Design and Analysis of Algorithms

Presented by Dr. Li Ning

Shenzhen Institutes of Advanced Technology, Chinese Academy of Science
Shenzhen, China

Divide and Conquer

Revisit: Merge Sort

Merge Sort

$m = \lfloor (|A| - 1)/2 \rfloor;$

$B = A[0, ..., m];$

$C = A[m + 1, ..., |A| - 1];$

$B =$MergeSort$(B);$

$C =$MergeSort$(C);$

$A =$Merge$(B,\ C);$

Merge Sort

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ & \uparrow & & \\ & m & & \end{array} \qquad \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ & & \uparrow & & \\ & & m & & \end{array}$$

$m = \lfloor (|A| - 1)/2 \rfloor;$

$B = A[0, ..., m];$

$C = A[m + 1, ..., |A| - 1];$

$B =$MergeSort$(B);$

$C =$MergeSort$(C);$

$A =$Merge$(B,\ C);$

Merge Sort

$$0 \quad \underset{\substack{\uparrow \\ m}}{1} \quad 2 \quad 3 \qquad\qquad 0 \quad 1 \quad \underset{\substack{\uparrow \\ m}}{2} \quad 3 \quad 4$$

$m = \lfloor (|A| - 1)/2 \rfloor;$

$B = A[0, ..., m];$

$$0 \quad 1 \quad \underset{\substack{\uparrow \\ m}}{2} \qquad 3 \quad 4$$

$C = A[m + 1, ..., |A| - 1];$

$B =$MergeSort$(B);$

$C =$MergeSort$(C);$

$A =$Merge$(B,\ C);$

Merge Sort

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ & \uparrow & & \\ & m & & \end{array} \qquad \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ & & \uparrow & & \\ & & m & & \end{array}$$

$m = \lfloor (|A| - 1)/2 \rfloor;$

$B = A[0, ..., m];$

$$\begin{array}{ccccc} 0 & 1 & 2 & & 3 \quad 4 \\ & & \uparrow & & \\ & & m & & \end{array}$$

$C = A[m + 1, ..., |A| - 1];$

$B =$MergeSort(B);

$C =$MergeSort(C);

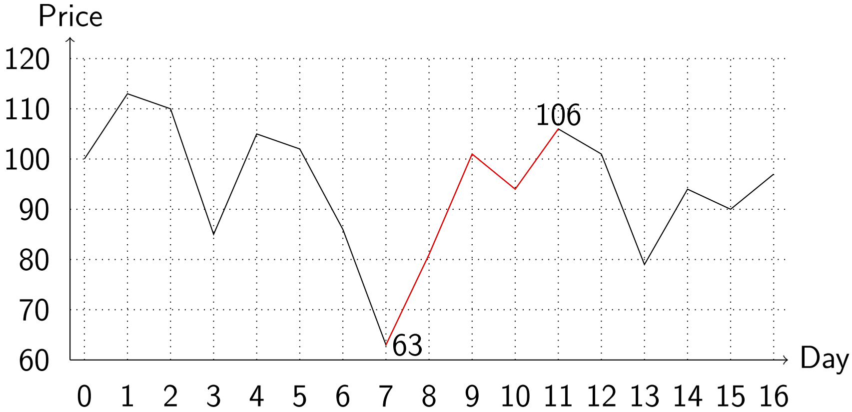
$A =$Merge$(B,\ C)$;

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

Example: The Maximum-Subarray Problem

Buy Low and Sell High

Buy Low and Sell High



Buy Low and Sell High

Buy Low and Sell High

Buy Low and Sell High

Buy Low and Sell High

Buy Low and Sell High

Brute Force: $O(n^2)$

Algorithm: BuySell(P)

```
n = |P|;
b = s = −1;
max = 0;
for i = 0 to n − 2 do
    for j = i + 1 to n − 1 do
        p = P[j] − P[i];
        if p > max then
            max = p;
            b = i;
            s = j;
        end
    end
end
Return b, s;
```

Sum of The Changes

Day	0	1	2	3	4	5	6	7
Price	100	110	90	103	115	108	95	101

Sum of The Changes

Day	0	1	2	3	4	5	6	7
Price	100	110	90	103	115	108	95	101

Sum of The Changes

Day	0	1	2	3	4	5	6	7
Price	100	110	90	103	115	108	95	101

$$
\begin{aligned}
-90 + 108 &= -90 + 103 - 103 + 115 - 115 + 108 \\
&= (103 - 90) + (115 - 103) + (108 - 115)
\end{aligned}
$$

Sum of The Changes

Day	0	1	2	3	4	5	6	7
Price	100	110	90	103	115	108	95	101
Change	0	10	-20	13	12	-7	-13	6

$$
\begin{aligned}
-90 + 108 &= -90 + 103 - 103 + 115 - 115 + 108 \\
&= (103 - 90) + (115 - 103) + (108 - 115)
\end{aligned}
$$

Sum of The Changes

Day	0	1	2	3	4	5	6	7
Price	100	110	90	103	115	108	95	101
Change	0	10	-20	13	12	-7	-13	6

$$\begin{aligned} -90 + 108 &= -90 + 103 - 103 + 115 - 115 + 108 \\ &= (103 - 90) + (115 - 103) + (108 - 115) \end{aligned}$$

$$\arg\max_{(i,j)} P[j] - P[i] \iff \arg\max_{(i,j)} \sum_{k=i+1}^{j} C[k]$$

Find The Maximum-Subarray

Problem: Given an array, find the continuous subarray of the maximum sum.

$L = [0, -2, 3, 1, -2, 5, -6, 2]$
maximum-subarray:
$3 + 1 - 2 + 5 = 7$

Algorithm: MaxSub(C)

$b = s = -1$;
$max = 0$;
for i, j **do**
    $sum = \sum_{k=i+1}^{j} C[k]$;
    if $sum > max$ **then**
        $max = p$;
        $b = i$;
        $s = j$;
    end
end
Return b, s;

Find The Maximum-Subarray

10	12	-30	5	20	-25	18	3

Find The Maximum-Subarray

10	12	-30	5	20	-25	18	3

Find The Maximum-Subarray

10	12	-30	5	20	-25	18	3
10	12	-30	5				

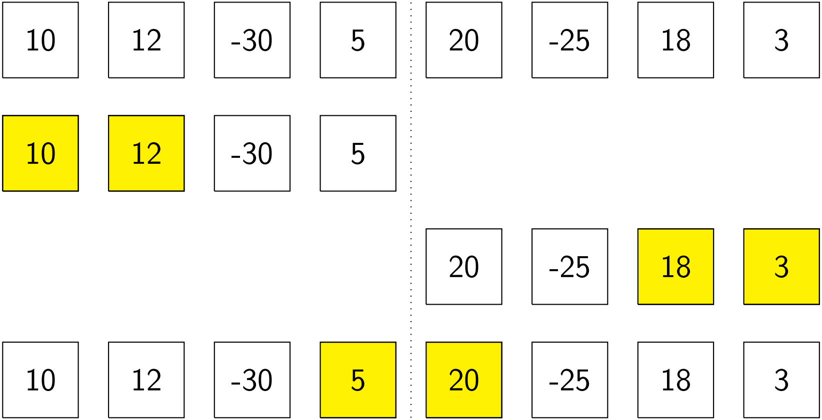
Find The Maximum-Subarray

10	12	-30	5	20	-25	18	3

10	12	-30	5

20	-25	18	3

Find The Maximum-Subarray



Maximum-Subarray Crossing The Separator

Algorithm: MaxCross(C, m)

```
s = 0;
if m < |C| - 1 then
    a = 0;
    for i = m to 0 do
        a = max(a, sum(C[i, m]));
    end
    b = 0;
    for i = m + 1 to |C| - 1 do
        b = max(b, sum(C[m + 1, i]));
    end
    s = a + b;
end
Return s;
```

Find The Maximum-Subarray

- Divide C into two parts:
  - A: the first half
  - B: the second half
- Find the maximum subarray in each part: $s1$ and $s2$.
- Find the maximum subarray crossing the separator: $s3$.
- Compare $s1, s2$ and $s3$ to find the maximum one.

