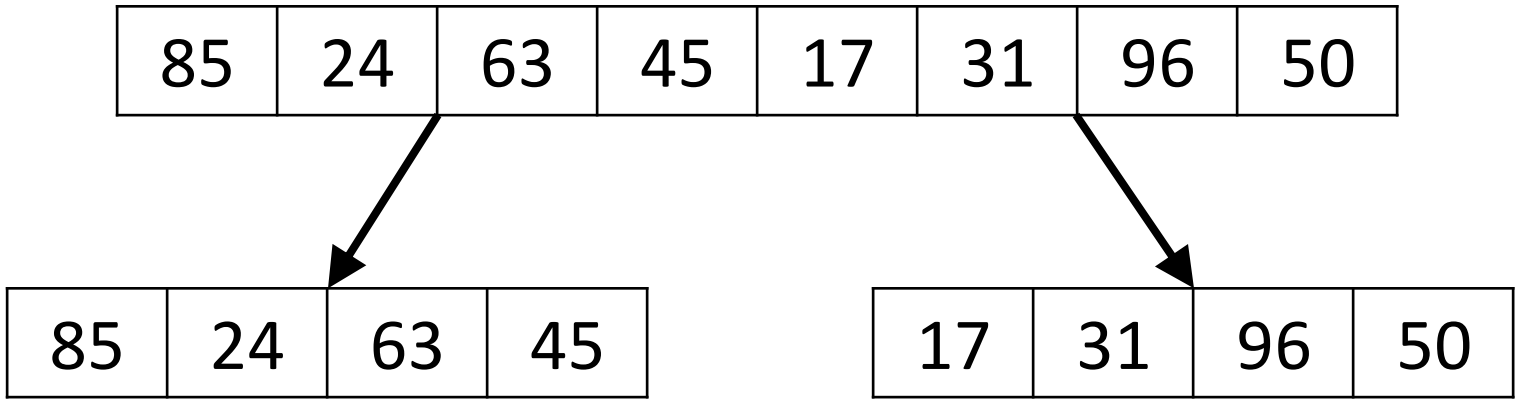
85	24	63	45	17	31	96	50

Divide and Conquer - Merge Sort



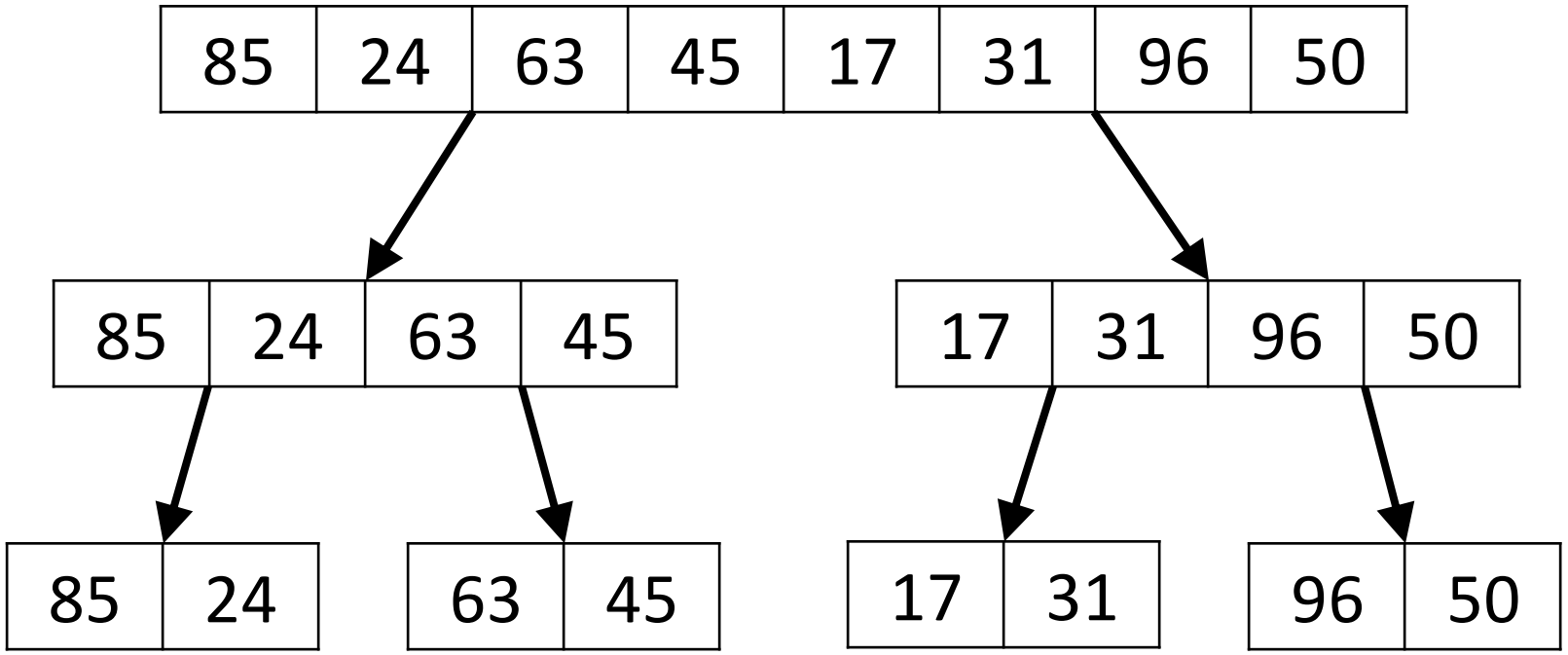
1. Divide: Split problem into two (roughly) equal parts.

