Divide & Conquer - 02

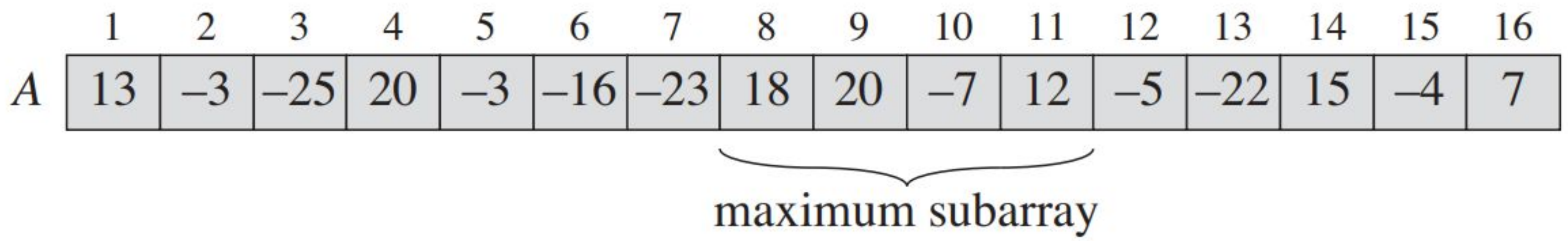
CS3026 – Analysis & Design of Algorithms

Angel Napa

UTEC UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Index

1 Maximum Subarray



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	13	–3	–25	20	–3	–16	–23	18	20	–7	12	–5	–22	15	–4	7