Algorithm: MaxSub2(C)

```
s = C[0];
if |C| > 1 then
    m = ⌊(|C| − 1)/2⌋;
    A = C[0, m];
    B = C[m + 1, |C| − 1];
    s1 = MaxSub2(A);
    s2 = MaxSub2(B);
    s3 = MaxCross(C, m);
    s = max(s1, s2, s3);
end
Return s;
```

Find The Maximum-Subarray: $O(n \log n)$

$$
\begin{array}{lll}
1: & T(n) = 2T(n/2) + n & \longrightarrow \quad n \\
2: & \quad = 4T(n/4) + 2(n/2) + n & \longrightarrow \quad 2n \\
3: & \quad = 8T(n/8) + 4(n/4) + 2(n/2) + n & \longrightarrow \quad 3n \\
i: & \quad = 2^i T(n/2^i) + 2^{i-1}(n/2^{i-1}) + \cdots + n & \longrightarrow i \cdot n \\
k: & \quad = 2^k T(n/2^k) + 2^{k-1}(n/2^{k-1}) + \cdots + n & \longrightarrow k \cdot n
\end{array}
$$

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

The Paradigm of Divide and Conquer

Divide and Conquer

1. **Divide** the problem instance into two/several smaller instances of the same problem.
2. **Conquer** the smaller problems.
3. **Combine** the results of the smaller problems to get the result of the original (large) instance.

Recall that in MergeSort,

1. **Divide** the numbers into two subsets.
2. **Conquer** the sorting problem on each of the subsets.
3. **Combine** the results, by Merge.

Recursion tree of $T(n) = 2T(n/2) + n$

$T(n)$ $\cdots$ n

Recursion tree of $T(n) = 2T(n/2) + n$

$$T(n) \qquad \cdots \qquad n$$

$$T(\tfrac{n}{2}) \qquad T(\tfrac{n}{2}) \qquad \cdots \qquad n$$

Recursion tree of $T(n) = 2T(n/2) + n$

$T(n)$ $\cdots$ n

$T(\frac{n}{2})$ $T(\frac{n}{2})$ $\cdots$ n

$T(\frac{n}{4})$ $T(\frac{n}{4})$ $T(\frac{n}{4})$ $T(\frac{n}{4})$ $\cdots$ n

---

Recursion tree of $T(n) = 2T(n/2) + n$

$$T(n) \qquad \cdots \quad n$$

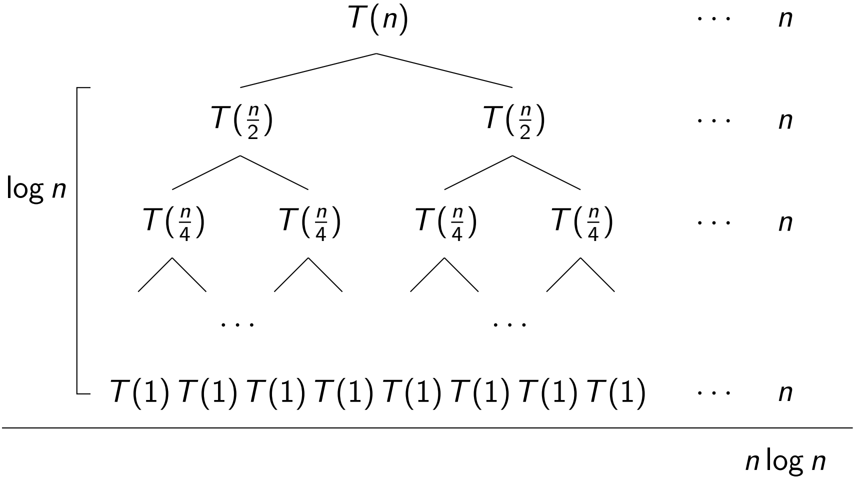
$$T(\tfrac{n}{2}) \quad T(\tfrac{n}{2}) \qquad \cdots \quad n$$

$$T(\tfrac{n}{4}) \quad T(\tfrac{n}{4}) \quad T(\tfrac{n}{4}) \quad T(\tfrac{n}{4}) \qquad \cdots \quad n$$

$$\cdots \qquad \cdots$$

$$T(1)\,T(1)\,T(1)\,T(1)\,T(1)\,T(1)\,T(1)\,T(1) \qquad \cdots \quad n$$

Recursion tree of $T(n) = 2T(n/2) + n$



Recursion tree of $T(n) = 3T(n/4) + n^2$

$T(n)$ n^2

Recursion tree of $T(n) = 3T(n/4) + n^2$

$$T(n) \qquad\qquad n^2$$

$$T\left(\frac{n}{4}\right) \qquad T\left(\frac{n}{4}\right) \qquad T\left(\frac{n}{4}\right) \qquad\qquad 3\left(\frac{n}{4}\right)^2$$

Recursion tree of $T(n) = 3T(n/4) + n^2$

$$T(n) \qquad\qquad n^2$$

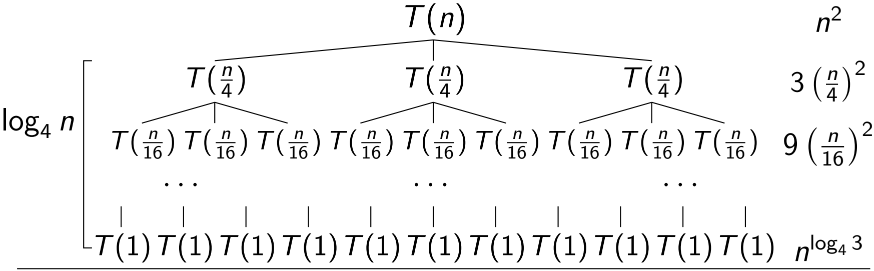
$$T\left(\tfrac{n}{4}\right) \quad T\left(\tfrac{n}{4}\right) \quad T\left(\tfrac{n}{4}\right) \qquad\qquad 3\left(\tfrac{n}{4}\right)^2$$

$$T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right)\ T\left(\tfrac{n}{16}\right) \qquad 9\left(\tfrac{n}{16}\right)^2$$

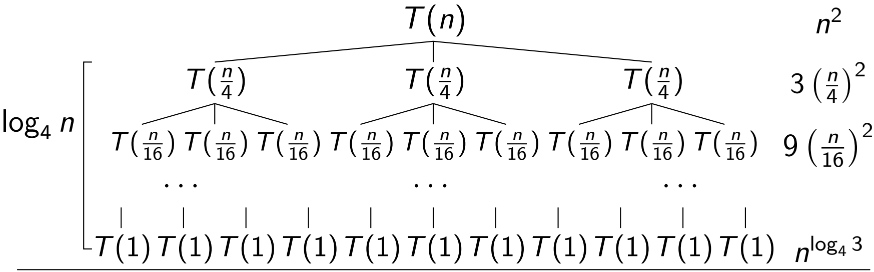
Recursion tree of $T(n) = 3T(n/4) + n^2$

$$
\begin{array}{c c}
T(n) & n^2 \\
T\left(\frac{n}{4}\right) \quad T\left(\frac{n}{4}\right) \quad T\left(\frac{n}{4}\right) & 3\left(\frac{n}{4}\right)^2 \\
T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right)\ T\left(\frac{n}{16}\right) & 9\left(\frac{n}{16}\right)^2 \\
\cdots \qquad \cdots \qquad \cdots & \\
T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1)\ T(1) & n^{\log_4 3} \\
\hline
\end{array}
$$