Divide and Conquer - Merge Sort



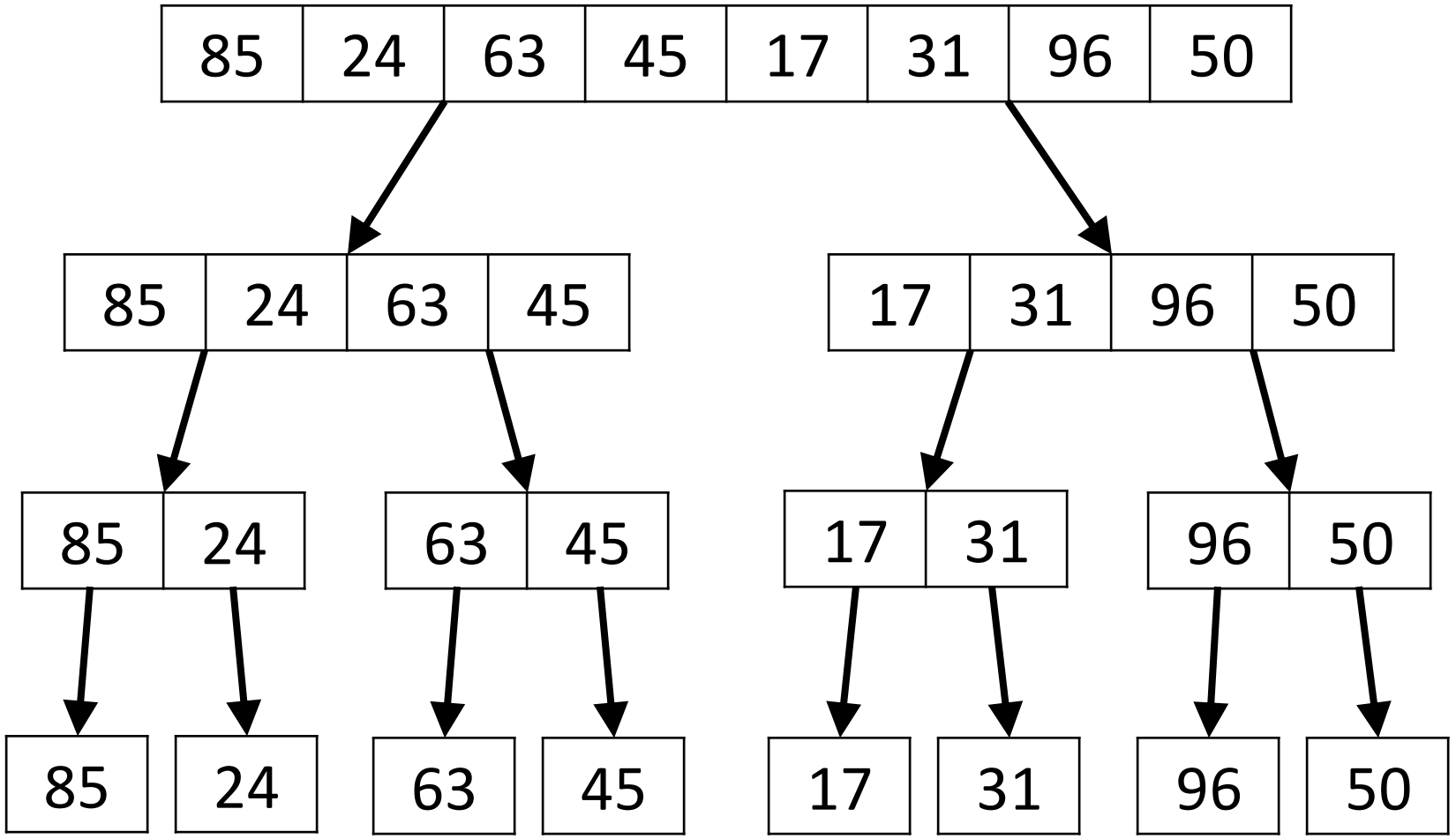
1. Divide: Split problem into two (roughly) equal parts.
2. Conquer: Sort the parts.

Divide and Conquer - Merge Sort



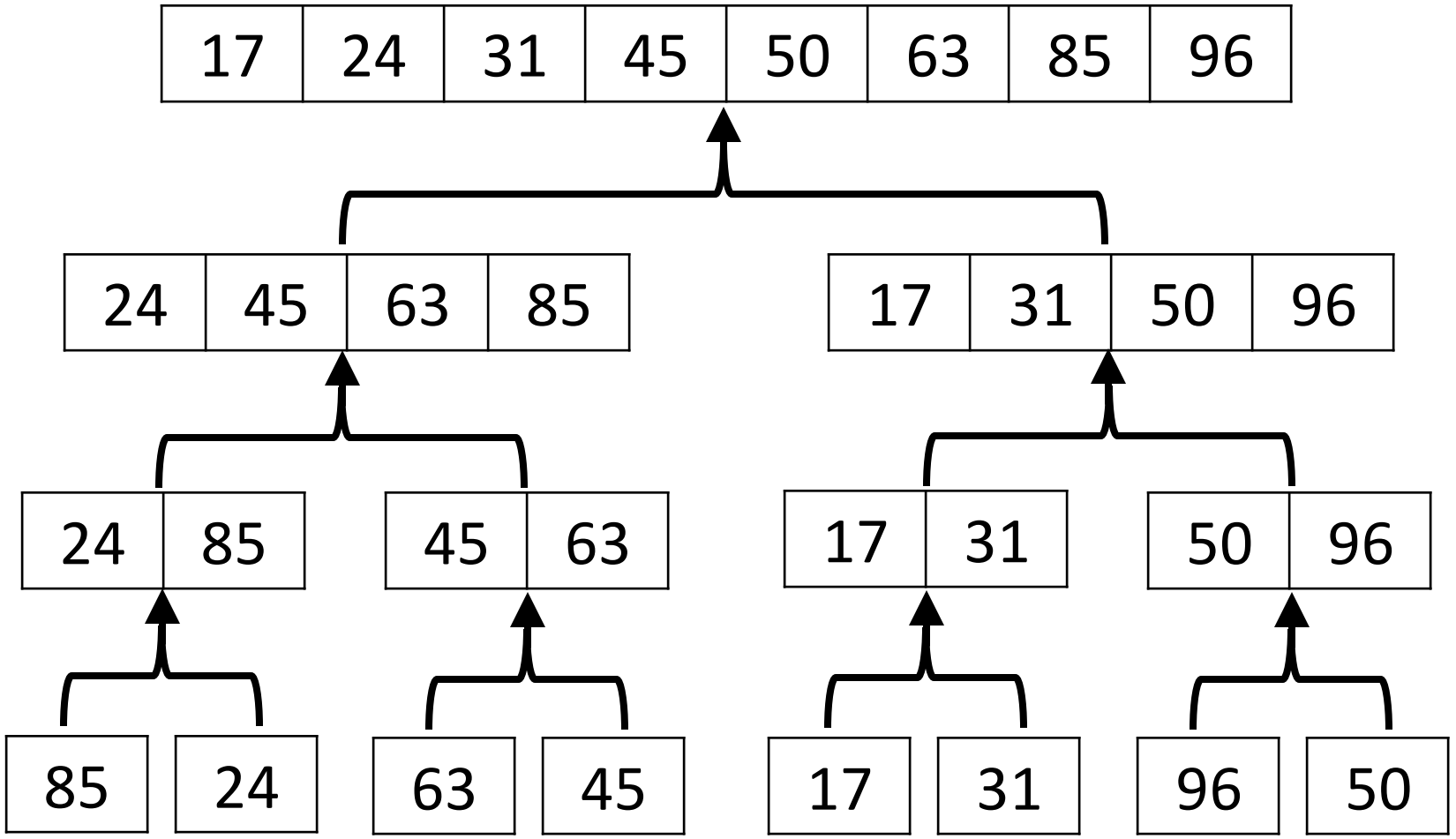
1. Divide: Split problem into two (roughly) equal parts.
2. Conquer: Sort the parts.