Figure 1: Cormen, Introduction to Algorithms

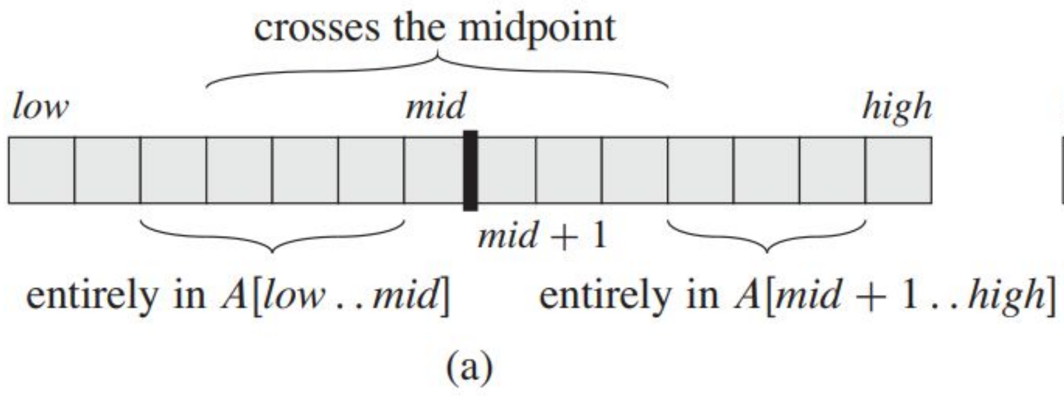
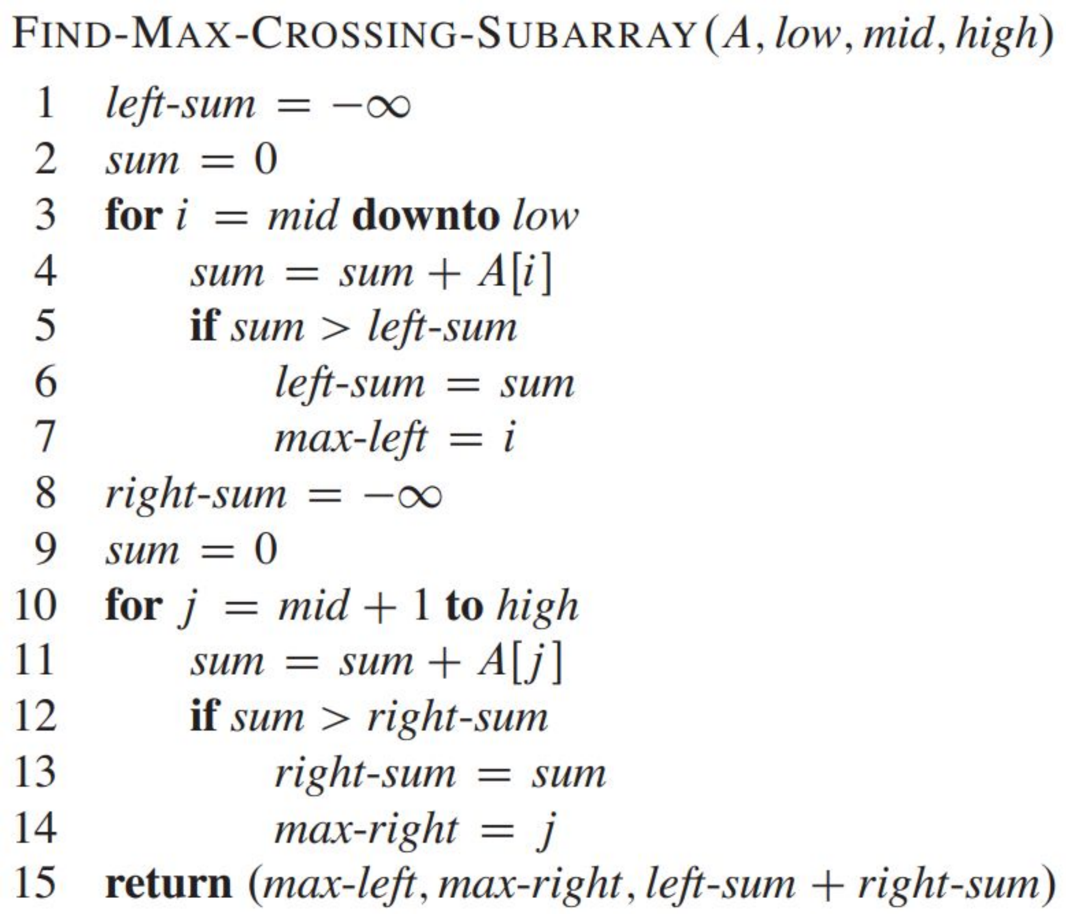
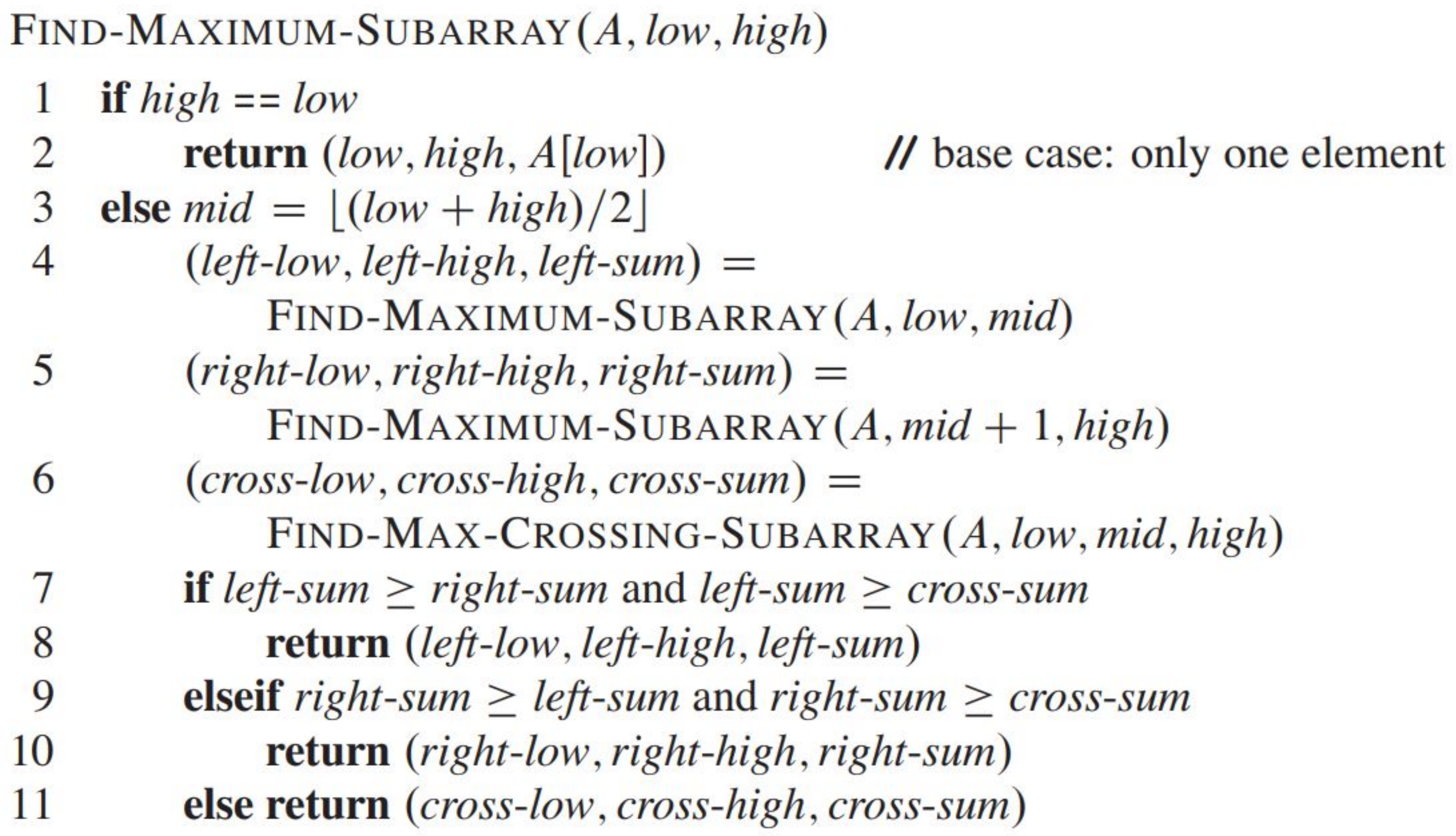


