

Chapter 2: Getting Started

Insertion Sort
Mergesort
Integer Multiplication
Matrix Multiplication

Insertion Sort	2
Loop Invariants	3
Mergesort	4
Divide and Conquer	5
Multiplying Big Integers	6
Recursive Version	7
Faster Multiplication	8
Matrix Multiplication	9
Strassen's Algorithm	10
Strassen's Algorithm Continued	11

Insertion Sort

```
INSERTION-SORT(A)
  for  $j \leftarrow 2$  to  $\text{length}(A)$  do
     $\text{key} \leftarrow A[j]$ 
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > \text{key}$  do
       $A[i + 1] \leftarrow A[i]$ 
       $i \leftarrow i - 1$ 
     $A[i + 1] \leftarrow \text{key}$ 
```

CS 5633 Analysis of Algorithms

Chapter 2: Slide - 2

Loop Invariants

- A loop invariant specifies what is true just before the loop condition is tested. The loop variable(s) should be used to describe the loop invariant.
- INSERTION-SORT outer loop, loop variable j
Loop Invariant: $A[1..j - 1]$ has been sorted.
- INSERTION-SORT inner loop, loop variable i
Loop Invariant: $\text{key} = \text{old value of } A[j]$,
 $A[i + 1..j - 1]$ has been shifted to $A[i + 2..j]$, and $\text{key} < \text{all elements in } A[i + 2..j]$

CS 5633 Analysis of Algorithms

Chapter 2: Slide - 3

Mergesort

```
MERGE-SORT(A, p, r)
  if  $p < r$  then
     $q \leftarrow \lfloor (p + r) / 2 \rfloor$ 
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

CS 5633 Analysis of Algorithms

Chapter 2: Slide - 4

Divide and Conquer

- Divide the problem into subproblems.
Divide n elements into 2 sets of $n/2$ elements.
- Conquer the subproblems by recursion.
Call MERGE-SORT on the 2 sequences.
- Combine the solutions to the subproblems.
Merge 2 sorted sequences into 1 sorted sequence.

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 5

Multiplying Big Integers

- ▷ BIT-MULTIPLY multiplies bit arrays A and B
- ▷ $+$ and $-$ denotes adding/subtracting bit arrays.
- ▷ It is assumed that n is a power of 2.
- ▷ Multiplying times 2^i is a bit shift.

```
BIT-MULTIPLY( $n, A, B$ )
   $C \leftarrow$  an array of  $2n$  zero bits
  for  $i \leftarrow 0$  to  $n - 1$  do
    if  $B[i] = 1$  then
       $C \leftarrow C + 2^i \cdot A$ 
  return  $C$ 
```

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 6

Recursive Version

```
BIT-MULTIPLY( $n, A, B$ )
   $C \leftarrow$  an array of  $2n$  zero bits
  if  $n = 1$  then
     $C[0] \leftarrow A[0] \wedge B[0]$ 
  return  $C$ 

Divide  $A$  and  $B$  into 4  $n/2$  bit arrays.
▷ E.g.,  $A = A_1 + 2^{n/2} \cdot A_2$ 
 $C \leftarrow C + \text{BIT-MULTIPLY}(n/2, A_1, B_1)$ 
 $C \leftarrow C + 2^{n/2} \cdot \text{BIT-MULTIPLY}(n/2, A_1, B_2)$ 
 $C \leftarrow C + 2^{n/2} \cdot \text{BIT-MULTIPLY}(n/2, A_2, B_1)$ 
 $C \leftarrow C + 2^n \cdot \text{BIT-MULTIPLY}(n/2, A_2, B_2)$ 
return  $C$ 
```

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 7

Faster Multiplication

```
BIT-MULTIPLY( $n, A, B$ )
   $C \leftarrow$  an array of  $2n$  zero bits
  if  $n = 1$  then  $C[0] \leftarrow A[0] \wedge B[0]$ 
  return  $C$ 

Divide  $A$  and  $B$  into 4  $n/2$  bit arrays.
▷ E.g.,  $A = A_1 + 2^{n/2} \cdot A_2$ 
 $P_1 \leftarrow \text{BIT-MULTIPLY}(n/2, A_1, B_1)$ 
 $C \leftarrow C + P_1$ 
 $P_2 \leftarrow \text{BIT-MULTIPLY}(n/2, A_2, B_2)$ 
 $C \leftarrow C + 2^n \cdot P_2$ 
 $P_3 \leftarrow \text{BIT-MULTIPLY}(n/2, A_1 - A_2, B_2 - B_1)$ 
 $C \leftarrow C + 2^{n/2} \cdot (P_3 - P_1 - P_2)$ 
return  $C$ 
```

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 8

Matrix Multiplication

▷ Multiply two $n \times n$ matrices

MATRIX-MULTIPLY(n, A, B)

$C \leftarrow$ an $n \times n$ matrix

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

$C[i, j] \leftarrow 0$

for $k \leftarrow 1$ **to** n **do**

$C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$

return C

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 9

Strassen's Algorithm Continued

$P_3 \leftarrow (A_3 + A_4) \cdot B_1$

$P_4 \leftarrow A_4 \cdot (B_3 - B_1)$

$C_3 \leftarrow P_3 + P_4$

$P_5 \leftarrow (A_1 + A_4) \cdot (B_1 + B_4)$

$P_6 \leftarrow (A_2 - A_4) \cdot (B_3 + B_4)$

$C_1 \leftarrow P_5 + P_4 - P_2 + P_6$

$P_7 \leftarrow (A_1 - A_3) \cdot (B_1 + B_2)$

$C_4 \leftarrow P_5 + P_1 - P_3 - P_7$

return C

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 11

Strassen's Algorithm

▷ Multiply two $n \times n$ matrices.

▷ It is assumed that n is a power of 2.

▷ The operation \cdot denotes a recursive call.

STRASSEN'S-ALGORITHM(n, A, B)

Do base case for $n = 1$, otherwise

Divide A and B into 8 $n/2 \times n/2$ matrices

▷ E.g., $A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$

$P_1 \leftarrow A_1 \cdot (B_2 - B_4)$

$P_2 \leftarrow (A_1 + A_2) \cdot B_4$

$C_2 \leftarrow P_1 + P_2$

CS 5633 Analysis of Algorithms

Chapter 2: Slide – 10