Divide and Conquer - Merge Sort



1. Divide: Split problem into two (roughly) equal parts.
2. Conquer: Sort the parts.

Divide and Conquer - Merge Sort



1. Divide: Split problem into two (roughly) equal parts.
2. Conquer: Sort the parts.
3. Combine: Merge the sorted parts.

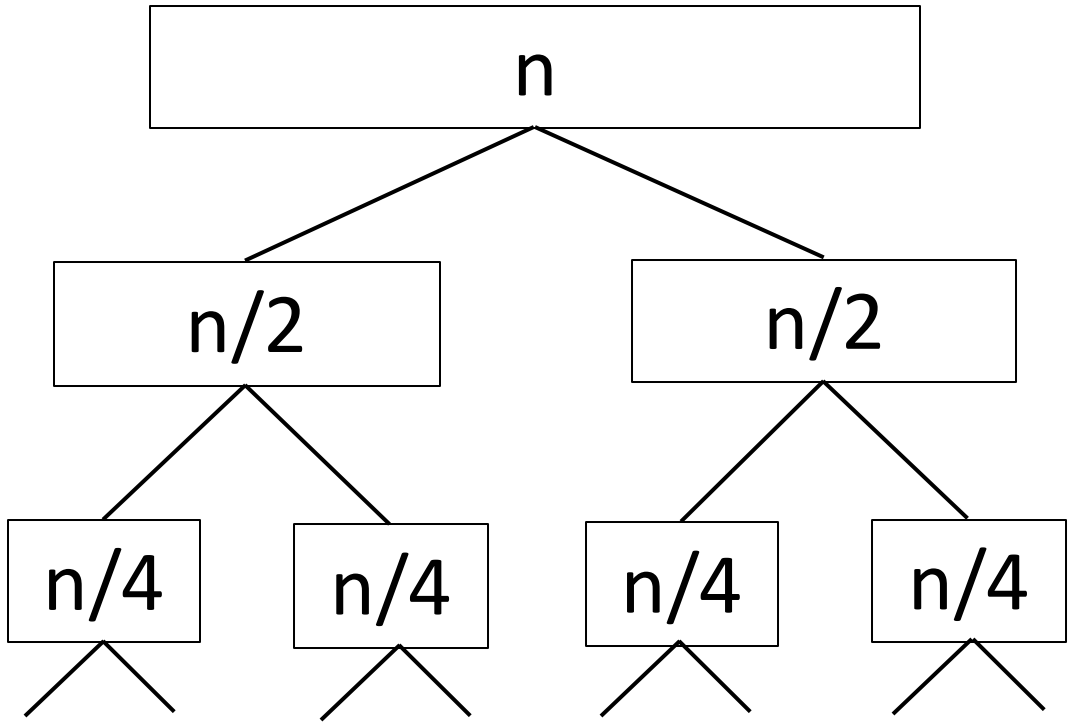
Merge Sort – Running Time

```
mergesort(a[0...n])
    if a.length > 1
        return merge(mergesort(a[0...floor(n/2)]),
                     mergesort(a[floor(n/2)+1...n]))
    else
        return a[0];
```

Can't count for/while loops with a recursive algorithm!

```
merge(x[0...m], y[0...k])
    if x.length == 0
        return y[0...k]
    if y.length == 0
        return x[0...m]
    if x[0] ≤ y[0]
        return x[0] ○ merge(x[1...m], y[0...k])
    else
        return y[0] ○ merge(x[0...k], y[1...k])
```

Merge Sort – Running Time



How much work is done at each level?

$O(n)$ – copy arrays and merge back together.

Suppose there are k levels. How tall is the tree in terms of n?

For level k, there are 2^k arrays.

At the bottom, there is one element per array.

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

$\Rightarrow O(n \log(n))$ total running time.

Recurrence Relations Running Time

Recursive divide and conquer running time can be characterized as:

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

Where, a – Number of subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Recurrence Relations Running Time

Recursive divide and conquer running time can be characterized as:

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

Where, a – Number of subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Master theorem: If $T(n) = aT(n/b) + O(n^d)$ for constants $a \geq 1, b > 1, d \geq 0$, then:

Recurrence Relations Running Time

Recursive divide and conquer running time can be characterized as:

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

Where, a – Number of subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

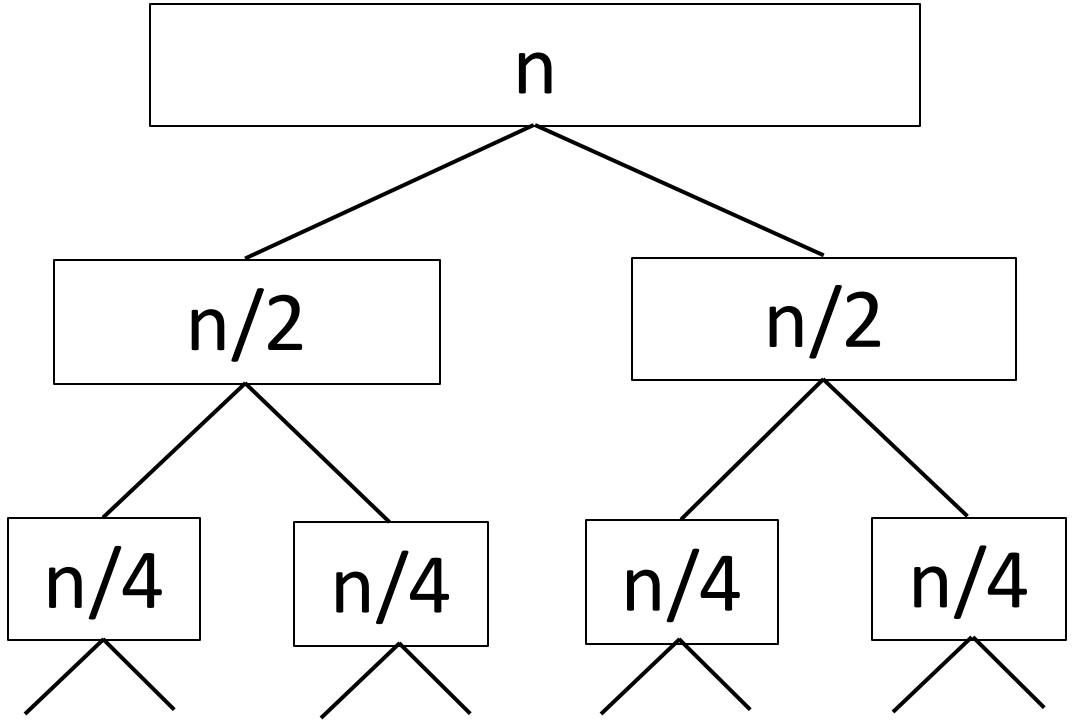
$C(n)$ – time to combine problems

Master theorem: If $T(n) = aT(n/b) + O(n^d)$ for constants $a \geq 1, b > 1, d \geq 0$, then:

