

# **CSE 421**

## **Integer Multiplication, Median**

Shayan Oveis Gharan

# Master Theorem

Suppose  $T(n) = a T\left(\frac{n}{b}\right) + cn^k$  for all  $n > b$ . Then,

- If  $a > b^k$  then  $T(n) = \Theta(n^{\log_b a})$
- If  $a < b^k$  then  $T(n) = \Theta(n^k)$
- If  $a = b^k$  then  $T(n) = \Theta(n^k \log n)$

Works even if it is  $\left\lceil \frac{n}{b} \right\rceil$  instead of  $\frac{n}{b}$ .

We also need  $a \geq 1, b > 1, k \geq 0$  and  $T(n) = O(1)$  for  $n \leq b$ .

# Integer Multiplication



# How to use Divide and Conquer?

Suppose we want to multiply two 2-digit integers (32,45).

We can do this by multiplying four 1-digit integers

Then, use add/shift to obtain the result:

$$x = 10x_1 + x_0$$

$$y = 10y_1 + y_0$$

$$\begin{aligned} xy &= (10x_1 + x_0)(10y_1 + y_0) \\ &= 100x_1y_1 + 10(x_1y_0 + x_0y_1) + x_0y_0 \end{aligned}$$

4	5	$y_1 y_0$	
3	2	$x_1 x_0$	
<hr/>			
1	0	$x_0 \cdot y_0$	
0	8	$x_0 \cdot y_1$	
1	5	$x_1 \cdot y_0$	
1	2	$x_1 \cdot y_1$	
<hr/>			
1	4	4	0

Same idea works when multiplying n-digit integers:

- Divide into 4 n/2-digit integers.
- Recursively multiply
- Then merge solutions

# A Divide and Conquer for Integer Mult

Let  $x, y$  be two  $n$ -bit integers

Write  $x = 2^{n/2}x_1 + x_0$  and  $y = 2^{n/2}y_1 + y_0$

where  $x_0, x_1, y_0, y_1$  are all  $n/2$ -bit integers.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \\xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\&= 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0\end{aligned}$$

Therefore,

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

So,

$$T(n) = \Theta(n^2).$$

We only need 3 values  
 $x_1y_1, x_0y_0, x_1y_0 + x_0y_1$   
Can we find all 3 by only  
3 multiplication?

# Key Trick: 4 multiplies at the price of 3

$$x = 2^{n/2} \cdot x_1 + x_0$$

$$y = 2^{n/2} \cdot y_1 + y_0$$

$$xy = (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0)$$

$$= 2^n \cdot x_1y_1 + 2^{n/2} \cdot (x_1y_0 + x_0y_1) + x_0y_0$$

$$\alpha = x_1 + x_0$$

$$\beta = y_1 + y_0$$

$$\alpha\beta = (x_1 + x_0)(y_1 + y_0)$$

$$= x_1y_1 + (x_1y_0 + x_0y_1) + x_0y_0$$

$$(x_1y_0 + x_0y_1) = \alpha\beta - x_1y_1 - x_0y_0$$

# Key Trick: 4 multiplies at the price of 3

Theorem [Karatsuba-Ofman, 1962] Can multiply two n-digit integers in  $O(n^{1.585\dots})$  bit operations.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \Rightarrow \alpha = x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \Rightarrow \beta = y_1 + y_0 \\xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\&= \underbrace{2^n \cdot x_1 y_1}_A + \underbrace{2^{n/2} \cdot (x_1 y_0 + x_0 y_1)}_{\alpha\beta - A - B} + \underbrace{x_0 y_0}_B\end{aligned}$$

To multiply two n-bit integers:

Add two n/2 bit integers.

Multiply **three** n/2-bit integers.

Add, subtract, and shift n/2-bit integers to obtain result.