Recursion tree of $T(n) = 3T(n/4) + n^2$

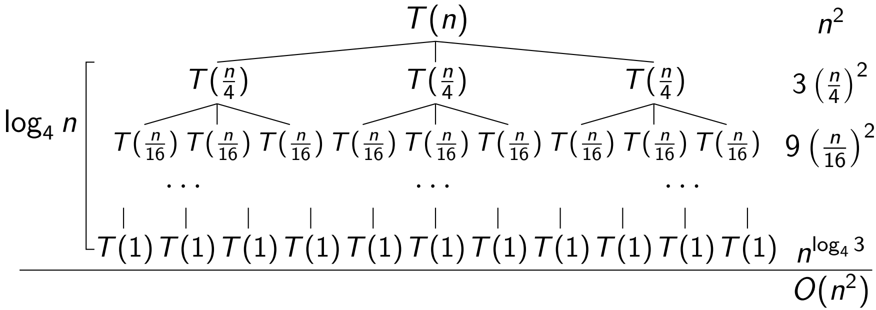


Recursion tree of $T(n) = 3T(n/4) + n^2$



$$1 + \frac{3}{16} + \left(\frac{3}{16}\right)^2 + \cdots = \Theta(1)$$

Recursion tree of $T(n) = 3T(n/4) + n^2$



$$1 + \frac{3}{16} + \left(\frac{3}{16}\right)^2 + \cdots = \Theta(1)$$

Master Theorem

$T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$

Master Theorem

$T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$

- if $af(n/b) = Kf(n)$ for some constant $K > 1$, then

$$T(n) = \Theta(n^{\log_b a})$$

Master Theorem

$T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$

- if $af(n/b) = Kf(n)$ for some constant $K > 1$, then

$$T(n) = \Theta(n^{\log_b a})$$

- if $af(n/b) = f(n)$, then

$$T(n) = \Theta(f(n) \log n)$$

Master Theorem

$T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$

- if $af(n/b) = Kf(n)$ for some constant $K > 1$, then

$$T(n) = \Theta(n^{\log_b a})$$

- if $af(n/b) = f(n)$, then

$$T(n) = \Theta(f(n) \log n)$$

- if $af(n/b) = \kappa f(n)$ for some constant $\kappa < 1$, then

$$T(n) = \Theta(f(n))$$

Master Theorem for $f(n) = O(n^d)$

$T(n) = aT(n/b) + O(n^d)$, where $a \geq 1$, $b > 1$ and $d \geq 0$,

- if $d < log_b a$ then $T(n) = O(n^{\log_b a})$;
- if $d = log_b a$, then $T(n) = O(n^d \log n)$;
- if $d > log_b a$, then $T(n) = O(n^d)$.

Master Theorem for $f(n) = O(n^d)$

$T(n) = aT(n/b) + O(n^d)$, where $a \geq 1$, $b > 1$ and $d \geq 0$,

- if $d < log_b a$ then $T(n) = O(n^{\log_b a})$;
- if $d = log_b a$, then $T(n) = O(n^d \log n)$;
- if $d > log_b a$, then $T(n) = O(n^d)$.

Recall that $af(n/b) = af(n)/b^d$. Then

- $d < log_b a \Leftrightarrow b^d < b^{\log_b a} = a \Leftrightarrow a/b^d > 1$;
- $d = log_b a \Leftrightarrow b^d = b^{\log_b a} = a \Leftrightarrow a/b^d = 1$;
- $d > log_b a \Leftrightarrow b^d > b^{\log_b a} = a \Leftrightarrow a/b^d < 1$.

Examples

$$T(n) = T(3n/4) + n$$

Examples

$T(n) = T(3n/4) + n$

- $a = 1$, $b = 4/3$, $d = 1$

Examples

$T(n) = T(3n/4) + n$

- $a = 1$, $b = 4/3$, $d = 1$
- $d = 1 > 0 = \log_{4/3} 1 = \log_b a \Rightarrow T(n) = O(n)$

Examples

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

Examples

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

- $a = 3$, $b = 2$, $d = 1$

Examples

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

- $a = 3$, $b = 2$, $d = 1$
- $d = 1 < \log_2 3 = \log_b a \Rightarrow T(n) = O(n^{\log_2 3})$

Examples

$$T(n) = 4T(n/2) + n \log n$$

Examples

$T(n) = 4T(n/2) + n \log n$

- $a = 4$, $b = 2$

Examples

$T(n) = 4T(n/2) + n \log n$

- $a = 4,\ b = 2$
- $f(n) = n \log n$.

Examples

$T(n) = 4T(n/2) + n \log n$

- $a = 4$, $b = 2$
- $f(n) = n \log n$.
- $af(n/b) = 4(n/2)\log(n/2) = 2n \log n - 2n$.

Examples

$T(n) = 4T(n/2) + n \log n$

- $a = 4,\ b = 2$
- $f(n) = n \log n$.
- $af(n/b) = 4(n/2)\log(n/2) = 2n \log n - 2n$.
- $2f(n) > af(n/b) > 1.9f(n)$ for **sufficient large** n.

Examples

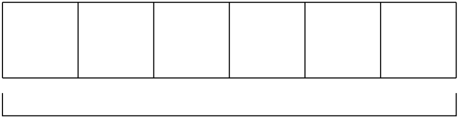
$T(n) = 4T(n/2) + n \log n$

- $a = 4,\ b = 2$
- $f(n) = n \log n$.
- $af(n/b) = 4(n/2)\log(n/2) = 2n\log n - 2n$.
- $2f(n) > af(n/b) > 1.9f(n)$ for **sufficient large** n.
- $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

Find The Lost Number

Find The Lost Number

Problem: Given $2^d - 1$ distinct numbers from $[0, 1, \ldots, 2^d - 1]$, find the lost one.



d bits

2^d numbers represented by d bits

Find The Lost Number

1	1	1

1	1	0

1	0	0

0	1	1

0	1	0

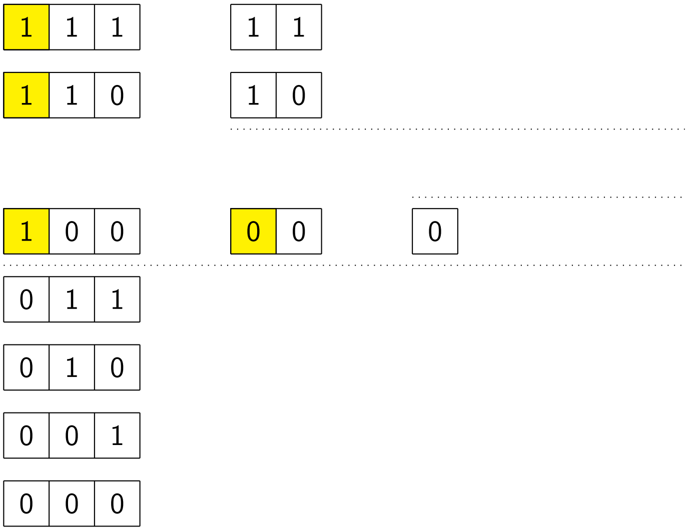
0	0	1

0	0	0

Find The Lost Number

1	1	1		1	1
1	1	0		1	0
1	0	0		0	0
0	1	1			
0	1	0			
0	0	1			
0	0	0			

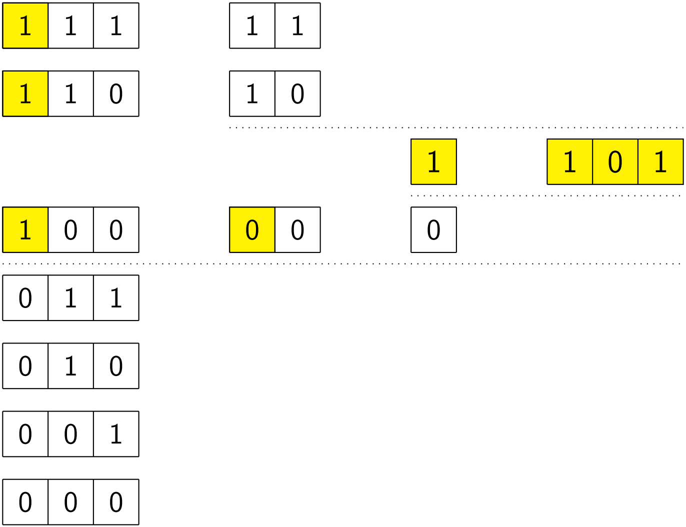
Find The Lost Number



Find The Lost Number

1 1 1	1 1	
1 1 0	1 0	
		1
1 0 0	0 0	0
0 1 1		
0 1 0		
0 0 1		
0 0 0		

Find The Lost Number



Find The Lost Number: $O(n)$

$$n = 2^d - 1$$

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

Matrix Multiplication

Matrix Multiplication

A $n \times m$ matrix

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,m-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{0,m-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,m-1} \end{bmatrix}$$