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Merge Sort – Running Time

$$\boldsymbol{T(n) = aT(n/b) + D(n) + C(n)}$$



a – Number of subproblems at each recursive call

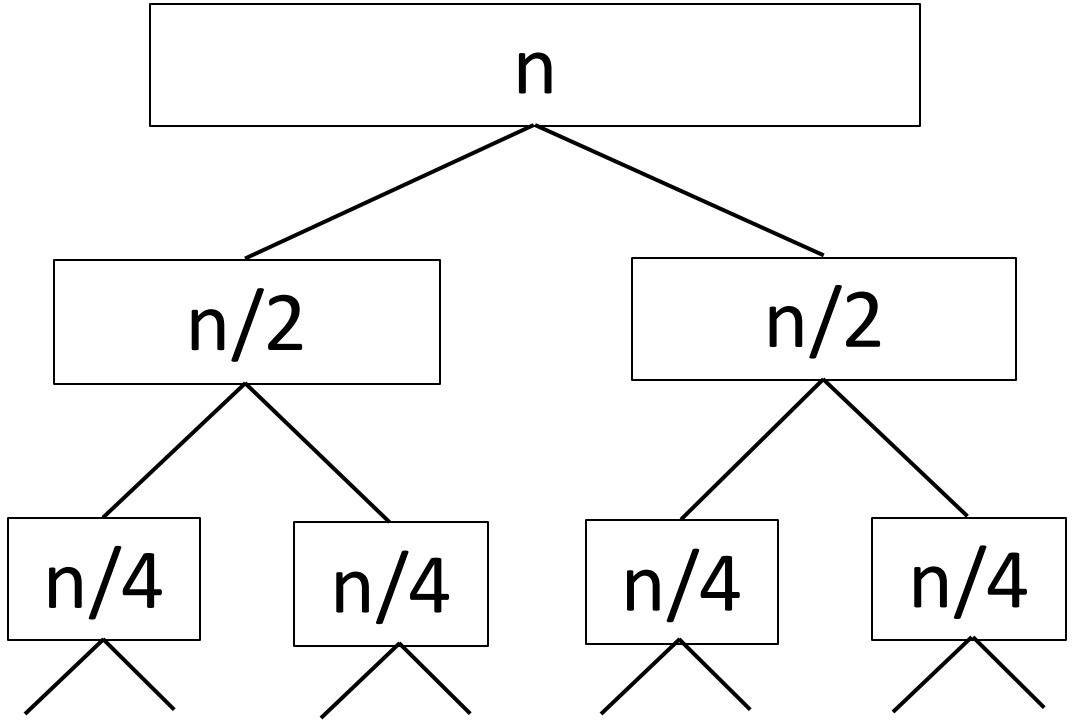
n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Merge Sort – Running Time

$$\boldsymbol{T(n) = aT(n/b) + D(n) + C(n)}$$



a – Number of subproblems at each recursive call **2**

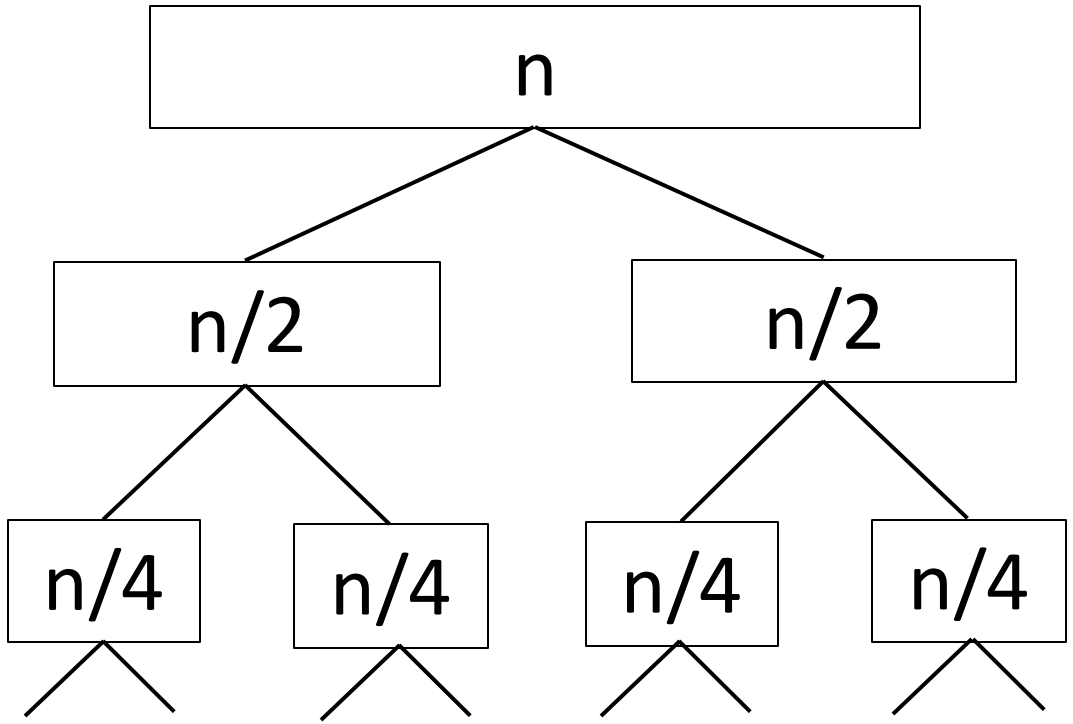
n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Merge Sort – Running Time