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(n^{\log_2 3}) = O(n^{1.585\dots})$$



# Integer Multiplication (Summary)

- Naïve:  $\Theta(n^2)$
- Karatsuba:  $\Theta(n^{1.585\dots})$
- **Amusing exercise**: generalize Karatsuba to do 5 size  $n/3$  subproblems  
This gives  $\Theta(n^{1.46\dots})$  time algorithm
- Best known algorithm runs in  $\Theta(n \log n \log \log n)$  using fast Fourier transform (section 5.5 of KT)  
but mostly unused in practice (unless you need really big numbers - a billion digits of  $\pi$ , say)
- Best lower bound  $O(n)$ : A fundamental open problem

Median

# Selecting k-th smallest

**Problem:** Given numbers  $x_1, \dots, x_n$  and an integer  $1 \leq k \leq n$  output the  $k$ -th smallest number

$$\text{Sel}(\{x_1, \dots, x_n\}, k)$$

**A simple algorithm:** Sort the numbers in time  $O(n \log n)$  then return the  $k$ -th smallest in the array.

Can we do better?

Yes, in time  $O(n)$  if  $k = 1$  or  $k = n$ .

Can we do  $O(n)$  for all possible values of  $k$ ?

Assume all numbers are distinct for simplicity.

# An Idea

Choose a number  $w$  from  $x_1, \dots, x_n$

Define

- $S_{<}(w) = \{x_i : x_i < w\}$
- $S_{=}(w) = \{x_i : x_i = w\}$
- $S_{>}(w) = \{x_i : x_i > w\}$

Can be computed in  
linear time

Solve the problem recursively as follows:

- If  $k \leq |S_{<}(w)|$ , output  $Sel(S_{<}(w), k)$
- Else if  $k \leq |S_{<}(w)| + |S_{=}(w)|$ , output  $w$
- Else output  $Sel(S_{>}(w), k - |S_{<}(w)| - |S_{=}(w)|)$

Ideally want  $|S_{<}(w)|, |S_{>}(w)| \leq n/2$ . In this case ALG runs in  $O(n) + O\left(\frac{n}{2}\right) + O\left(\frac{n}{4}\right) + \dots + O(1) = O(n)$ .

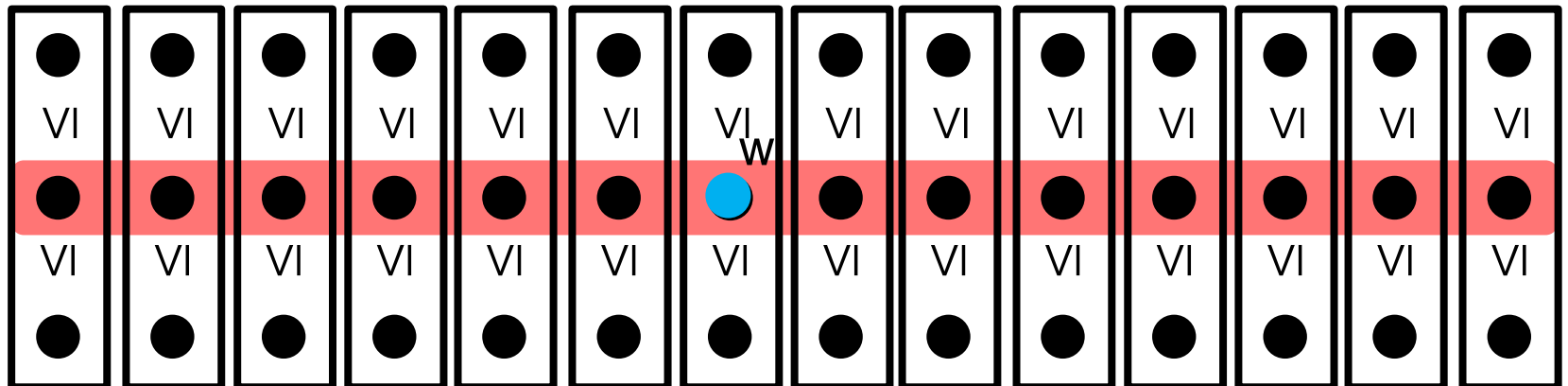
# How to choose $w$ ?