When $n = m$, A is called a square matrix. For two $n \times n$ matrices A and B, the multiplication is defined as $C = A \times B$, where C is also a $n \times n$ matrix, and

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

Matrix Multiplication

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$

Matrix Multiplication

$$\begin{array}{|c|c|c|}\hline a_{0,0} & a_{0,1} & a_{0,2} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} \\ \hline\end{array} \qquad \begin{array}{|c|c|c|}\hline b_{0,0} & b_{0,1} & b_{0,2} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} \\ \hline\end{array}$$

$$\begin{array}{|c|c|c|}\hline c_{0,0} & \phantom{c_{0,0}} & \phantom{c_{0,0}} \\ \hline & & \\ \hline & & \\ \hline\end{array} \qquad c_{0,0} = \sum_{k=0}^{2} a_{0,k} b_{k,0}$$

Matrix Multiplication

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$

$c_{0,0}$	$c_{0,1}$	

$$c_{0,0} = \sum_{k=0}^{2} a_{0,k} b_{k,0}$$

$$c_{0,1} = \sum_{k=0}^{2} a_{0,k} b_{k,1}$$

Matrix Multiplication

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$

$c_{0,0}$	$c_{0,1}$	$c_{0,2}$
$c_{1,0}$	$c_{1,1}$	$c_{1,2}$
$c_{2,0}$	$c_{2,1}$	$c_{2,2}$

$$c_{0,0} = \sum_{k=0}^{2} a_{0,k} b_{k,0}$$

$$c_{0,1} = \sum_{k=0}^{2} a_{0,k} b_{k,1}$$

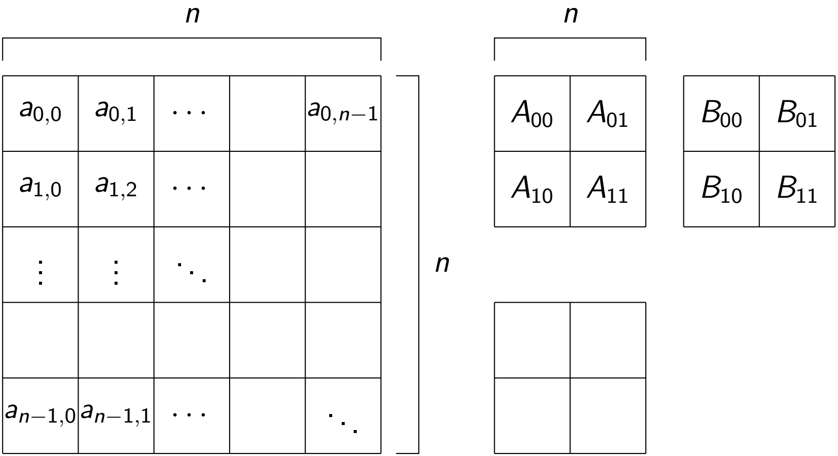
$$c_{2,2} = \sum_{k=0}^{2} a_{2,k} b_{k,2}$$

Follow The Definition: $O(n^3)$

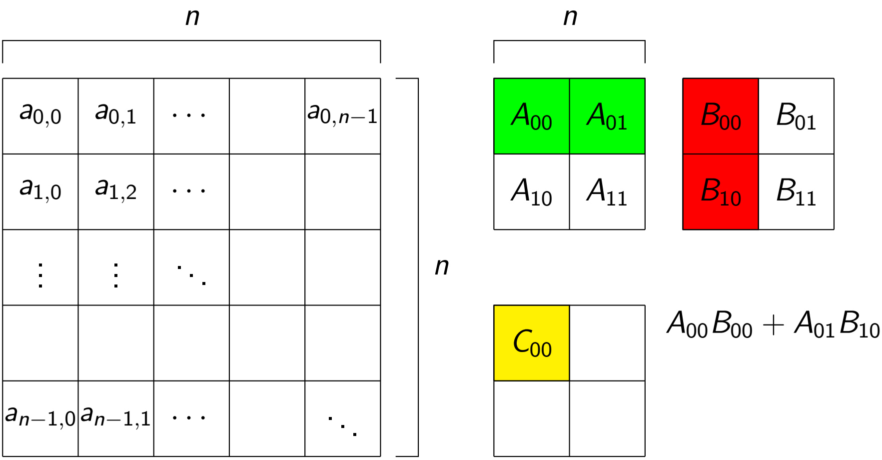
Algorithm: MatrixMul(A, B)

$C = \mathbf{0}^{n \times n}$;
for $i = 0$ *to* $n - 1$ **do**
    for $j = 0$ *to* $n - 1$ **do**
        for $k = 0$ *to* $n - 1$ **do**
            $c_{i,j} = c_{i,j} + a_{i,k} b_{k,j}$;
        end
    end
end
Return C;

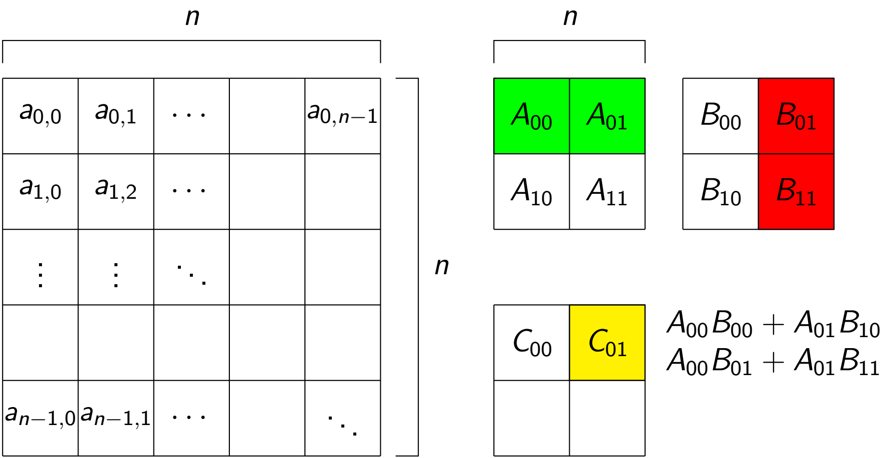
Simple Divide and Conquer Algorithm



Simple Divide and Conquer Algorithm



Simple Divide and Conquer Algorithm



Simple Divide and Conquer Algorithm

$$
\overbrace{\begin{array}{|c|c|c|c|c|}
\hline
a_{0,0} & a_{0,1} & \cdots & & a_{0,n-1} \\
\hline
a_{1,0} & a_{1,2} & \cdots & & \\
\hline
\vdots & \vdots & \ddots & & \\
\hline
 & & & & \\
\hline
a_{n-1,0} & a_{n-1,1} & \cdots & & \ddots \\
\hline
\end{array}}^{n}
\left.\vphantom{\begin{array}{c} a \\ a \\ a \\ a \\ a \end{array}}\right\} n
$$

$$
\overbrace{\begin{array}{|c|c|}
\hline
A_{00} & A_{01} \\
\hline
A_{10} & A_{11} \\
\hline
\end{array}}^{n}
\qquad
\begin{array}{|c|c|}
\hline
B_{00} & B_{01} \\
\hline
B_{10} & B_{11} \\
\hline
\end{array}
$$

$$
\begin{array}{|c|c|}
\hline
C_{00} & C_{01} \\
\hline
C_{10} & \\
\hline
\end{array}
\qquad
\begin{array}{l}
A_{00}B_{00} + A_{01}B_{10} \\
A_{00}B_{01} + A_{01}B_{11} \\
A_{10}B_{00} + A_{11}B_{10}
\end{array}
$$