$$\boldsymbol{T(n) = aT(n/b) + D(n) + C(n)}$$



a – Number of subproblems at each recursive call **2**

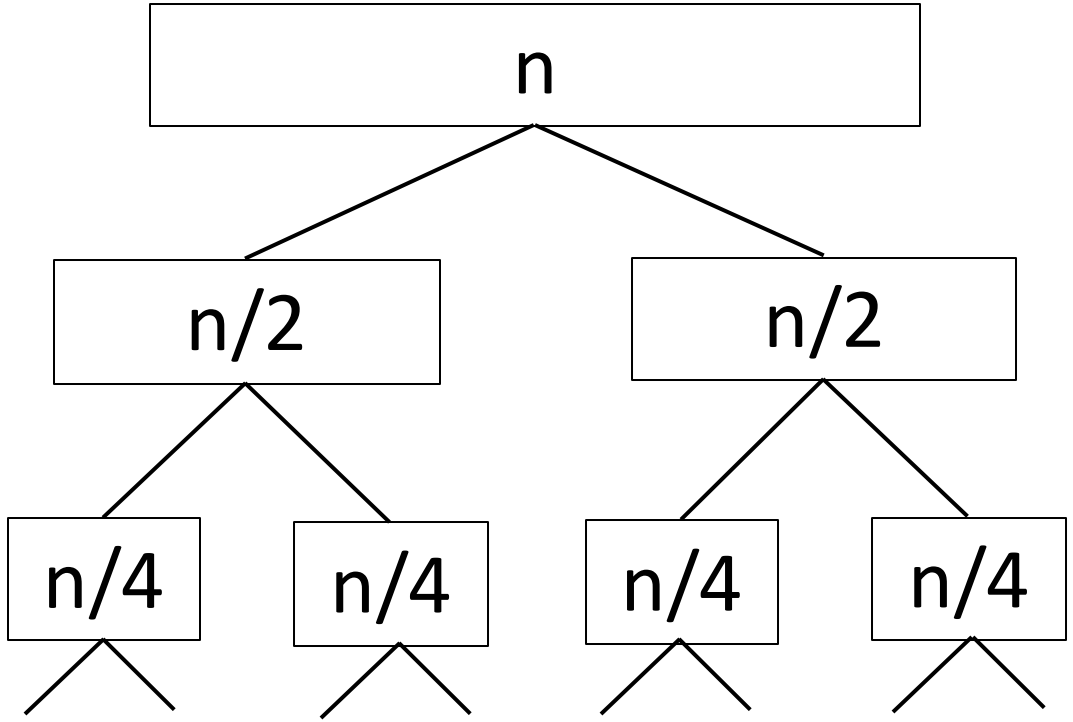
n/b – Size of subproblem relative to previous $\frac{n}{2}$

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Merge Sort – Running Time

$$\boldsymbol{T(n) = aT(n/b) + D(n) + C(n)}$$



a – Number of subproblems at each recursive call **2**

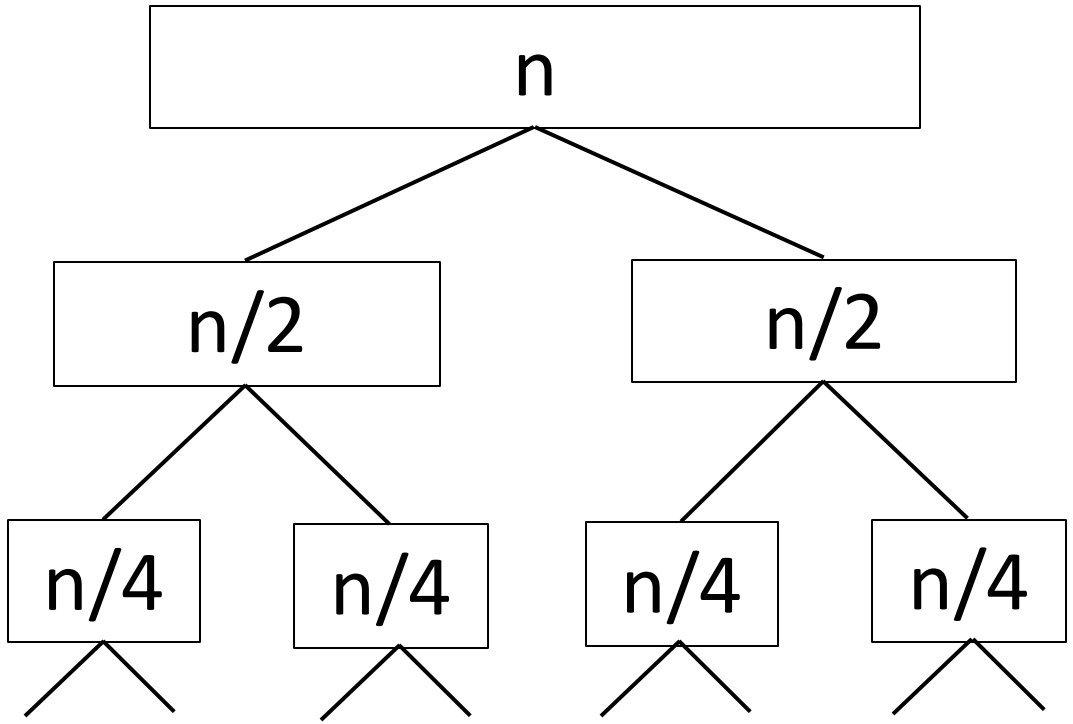
n/b – Size of subproblem relative to previous $\frac{n}{2}$

$D(n)$ – time to divide problems $O(n)$

$C(n)$ – time to combine problems

Merge Sort – Running Time

$$T(n) = 2T(n/2) + O(n)$$



a – Number of subproblems at each recursive call **2**

n/b – Size of subproblem relative to previous $\frac{n}{2}$

$D(n)$ – time to divide problems $O(n)$

$C(n)$ – time to combine problems $O(n)$

Merge Sort – Running Time

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

a – Number of subproblems at each recursive call **$\boldsymbol{2}$**

n/b – Size of subproblem relative to previous **$\boldsymbol{\frac{n}{2}}$**

$D(n)$ – time to divide problems **$\boldsymbol{O(n)}$**

$C(n)$ – time to combine problems **$\boldsymbol{O(n)}$**

Merge Sort – Running Time

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$\boldsymbol{\log_b a = ??}$

a – Number of subproblems at each recursive call **2**

n/b – Size of subproblem relative to previous $\boldsymbol{\frac{n}{2}}$

$D(n)$ – time to divide problems $\boldsymbol{O(n)}$

$C(n)$ – time to combine problems $\boldsymbol{O(n)}$

Merge Sort – Running Time

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$$\boldsymbol{\log_b a = \log_2 2 = 1 = d}$$

a – Number of subproblems at each recursive call **2**

n/b – Size of subproblem relative to previous $\boldsymbol{\frac{n}{2}}$

$D(n)$ – time to divide problems $\boldsymbol{O(n)}$

$C(n)$ – time to combine problems $\boldsymbol{O(n)}$

Merge Sort – Running Time

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) \in \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$$\boldsymbol{\log_b a = \log_2 2 = 1 = d}$$

$$\boldsymbol{\Rightarrow T(n) \in O(n \log n)}$$

Running time of Merge Sort.

a – Number of subproblems at each recursive call **2**

n/b – Size of subproblem relative to previous $\boldsymbol{\frac{n}{2}}$

$D(n)$ – time to divide problems $\boldsymbol{O(n)}$

$C(n)$ – time to combine problems $\boldsymbol{O(n)}$

Binary Search

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = ??$

$b = ??$

$D(n) \in ??$

$C(n) \in ??$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = 1$

$b = ??$

$D(n) \in ??$

$C(n) \in ??$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = 1$

$b = 2$

$D(n) \in$??