Suppose we choose  $w$  uniformly at random  
similar to the pivot in quicksort.

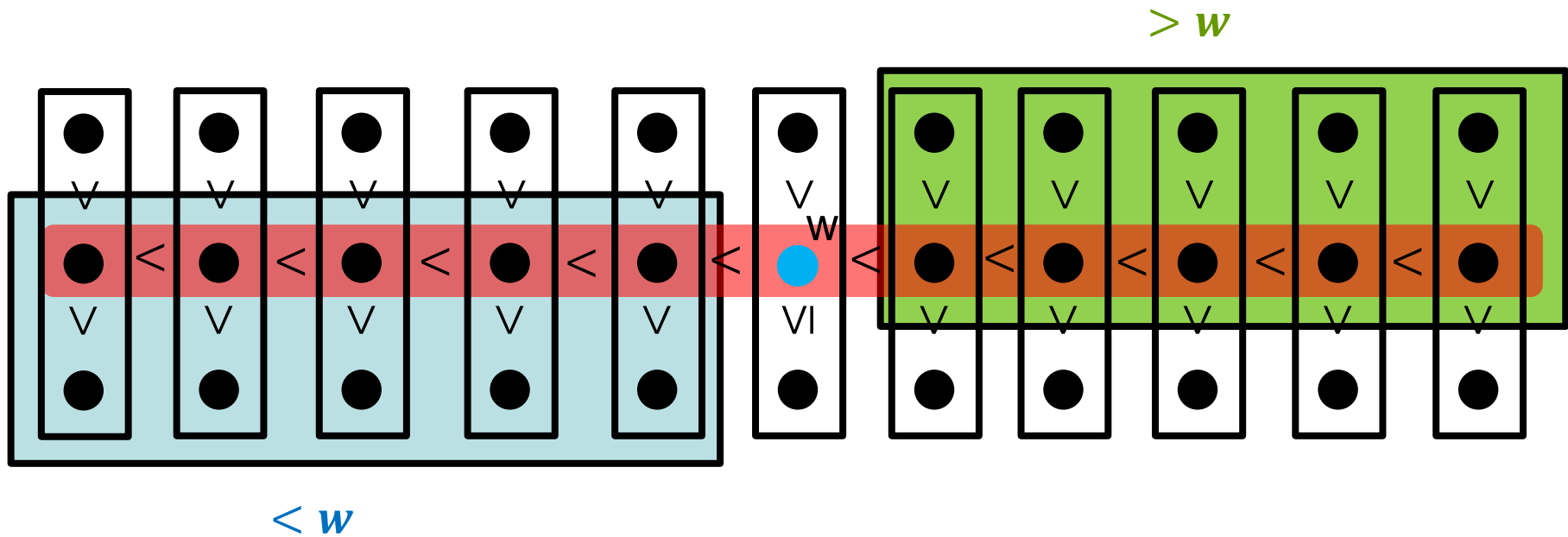
Then,  $\mathbb{E}[|S_{<}(w)|] = \mathbb{E}[|S_{>}(w)|] = n/2$ . Algorithm runs in  $O(n)$  in expectation.

Can we get  $O(n)$  running time deterministically?