Figure 1: Cormen, Introduction to Algorithms



```
FIND-MAX-CROSSING-SUBARRAY($A$, $low$, $mid$, $high$)
left-sum = $-\infty$
sum = 0
for $i$ = mid downto low
    sum = sum + $A[i]$
    if sum > left-sum
        left-sum = sum
        max-left = $i$
right-sum = $-\infty$
sum = 0
for $j$ = mid + 1 to high
    sum = sum + $A[j]$
    if sum > right-sum
        right-sum = sum
        max-right = $j$
return (max-left, max-right, left-sum + right-sum)
```

Figure 1: Cormen, Introduction to Algorithms



```
FIND-MAXIMUM-SUBARRAY$(A, low, high)$
if $high == low$
    return $(low, high, A[low])$            // base case: only one element
else $mid = \lfloor (low + high)/2 \rfloor$
    $(left\text{-}low, left\text{-}high, left\text{-}sum) =$
        FIND-MAXIMUM-SUBARRAY$(A, low, mid)$
    $(right\text{-}low, right\text{-}high, right\text{-}sum) =$
        FIND-MAXIMUM-SUBARRAY$(A, mid + 1, high)$
    $(cross\text{-}low, cross\text{-}high, cross\text{-}sum) =$
        FIND-MAX-CROSSING-SUBARRAY$(A, low, mid, high)$
    if $left\text{-}sum \geq right\text{-}sum$ and $left\text{-}sum \geq cross\text{-}sum$
        return $(left\text{-}low, left\text{-}high, left\text{-}sum)$
    elseif $right\text{-}sum \geq left\text{-}sum$ and $right\text{-}sum \geq cross\text{-}sum$
        return $(right\text{-}low, right\text{-}high, right\text{-}sum)$
    else return $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$
```

Figure 1: Cormen, Introduction to Algorithms

2 Product of Natural Numbers

Problem

```
    9999     A
    7777     B
  ------
   69993     C
  69993      D
 69993       E
69993        F
--------
77762223     G
```