$C(n) \in$??

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = 1$

$b = 2$

$D(n) \in O(1)$

$C(n) \in$??

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = 1$

$b = 2$

$D(n) \in O(1)$

$C(n) \in O(1)$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

$$T(n) = T(n/2) + O(1)$$

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

a – Num subproblems at each recursive call

n/b – Size of subproblem relative to previous

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

$a = 1$

$b = 2$

$D(n) \in O(1)$

$C(n) \in O(1)$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

$$T(n) = T(n/2) + O(1)$$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$a = 1$

$b = 2$

$D(n) \in O(1)$

$C(n) \in O(1)$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

$$T(n) = T(n/2) + O(1)$$

$$\log_b a = \log_2 1 = 0 = d$$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$a = 1$

$b = 2$

$D(n) \in O(1)$

$C(n) \in O(1)$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Binary Search

$$T(n) = T(n/2) + O(1)$$

$$\log_b a = \log_2 1 = 0 = d$$

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

$a = 1$

$b = 2$

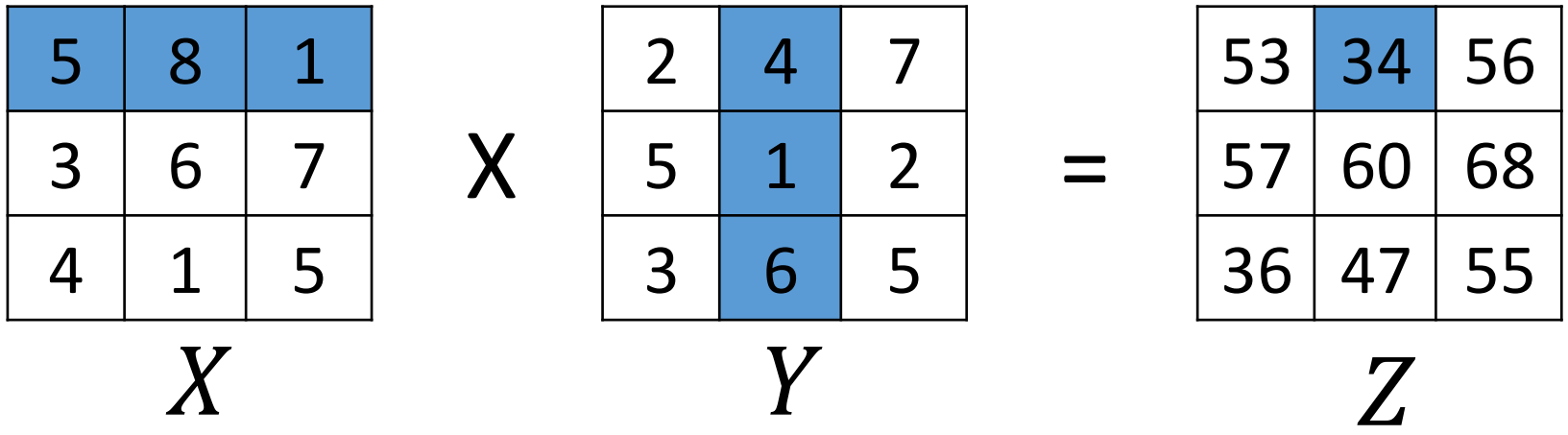
$D(n) \in O(1)$

$C(n) \in O(1)$

```
binary_search(sorted a[0...n], T)
   if a.length == 0
      return false
   else if T < a[n/2]
      return binary_search(a[0...n/2], T)
   else if T > a[n/2]
      return binary_search(a[(n/2)+1...n], T)
   else return true
```

Matrix Multiplication

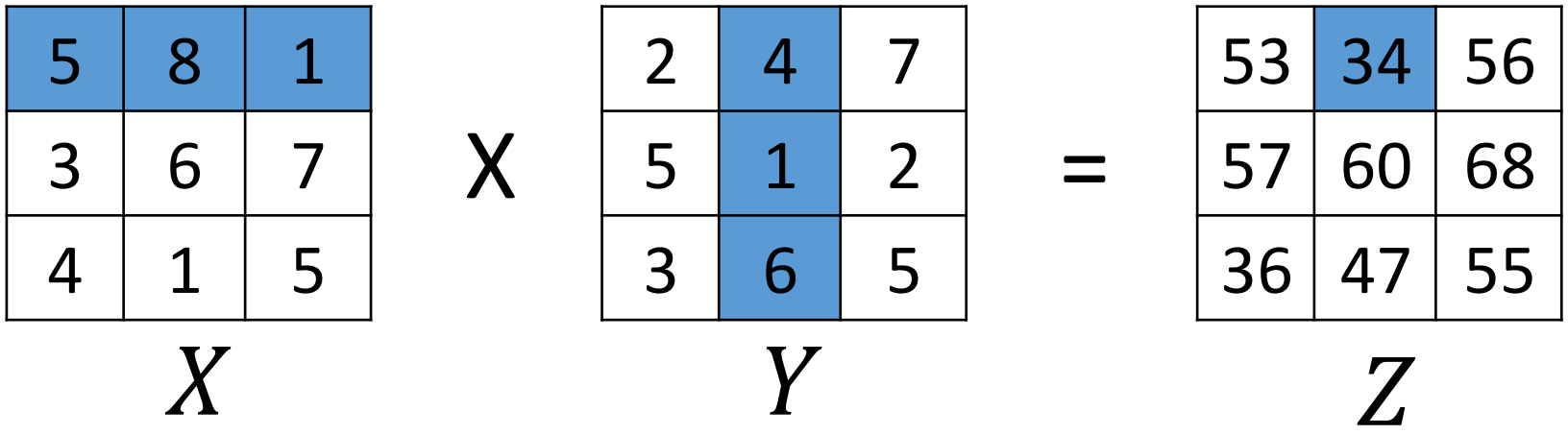
$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$



X Y Z

Matrix Multiplication

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$



X Y Z

X, Y, Z are $n \times n$ matrices. Running time?