- Partition numbers into sets of size 3.
- Sort each set (takes  $O(n)$ )
- $w = Sel(\textit{midpoints}, n/6)$



# How to lower bound $|S_{<}(w)|$ , $|S_{>}(w)|$ ?



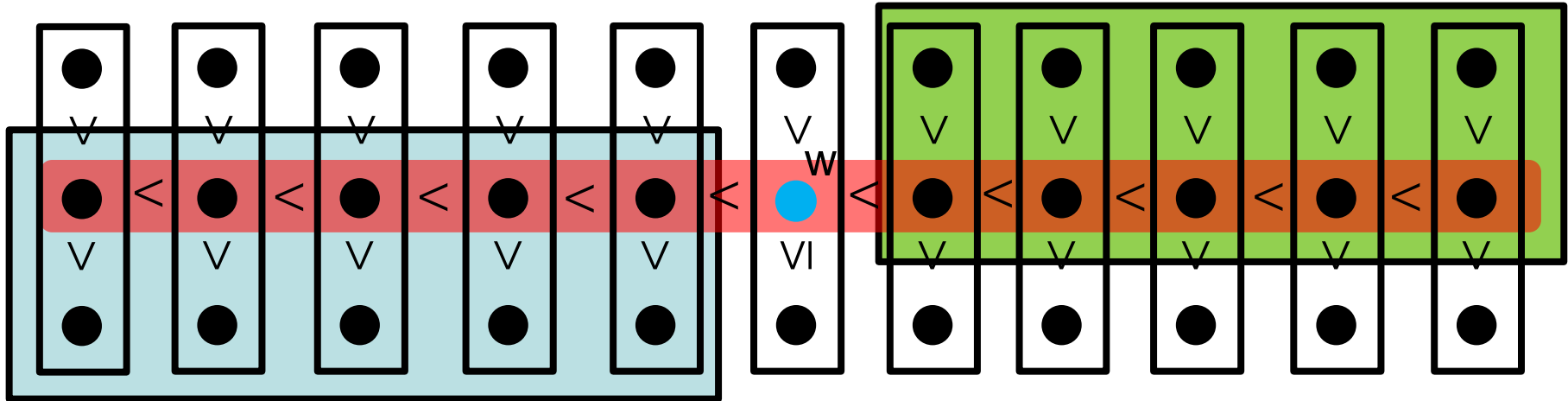
- $|S_{<}(w)| \geq 2 \binom{n}{6} = \frac{n}{3}$
- $|S_{>}(w)| \geq 2 \binom{n}{6} = \frac{n}{3}$ .



$$\frac{n}{3} \leq |S_{<}(w)|, |S_{>}(w)| \leq \frac{2n}{3}$$

So, what is the running time?