Simple Divide and Conquer Algorithm

$$
\overbrace{\left.\begin{array}{|c|c|c|c|c|}
\hline
a_{0,0} & a_{0,1} & \cdots & & a_{0,n-1} \\
\hline
a_{1,0} & a_{1,2} & \cdots & & \\
\hline
\vdots & \vdots & \ddots & & \\
\hline
 & & & & \\
\hline
a_{n-1,0} & a_{n-1,1} & \cdots & & \ddots \\
\hline
\end{array}\right]}^{n} \; n
$$

$$
\overbrace{\begin{array}{|c|c|}
\hline
A_{00} & A_{01} \\
\hline
A_{10} & A_{11} \\
\hline
\end{array}}^{n}
\qquad
\begin{array}{|c|c|}
\hline
B_{00} & B_{01} \\
\hline
B_{10} & B_{11} \\
\hline
\end{array}
$$

$$
\begin{array}{|c|c|}
\hline
C_{00} & C_{01} \\
\hline
C_{10} & C_{11} \\
\hline
\end{array}
\qquad
\begin{array}{l}
A_{00}B_{00} + A_{01}B_{10} \\
A_{00}B_{01} + A_{01}B_{11} \\
A_{10}B_{00} + A_{11}B_{10} \\
A_{10}B_{01} + A_{11}B_{11}
\end{array}
$$

Simple Divide and Conquer Algorithm: $O(n^3)$

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

Algorithm: MatrixMul2(A, B)

$C = \mathbf{0}^{n \times n}$;
for $i, j = 0, 1$ **do**
    $M_0 = MatrixMul2(A_{i0}, B_{0j})$;
    $M_1 = MatrixMul2(A_{i1}, B_{1j})$;
    $C_{ij} = M_0 + M_1$;
end
Return C;

Simple Divide and Conquer Algorithm: $O(n^3)$

- $T(n) = 8T(n/2) + n^2$
- $a = 8$
- $b = 2$
- $d = 2$

Algorithm: MatrixMul2(A, B)

$C = \mathbf{0}^{n \times n}$;
for $i, j = 0, 1$ **do**
    $M_0 = MatrixMul2(A_{i0}, B_{0j})$;
    $M_1 = MatrixMul2(A_{i1}, B_{1j})$;
    $C_{ij} = M_0 + M_1$;
end
Return C;

Simple Divide and Conquer Algorithm: $O(n^3)$

- $T(n) = 8T(n/2) + n^2$
- $a = 8$
- $b = 2$
- $d = 2$
- $\log_b a = 3 > d$

Algorithm: MatrixMul2(A, B)

$C = \mathbf{0}^{n \times n}$;
for $i, j = 0, 1$ **do**
  $M_0 = MatrixMul2(A_{i0}, B_{0j})$;
  $M_1 = MatrixMul2(A_{i1}, B_{1j})$;
  $C_{ij} = M_0 + M_1$;
end
Return C;

Simple Divide and Conquer Algorithm: $O(n^3)$

- $T(n) = 8T(n/2) + n^2$
- $a = 8$
- $b = 2$
- $d = 2$
- $\log_b a = 3 > d$
- $T(n) = O(n^{\log_b a})$

Algorithm: MatrixMul2(A, B)

$C = \mathbf{0}^{n\times n}$;
for $i, j = 0, 1$ **do**
  $M_0 = MatrixMul2(A_{i0}, B_{0j})$;
  $M_1 = MatrixMul2(A_{i1}, B_{1j})$;
  $C_{ij} = M_0 + M_1$;
end
Return C;

Better Divide and Conquer Algorithm

$T(n) = 8T(n/2) + n^2$
$\Rightarrow T(n) = O(n^3)$

What about

- $T(n) = \color{red}{7}T(n/2) + n^2$

Better Divide and Conquer Algorithm

$T(n) = 8T(n/2) + n^2$
$\Rightarrow T(n) = O(n^3)$

What about

- $T(n) = \textcolor{red}{7}T(n/2) + n^2$
- $a = 7$
- $b = 2$
- $d = 2$

Better Divide and Conquer Algorithm

$$T(n) = 8T(n/2) + n^2$$
$$\Rightarrow T(n) = O(n^3)$$

What about

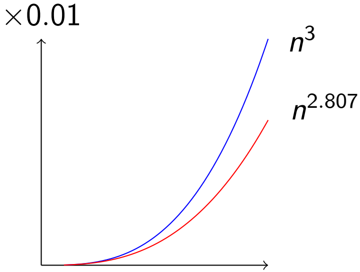
- $T(n) = 7T(n/2) + n^2$
- $a = 7$
- $b = 2$
- $d = 2$
- $\log_b a \approx 2.807 > d$

Better Divide and Conquer Algorithm

$T(n) = 8T(n/2) + n^2$
$\Rightarrow T(n) = O(n^3)$

What about

- $T(n) = 7T(n/2) + n^2$
- $a = 7$
- $b = 2$
- $d = 2$
- $\log_b a \approx 2.807 > d$
- $T(n) = O(n^{\log_b a}) = O(n^{2.807})$



Better Divide and Conquer Algorithm

$$\begin{aligned}
C_{00} &= A_{00}B_{00} + A_{01}B_{10} \\
C_{01} &= A_{00}B_{01} + A_{01}B_{11} \\
C_{10} &= A_{10}B_{00} + A_{11}B_{10} \\
C_{11} &= A_{10}B_{01} + A_{11}B_{11}
\end{aligned}$$

- 8 multiplications; each cost $O(n^3)$ operations
- 4 additions; each cost $O(n^2)$ operations

Better Divide and Conquer Algorithm

$$
\begin{aligned}
S_1 &= B_{01} - B_{11} & P_1 = A_{00}S_1 &= A_{00}B_{01} - A_{00}B_{11} \\
S_2 &= A_{00} + A_{01} & P_2 = S_2B_{11} &= A_{00}B_{11} + A_{01}B_{11} \\
S_3 &= A_{10} + A_{11} & P_3 = S_3B_{00} &= A_{10}B_{00} + A_{11}B_{00} \\
S_4 &= B_{10} - B_{00} & P_4 = A_{11}S_4 &= A_{11}B_{10} - A_{11}B_{00} \\
S_5 &= A_{00} + A_{11} & P_5 = S_5S_6 &= A_{00}B_{00} + A_{00}B_{11} + \\
S_6 &= B_{00} + B_{11} & & \quad A_{11}B_{00} + A_{11}B_{11} \\
S_7 &= A_{01} - A_{11} & P_6 = S_7S_8 &= A_{01}B_{10} + A_{01}B_{11} - \\
S_8 &= B_{10} + B_{11} & & \quad A_{11}B_{10} - A_{11}B_{11} \\
S_9 &= A_{00} - A_{10} & P_7 = S_9S_{10} &= A_{00}B_{00} + A_{00}B_{01} - \\
S_{10} &= B_{00} + B_{01} & & \quad A_{10}B_{00} - A_{10}B_{01}
\end{aligned}
$$

Better Divide and Conquer Algorithm: $O(n^{2.807})$

$$
\begin{aligned}
C_{00} &= P_5 + P_4 - P_2 + P_6 \\
C_{01} &= P_1 + P_2 \\
C_{10} &= P_3 + P_4 \\
C_{11} &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