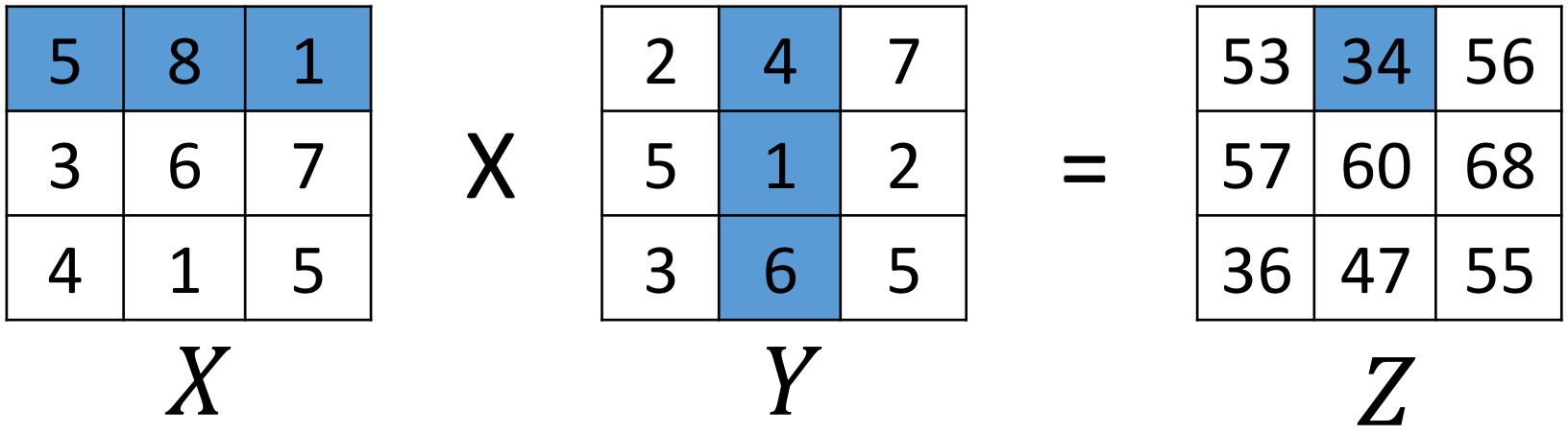
How big is the solution - ??

How long to calculate a single entry - ??

Total time - ??

Matrix Multiplication

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$



X Y Z

X, Y, Z are $n \times n$ matrices. Running time?

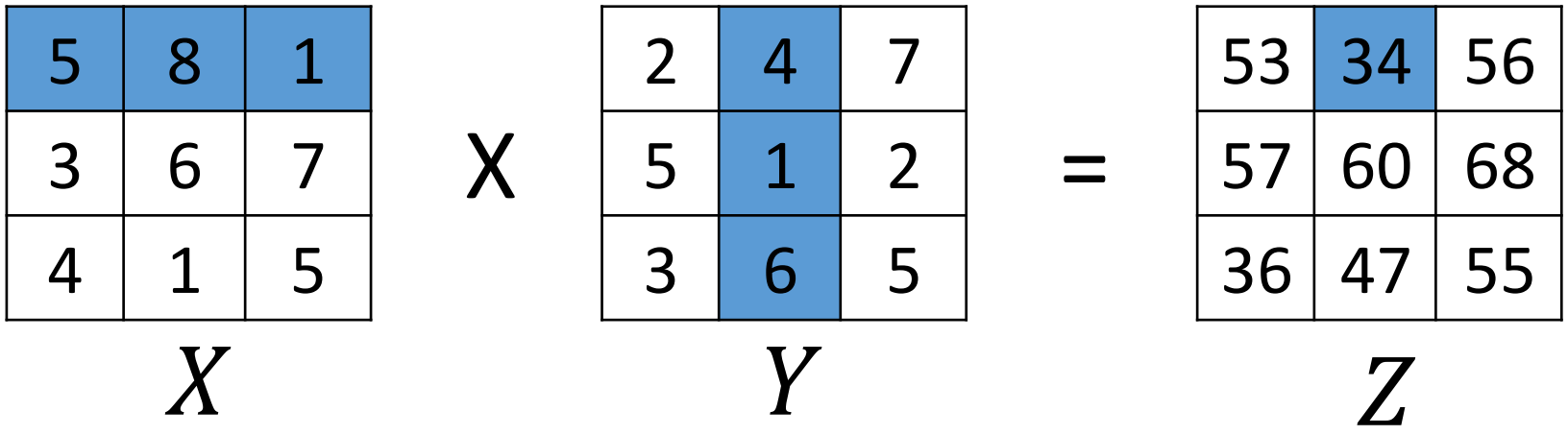
How big is the solution - n^2

How long to calculate a single entry - ??

Total time - ??

Matrix Multiplication

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$



X, Y, Z are $n \times n$ matrices. Running time?

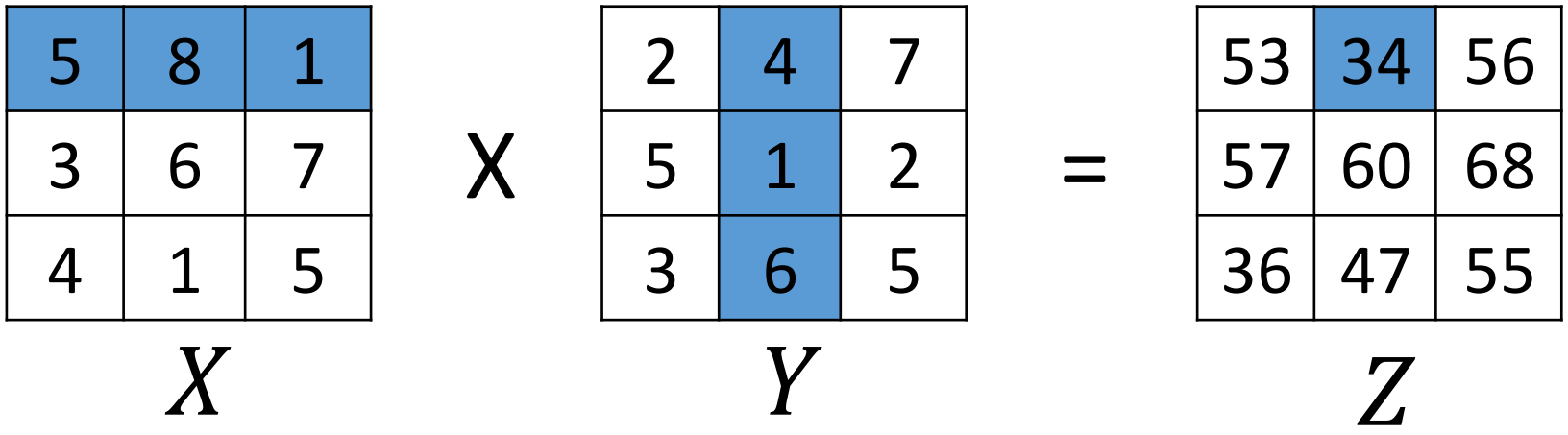
How big is the solution - n^2

How long to calculate a single entry - $O(n)$

Total time - ??