# Asymptotic Running Time?



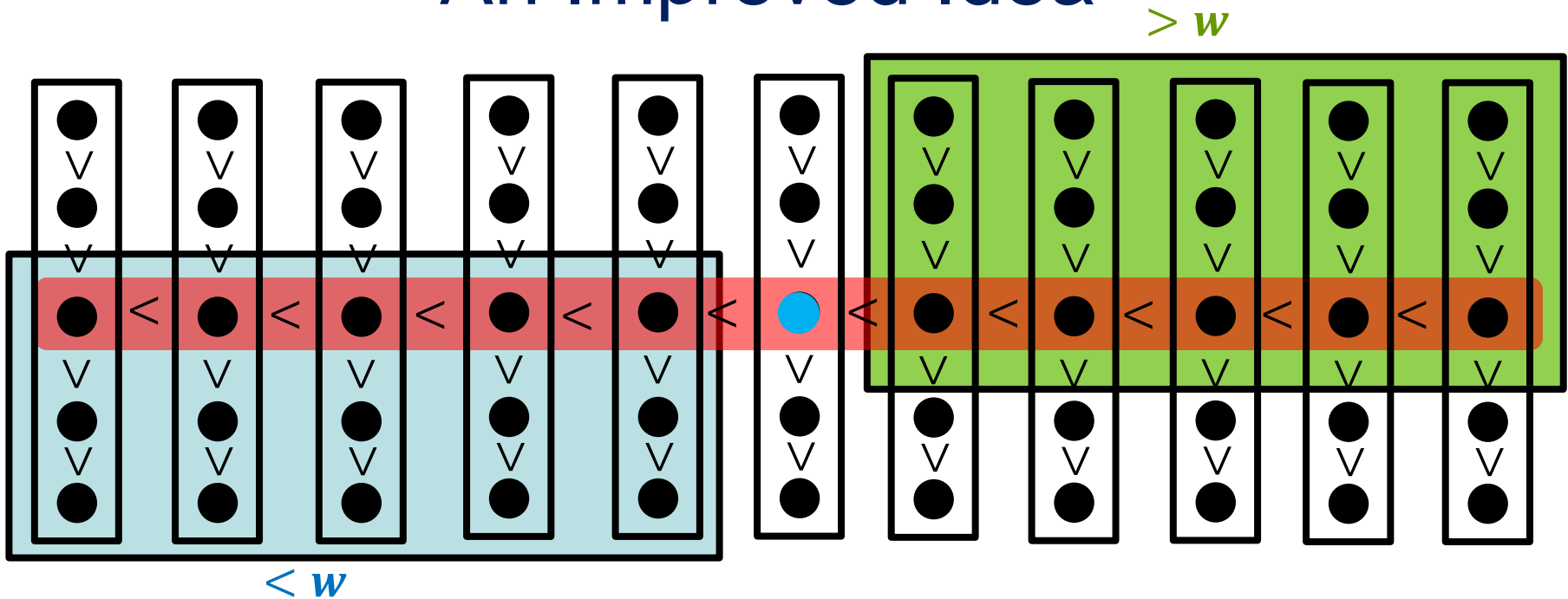
- If  $k \leq |S_{<}(w)|$ , output  $Sel(S_{<}(w), k)$
- Else if  $k \leq |S_{<}(w)| + |S_{=}(w)|$ , output  $w$
- Else output  $Sel(S_{>}(w), k - |S_{<}(w)| - |S_{=}(w)|)$

$O(n \log n)$  again?  
So, what is the point?

Where  $\frac{n}{3} \leq |S_{<}(w)|, |S_{>}(w)| \leq \frac{2n}{3}$

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) \Rightarrow T(n) = O(n \log n)$$

# An Improved Idea



Partition into  $n/5$  sets. Sort each set and set  $w = \text{Sel}(\text{midpoints}, n/10)$

- $|S_{<}(w)| \geq 3 \left(\frac{n}{10}\right) = \frac{3n}{10}$
  - $|S_{>}(w)| \geq 3 \left(\frac{n}{10}\right) = \frac{3n}{10}$
- ➔
- $\frac{n}{3} \leq |S_{<}(w)|, |S_{>}(w)| \leq \frac{2n}{3}$

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n) \Rightarrow T(n) = O(n)$$



# An Improved Idea

```

Sel(S, k) {
  n ← |S|
  If (n < ??) return ??
  Partition S into n/5 sets of size 5
  Sort each set of size 5 and let M be the set of medians, so
  |M|=n/5
  Let w=Sel(M,n/10)
  For i=1 to n{
    If  $x_i < w$  add x to  $S_{<}(w)$ 
    If  $x_i > w$  add x to  $S_{>}(w)$ 
    If  $x_i = w$  add x to  $S_{=}(w)$ 
  }
  If ( $k \leq |S_{<}(w)|$ )
    return Sel( $S_{<}(w), k$ )
  else if ( $k \leq |S_{<}(w)| + |S_{=}(w)|$ )
    return w;
  else
    return Sel( $S_{>}(w), k - |S_{<}(w)| - |S_{=}(w)|$ )
}

```

We can maintain each set in an array

# D&C Summary

## Idea:

“Two halves are better than a whole”

- if the base algorithm has super-linear complexity.

“If a little's good, then more's better”

- repeat above, recursively
- Applications: Many.
  - Binary Search, Merge Sort, (Quicksort),
  - Root of a Function
  - Closest points,
  - Integer multiplication
  - Median
  - Matrix Multiplication