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

- S: 10 additions
- P: 7 multiplications
- C: 8 additions

Better Divide and Conquer Algorithm: $O(n^{2.807})$

$$
\begin{aligned}
C_{00} &= P_5 + P_4 - P_2 + P_6 \\
C_{01} &= P_1 + P_2 \\
C_{10} &= P_3 + P_4 \\
C_{11} &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

- $T(n) = 7T(n/2) + O(n^2)$
- $a = 7$
- $b = 2$
- $d = 2$

- S: 10 additions
- P: 7 multiplications
- C: 8 additions

Better Divide and Conquer Algorithm: $O(n^{2.807})$

$$
\begin{aligned}
C_{00} &= P_5 + P_4 - P_2 + P_6 \\
C_{01} &= P_1 + P_2 \\
C_{10} &= P_3 + P_4 \\
C_{11} &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

- $T(n) = 7T(n/2) + O(n^2)$
- $a = 7$
- $b = 2$
- $d = 2$
- $\log_b a = 2.807$
- $T(n) = n^{2.807}$

- S: 10 additions
- P: 7 multiplications
- C: 8 additions

Fibonacci Number

The n-th Fibonacci number is defined by

$$F_n = \begin{cases} F_{n-1} + F_{n-2}, & n \geq 2 \\ 1, & n = 1 \\ 0, & n = 0 \end{cases}$$

$$\begin{bmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

Find The Medians

Median of An Array

Problem: Given an array $A = [a_0, a_1, \cdots, a_{n-1}]$ of n numbers, find the $(\lfloor n/2 \rfloor)$-th smallest element.

Problem: Given an array $A = [a_0, a_1, \cdots, a_{n-1}]$ of n numbers and $1 \le i \le n$, find the i-th smallest element.

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 6$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 6$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 1$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 1$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 3$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 3$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 5$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$i = 5$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$

-5	-1	1	2	3	7	$n = 6$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$
-5	-1	1	2	3	7	$i = 3$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$

-5	-1	1	2	3	7	$i = 3$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$

-5	-1	1	2	3	7	$i = 5$

Find The i-th Smallest Element

3	-5	7	2	-1	1	$n = 6$

-5	-1	1	2	**3**	7	$i = 5$

Sort at First: $O(n \log n)$

Finding the i-th smallest element is trivial if A is sorted.

- Sort A in the increasing order;
- Return the i-th number.

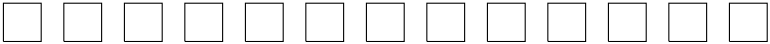
Special Cases

- The smallest element ($i = 1$) can be found in $O(n)$ comparisons.
- The largest element ($i = n$) can be found in $O(n)$ comparisons.

There is a lower bound on the number of necessary comparisons to find the i-th smallest element: $\Omega(n)$.

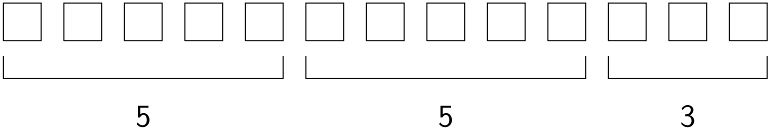
What is the upper bound?

Try with Divide and Conquer



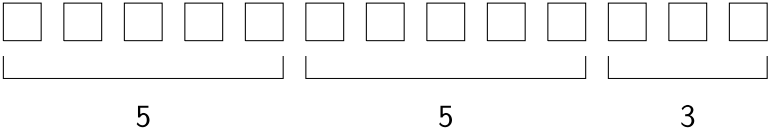
find the i-th smallest one of n distinct elements

Try with Divide and Conquer

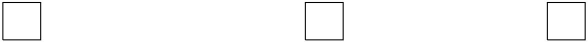


find the i-th smallest one of n distinct elements

Try with Divide and Conquer

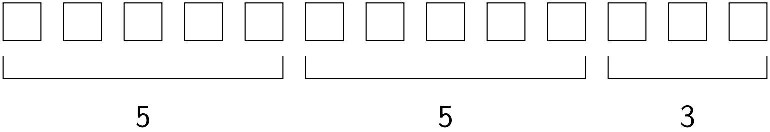


find the medians of subarrays

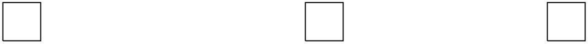


find the i-th smallest one of n distinct elements

Try with Divide and Conquer



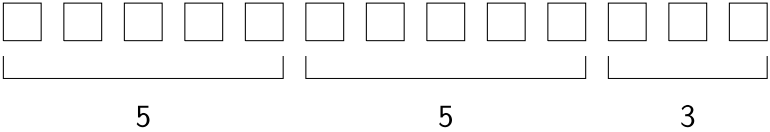
find the medians of subarrays



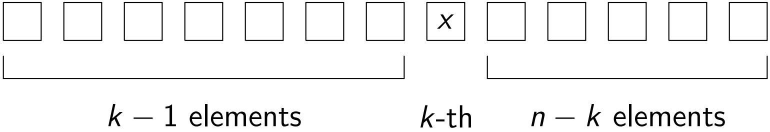
find the median of medians

find the i-th smallest one of n distinct elements

Try with Divide and Conquer



divide the elements



find the i-th smallest one of n distinct elements

Try with Divide and Conquer

5 5 3

divide the elements

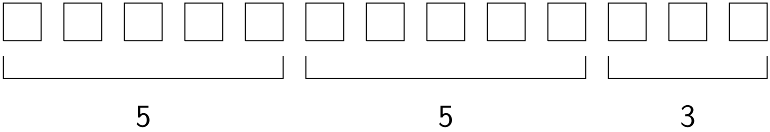
x

$k-1$ elements k-th $n-k$ elements

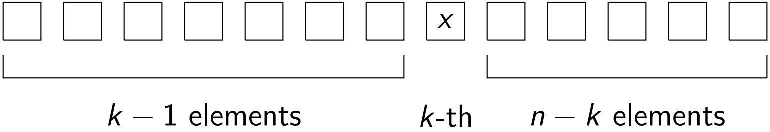
$i = k$

find the i-th smallest one of n distinct elements

Try with Divide and Conquer



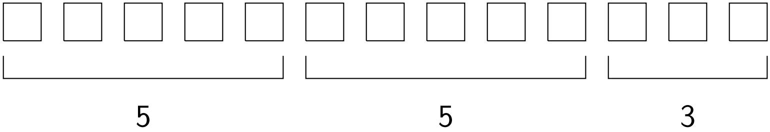
divide the elements



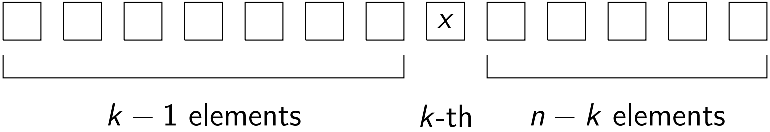
$i < k$

find the i-th smallest one of n distinct elements

Try with Divide and Conquer



divide the elements



$i > k$

find the i-th smallest one of n distinct elements

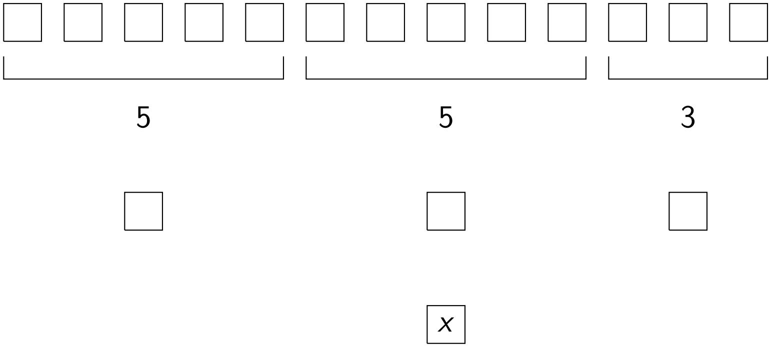
Find The i-th Smallest Element

Algorithm: Median(A, i)

if $|A| \leq 5$ **then** **return** The i-th smallest element of A ;
Divide A into groups of size 5;
$M =$ medians of the groups;
$x =$Median(M, $\lceil |M|/2 \rceil$);
$B =$ elements in A smaller than x;
$C =$ elements in A larger than x;
$k = |B| + 1$;
if $i < k$ **then** **return** Median(B, i) ;
else if $i > k$ **then** **return** Median(C, $i - k$) ;
else **return** x ;