Matrix Multiplication

$$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$$



X Y Z

X, Y, Z are $n \times n$ matrices. Running time?

How big is the solution - n^2

How long to calculate a single entry - $O(n)$

Total time - $O(n^3)$

Matrix Addition

5	8	1
3	6	7
4	1	5

X

$+$

2	4	7
5	1	2
3	6	5

Y

$=$

7	12	8
8	7	9
7	7	10

Z

$Z_{ij} = X_{ij} + Y_{ij}$

X, Y, Z are $n \times n$ matrices. Running time?
How big is the solution - ?
How long to calculate a single entry - ?
Total time - ?

Matrix Addition

5	8	1
3	6	7
4	1	5

X

+

2	4	7
5	1	2
3	6	5

Y

=

7	12	8
8	7	9
7	7	10

Z

$Z_{ij} = X_{ij} + Y_{ij}$

X, Y, Z are $n \times n$ matrices. Running time?

How big is the solution - n^2

How long to calculate a single entry - $O(1)$

Total time - $O(n^2)$

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Alternate approach:

1. Break X and Y into $\frac{n}{2} \times \frac{n}{2}$ blocks.
2. Multiply blocks and add to get Z.

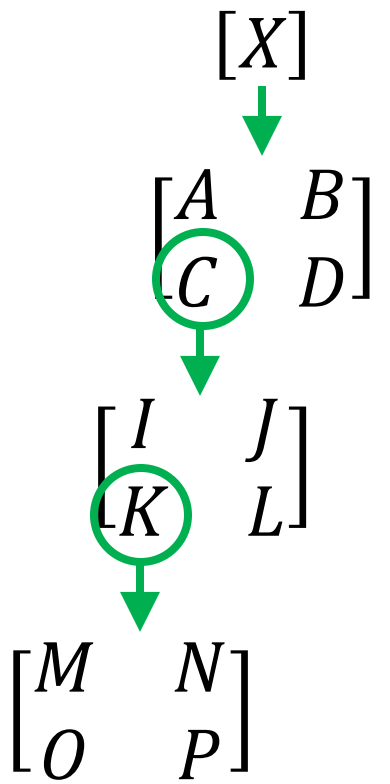
Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Alternate approach:

1. <u>Recursively</u> break X and Y into $\frac{n}{2} \times \frac{n}{2}$ blocks.
2. Multiply blocks and add to get Z.



Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ x } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ x } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = ??$

$b = ??$

$D(n) \in ??$

$C(n) \in ??$

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b =$??

$D(n) \in$??

$C(n) \in$??

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in$??

$C(n) \in$??

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in$??

4 matrix additions!

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 8T(n/2) + O(n^2)$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 8T(n/2) + O(n^2)$

$\log_b a = \log_2 8 = 3$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 8T(n/2) + O(n^2)$

$\log_b a = \log_2 8 = 3 > 2 = d$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Running time?

$a = 8$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 8T(n/2) + O(n^2)$

$\log_b a = \log_2 8 = 3 > 2 = d$

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

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$P_1 = A(F - H) \quad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \quad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \quad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ X } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ X } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$\boldsymbol{P_1 + P_2}$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$\boldsymbol{P_1 + P_2}$
$\boldsymbol{= A(F - H) + (A + B)H}$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$$\boldsymbol{P_1 + P_2}$$
$$\boldsymbol{= A(F - H) + (A + B)H}$$
$$\boldsymbol{= AF - AH + AH + BH}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ X } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

$$\boldsymbol{P_1 + P_2}$$
$$\boldsymbol{= A(F - H) + (A + B)H}$$
$$\boldsymbol{= AF - AH + AH + BH}$$
$$\boldsymbol{= AF + BH}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ X } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ X } \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X Y Z

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = ??$

$b = ??$

$D(n) \in ??$

$C(n) \in ??$

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = ??$

$D(n) \in ??$

$C(n) \in ??$

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in$??

$C(n) \in$??

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in$??

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

18 matrix additions/subtractions!

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

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

a – Number of subproblems

n/b – Size of subproblem

$D(n)$ – time to divide problems

$C(n)$ – time to combine problems

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 7T(n/2) + O(n^2)$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 7T(n/2) + O(n^2)$

$\log_b a = \log_2 7 \cong 2.81 > 2 = d$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 7T(n/2) + O(n^2)$

$\log_b a = \log_2 7 \cong 2.81 > 2 = d$

$\Rightarrow T(n) = O(n^{2.81})$

Master theorem:

If $T(n) = aT(n/b) + O(n^d)$, then

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Matrix Multiplication

$$Z = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

for,

$$P_1 = A(F - H) \qquad P_5 = (A + D)(E + H)$$
$$P_2 = (A + B)H \qquad P_6 = (B - D)(G + H)$$
$$P_3 = (C + D)E \qquad P_7 = (A - C)(E + F)$$
$$P_4 = D(G - E)$$

Running time?

$a = 7$

$b = 2$

$D(n) \in O(n^2)$

$C(n) \in O(n^2)$

$T(n) = 7T(n/2) + O(n^2)$

$\log_b a = \log_2 7 \cong 2.81 > 2 = d$

$\Rightarrow T(n) = O(n^{2.81})$

$O(n^{2.81})$ vs $O(n^3)$?

1 second vs 2.6 seconds

1 hour vs 4.5 hours

1 day vs 5.5 days

3 days vs 18 days

Matrix Multiplication

History of Matrix Multiplication Algorithms

- $O(n^3)$ – 1812 (Definition of matrix multiplication)
- $O(n^{2.81})$ – 1969
- $O(n^{2.3755})$ – 1990
- $O(n^{2.3737})$ – 2010
- $O(n^{2.3729})$ – 2013
- $O(n^{2.3728639})$ – 2014
- $O(n^{2.3728596})$ – 2020
- $O(n^{2.371866})$ – 2022
- $O(n^{2.371552})$ – 2024

Lower bound: $O(n^2)$. Need to look at all elements of both matrices.

Runni

ds

$T(n)$

log

$\Rightarrow T(n) = O(n^{2.81})$