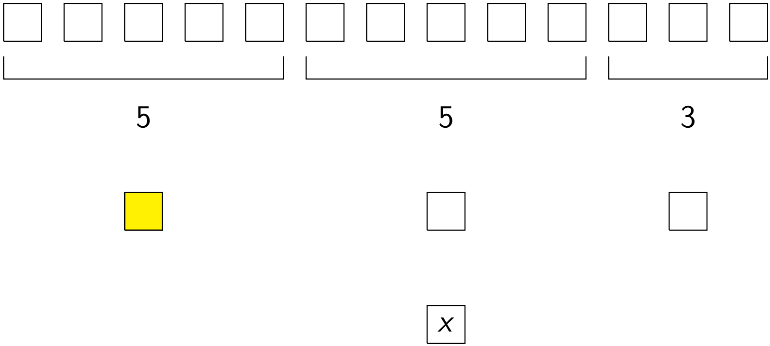
$$T(n) \leq T(max(k-1, n-k)) + T(\lceil n/5 \rceil) + O(n)$$

Efficiency of The Pivot



$$max(k-1, n-k) \leq 7n/10$$

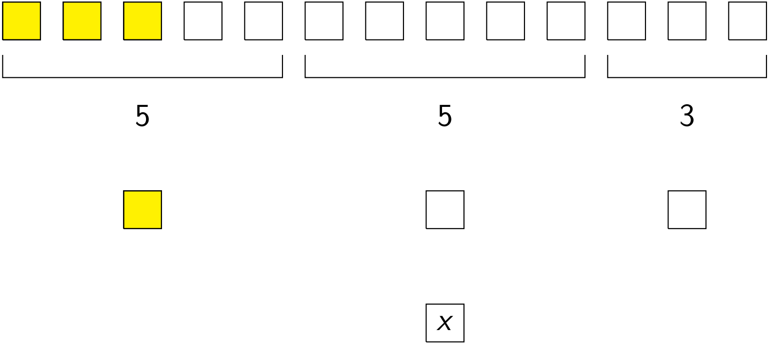
Efficiency of The Pivot



at least half of the group medians $\leq x$: $n/10$

$max(k-1, n-k) \leq 7n/10$

Efficiency of The Pivot



at least half of the group medians $\leq x$: $n/10$

3 numbers inside each group $\leq$ the median: $3n/10$

$max(k-1, n-k) \leq 7n/10$

Find The i-th Smallest Element: $O(n)$

$$T(n) \leq T(7n/10) + T(n/5) + O(n)$$

If $T(n) \leq T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

Find The i-th Smallest Element: $O(n)$

$$T(n) \le T(7n/10) + T(n/5) + O(n)$$

If $T(n) \le T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

- Induction on n: $T(n) \le C \cdot n$ for some $C > 1$.

Find The i-th Smallest Element: $O(n)$

$$T(n) \le T(7n/10) + T(n/5) + O(n)$$

If $T(n) \le T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

- Induction on n: $T(n) \le C \cdot n$ for some $C > 1$.
- Base case $n = 1$: $T(1) = 1 < C$.

Find The i-th Smallest Element: $O(n)$

$$T(n) \leq T(7n/10) + T(n/5) + O(n)$$

If $T(n) \leq T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

- Induction on n: $T(n) \leq C \cdot n$ for some $C > 1$.
- Base case $n = 1$: $T(1) = 1 < C$.
- Assume $T(k) \leq C \cdot k$ for all $k < n$.

Find The i-th Smallest Element: $O(n)$

$$T(n) \leq T(7n/10) + T(n/5) + O(n)$$

If $T(n) \leq T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

- Induction on n: $T(n) \leq C \cdot n$ for some $C > 1$.
- Base case $n = 1$: $T(1) = 1 < C$.
- Assume $T(k) \leq C \cdot k$ for all $k < n$.
- $T(n) \leq C \cdot a \cdot n + C \cdot b \cdot n + c \cdot n = (C(a + b) + c) \cdot n$.

Find The i-th Smallest Element: $O(n)$

$$T(n) \le T(7n/10) + T(n/5) + O(n)$$

If $T(n) \le T(a \cdot n) + T(b \cdot n) + c \cdot n$ and $a + b < 1$, then $T(n) = O(n)$.

- Induction on n: $T(n) \le C \cdot n$ for some $C > 1$.
- Base case $n = 1$: $T(1) = 1 < C$.
- Assume $T(k) \le C \cdot k$ for all $k < n$.
- $T(n) \le C \cdot a \cdot n + C \cdot b \cdot n + c \cdot n = (C(a+b) + c) \cdot n$.
- $C(a+b) + c \le C$ as long as $C \ge c/(1-a-b)$.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.
- half of the medians $\leq$ than x: $n/6$.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.
- half of the medians $\leq$ than x: $n/6$.
- 2 numbers $\leq$ than the median: $n/3$.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.
- half of the medians $\leq$ than x: $n/6$.
- 2 numbers $\leq$ than the median: $n/3$.
- $T(n) \leq T(2n/3) + T(n/3) + O(n)$.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.
- half of the medians $\leq$ than x: $n/6$.
- 2 numbers $\leq$ than the median: $n/3$.
- $T(n) \leq T(2n/3) + T(n/3) + O(n)$.
- $T(n) = O(n \log n)$.

Divide The Elements into Groups of Size k

$k = 5 \Rightarrow O(n)$. What about $k = 3$?

- n elements are divided into $n/3$ groups, each of size 3.
- x be the median of the medians.
- half of the medians $\leq$ than x: $n/6$.
- 2 numbers $\leq$ than the median: $n/3$.
- $T(n) \leq T(2n/3) + T(n/3) + O(n)$.
- $T(n) = O(n \log n)$.
- Other k?

Random Pivot: $O(n)$ in Expectation

Algorithm: RandMedian(A, i)

if $|A|$ *is small* **then** **return** The i-th smallest element of A ;
$x =$ random element in A;
$B =$ elements in A smaller than x;
$C =$ elements in A larger than x;
$k = |B| + 1$;
if $i < k$ **then** **return** RandMedian(B, i) ;
else if $i > k$ **then** **return** RandMedian(C, $i - k$) ;
else **return** x ;

The Fast Fourier Transform

Value Representation of Polynomials

Fact: A degree-d polynomial

$$A(x) = a_0 + a_1 x + \cdots + a_d x^d$$

is uniquely characterized by its values at $d + 1$ distinct points.

Value Representation of Polynomials

Fact: A degree-d polynomial

$$A(x) = a_0 + a_1 x + \cdots + a_d x^d$$

is uniquely characterized by its values at $d + 1$ distinct points.

The polynomial given by the coefficients $a_0, a_1, \ldots, a_d$ can be alternatively represented by values

$$A(x_0), A(x_1), \ldots, A(x_d)$$

at points $x_0, x_1, \ldots, x_d$.

Value Representation of Polynomials

Why the value representations? Consider the multiplication of two polynomials: $C(x) = A(x) \times B(x)$.

Value Representation of Polynomials

Why the value representations? Consider the multiplication of two polynomials: $C(x) = A(x) \times B(x)$.

- If $A(x) = \sum_{i=0}^{d} a_i x^i$ and $B(x) = \sum_{i=0}^{d} b_i x^i$, then

$$C(x) = \sum_{i=0}^{2d} c_i x^i \text{ and } c_i = \sum_{k=0}^{i} a_k b_{i-k}$$

Value Representation of Polynomials

Why the value representations? Consider the multiplication of two polynomials: $C(x) = A(x) \times B(x)$.

- If $A(x) = \sum_{i=0}^{d} a_i x^i$ and $B(x) = \sum_{i=0}^{d} b_i x^i$, then

$$C(x) = \sum_{i=0}^{2d} c_i x^i \text{ and } c_i = \sum_{k=0}^{i} a_k b_{i-k}$$

- Given $A(x)$ as $A(x_0)$, $A(x_1)$, …, $A(x_{2d})$, and $B(x)$ is given by $B(x_0)$, $B(x_1)$, …, $B(x_{2d})$, then $C(x)$ can be represented by

$$C(x_0) = A(x_0)B(x_0), \ldots, C(x_{2d}) = A(x_{2d})B(x_{2d}).$$

Transform between Representations

$$a_0, a_1, \ldots, a_d \Rightarrow A(x_0), A(x_1), \ldots, A(x_d)$$

For each point x_i, calculate

$$A(x_i) = \sum_{j=0}^{d} a_i x_i^j$$

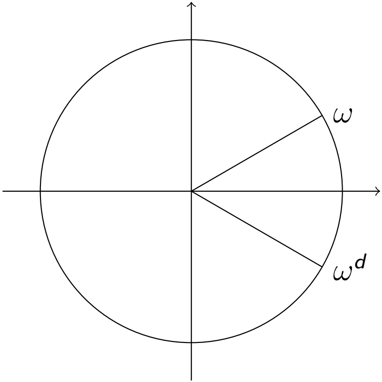
It requires $O(d \times d)$ operations.

Careful Selection of The Points

$x_0 = 1, x_1 = \omega, x_2 = \omega^2, \ldots, x_d = \omega^d$

ω: $(d+1)$-th (complex) root of 1

$\omega = e^{i\theta} = \cos\theta + i\sin\theta$
$\theta = 2\pi/(d+1)$



Divide The Calculations

For $x_i = \omega^i$, $i = 0, 1, \ldots, d$ with $d = 2k + 1$,

$$\sum_{j=0}^{d} a_i x_i^j \quad = \quad \sum_{j=0}^{k} a_{2j} x_i^{2j} + \sum_{j=0}^{k} a_{2j+1} x_i^{2j+1}$$

where $A_e(x) = \sum_{j=0}^{k} a_{2j} x^j$, $A_o(x) = \sum_{j=0}^{k} a_{2j+1} x^j$.

Divide The Calculations

For $x_i = \omega^i$, $i = 0, 1, \ldots, d$ with $d = 2k + 1$,

$$\begin{aligned} \sum_{j=0}^{d} a_i x_i^j &= \sum_{j=0}^{k} a_{2j} x_i^{2j} + \sum_{j=0}^{k} a_{2j+1} x_i^{2j+1} \\ &= \sum_{j=0}^{k} a_{2j} x_i^{2j} + x_i \sum_{j=0}^{k} a_{2j+1} x_i^{2j} \end{aligned}$$

where $A_e(x) = \sum_{j=0}^{k} a_{2j} x^j$, $A_o(x) = \sum_{j=0}^{k} a_{2j+1} x^j$.

Divide The Calculations

For $x_i = \omega^i$, $i = 0, 1, \ldots, d$ with $d = 2k+1$,

$$\begin{aligned}
\sum_{j=0}^{d} a_i x_i^j &= \sum_{j=0}^{k} a_{2j} x_i^{2j} + \sum_{j=0}^{k} a_{2j+1} x_i^{2j+1} \\
&= \sum_{j=0}^{k} a_{2j} x_i^{2j} + x_i \sum_{j=0}^{k} a_{2j+1} x_i^{2j} \\
&= A_e(x_i^2) + x_i A_o(x_i^2).
\end{aligned}$$

where $A_e(x) = \sum_{j=0}^{k} a_{2j} x^j$, $A_o(x) = \sum_{j=0}^{k} a_{2j+1} x^j$.

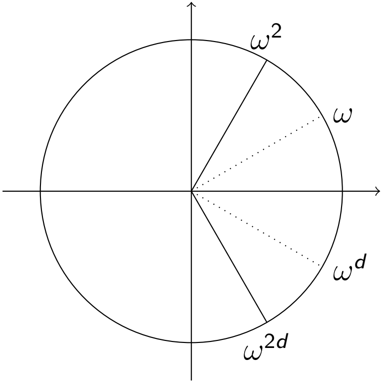
Careful Selection of The Points

$(k+1) = (d+1)/2$. Thus the $x_0, x_2, x_4, \ldots, x_{2k}$ are the $(k+1)$-th roots of 1.

$$x_0 = 1, x_1 = \omega, x_2 = \omega^2, \ldots, x_d = \omega^d$$

$\omega^d = e^{-i \cdot \theta}$

$\omega^{2d} = e^{-i \cdot 2\theta}$



The Fast Fourier Transform

- $a = [a_0, a_1, \ldots, a_d]$
- $d = 2k + 1$
- $(d+1)$-th roots: $1, \omega, \omega^2, \ldots, \omega^d$
- $(k+1)$-th roots: $1, \omega^2, \omega^4, \ldots, \omega^{2k}$

Algorithm: FFT(a, ω)

---

if $\omega = 1$ **then return** a_0;
$a_e = [a_0, a_2, \ldots, a_{2k}]$;
$a_0 = [a_1, a_3, \ldots, a_{2k+1}]$;
$re =$FFT(a_e, ω^2);
$ro =$FFT(a_o, ω^2);
for $i = 0$ *to* d **do**
    // $A(\omega_i) = \sum_{j=0}^{d} a_j \omega_i^j$;
    $r_i = re_i + \omega^i ro_i$;
end
return $[r_0, r_1, \ldots, r_d]$;

The Fast Fourier Transform

- $a = [a_0, a_1, \ldots, a_d]$
- $d = 2k + 1$
- $(d+1)$-th roots: $1, \omega, \omega^2, \ldots, \omega^d$
- $(k+1)$-th roots: $1, \omega^2, \omega^4, \ldots, \omega^{2k}$
- $T(d) = 2T(d/2) + O(d)$
- $T(d) = O(d \log d)$

Algorithm: FFT(a, ω)

if $\omega = 1$ **then return** a_0;
$a_e = [a_0, a_2, \ldots, a_{2k}]$;
$a_0 = [a_1, a_3, \ldots, a_{2k+1}]$;
$re =$FFT(a_e, ω^2);
$ro =$FFT(a_o, ω^2);
for $i = 0$ *to* d **do**
  // $A(\omega_i) = \sum_{j=0}^{d} a_j \omega_i^j$;
  $r_i = re_i + \omega^i ro_i$;
end
return $[r_0, r_1, \ldots, r_d]$;

Transform between The Representations

$$x_0 = 1, x_1 = \omega, x_2 = \omega^2, \ldots, x_d = \omega^d$$

$$\begin{bmatrix} A(1) \\ A(\omega) \\ \vdots \\ A(\omega^d) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^d \\ & & \vdots & & \\ 1 & \omega^d & \omega^{2d} & \cdots & \omega^{d^2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}$$

$$M(\omega)^{-1} = \frac{1}{n} M(\omega^{-1})$$

Multiplication of Polynomials

Given $A(x) = \sum_{i=0}^{d} a_i x^i$ and $B(x) = \sum_{i=0}^{d} b_i x^i$,

- calculate the value representation of $A(x)$ and $B(x)$: $O(d \log d)$
- calculate the value representation of $C(x) = A(x)B(x)$: $O(d)$
- calculate the c_i's for $C(x) = \sum_{i=0}^{d} c_i x^i$: $O(d \log d)$

In total, it costs $O(d \log d)$.

THANK YOU

中国科学院深圳先进技术研究院
SHENZHEN INSTITUTES OF ADVANCED TECHNOLOGY
CHINESE ACADEMY OF SCIENCES