Problem

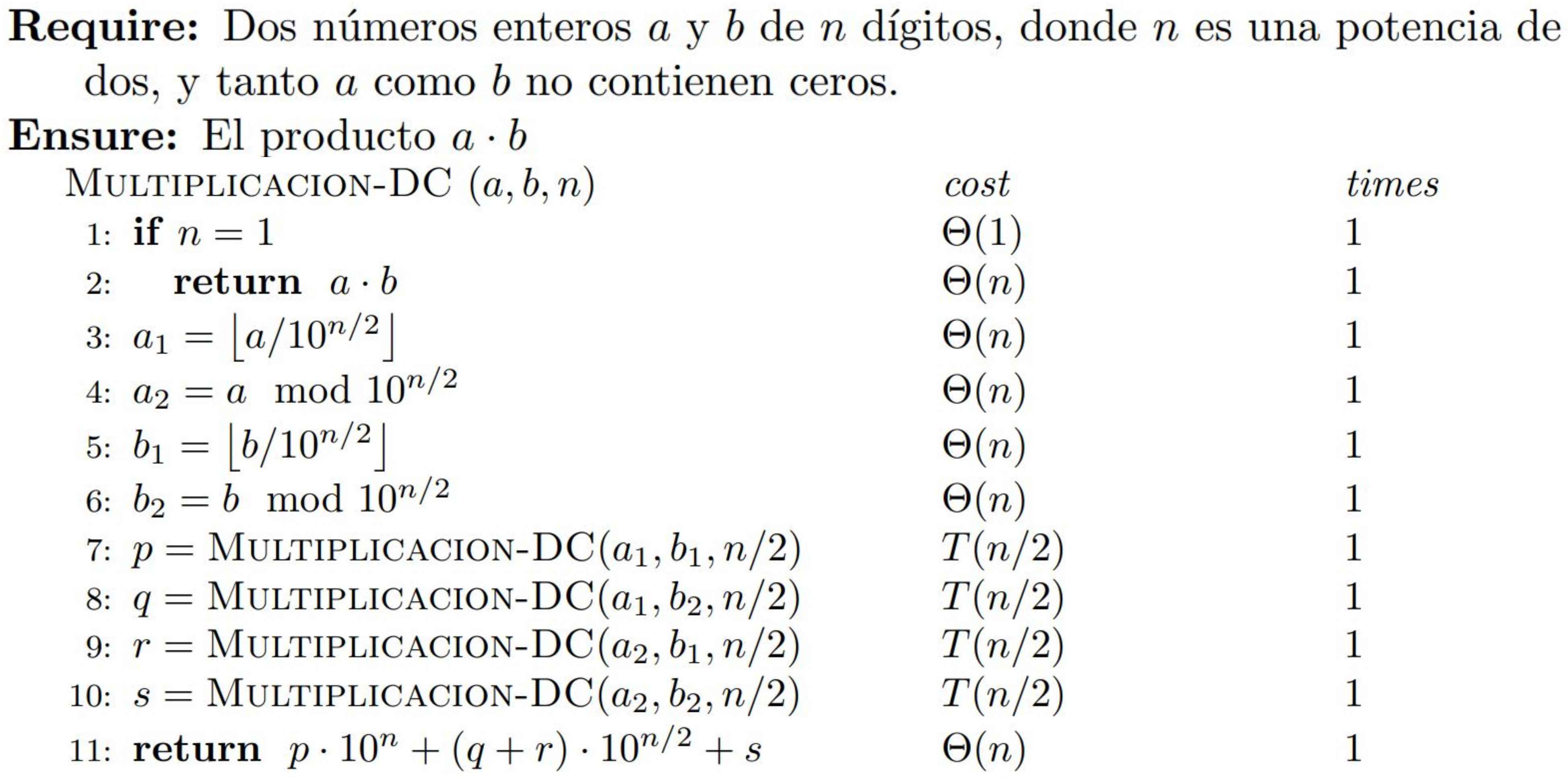
Require: Dos números enteros representados por $a[1..n], b[1..n]$
Ensure: El producto $a \cdot b$

Simple Approach

```
MULTIPLICACION-BASICA($a, b, n$)
1: total $= 0$
2: for $j = 1$ to $n$
3:     sum $= 0$
4:     for $i = 1$ to $n$
5:         sum $=$ sum $\cdot 10 + b[j] \cdot a[i]$
6:     total $=$ total $\cdot 10 +$ sum
7: return total
```

cost	*times*
c_1	1
c_2	$n+1$
c_3	n
c_4	$(n+1) \cdot n$
c_5	$n \cdot n$
c_6	n
c_7	1

One D&C approach



Require: Dos números enteros a y b de n dígitos, donde n es una potencia de dos, y tanto a como b no contienen ceros.

Ensure: El producto $a \cdot b$

```
MULTIPLICACION-DC $(a, b, n)$                                  cost          times
 1: if $n = 1$                                                 $\Theta(1)$   1
 2:     return $a \cdot b$                                     $\Theta(n)$   1
 3: $a_1 = \lfloor a/10^{n/2} \rfloor$                         $\Theta(n)$   1
 4: $a_2 = a \mod 10^{n/2}$                                    $\Theta(n)$   1
 5: $b_1 = \lfloor b/10^{n/2} \rfloor$                         $\Theta(n)$   1
 6: $b_2 = b \mod 10^{n/2}$                                    $\Theta(n)$   1
 7: $p =$ MULTIPLICACION-DC$(a_1, b_1, n/2)$                   $T(n/2)$      1
 8: $q =$ MULTIPLICACION-DC$(a_1, b_2, n/2)$                   $T(n/2)$      1
 9: $r =$ MULTIPLICACION-DC$(a_2, b_1, n/2)$                   $T(n/2)$      1
10: $s =$ MULTIPLICACION-DC$(a_2, b_2, n/2)$                   $T(n/2)$      1
11: return $p \cdot 10^n + (q + r) \cdot 10^{n/2} + s$         $\Theta(n)$   1
```

Karatsuba Algorithm

```
KARATSUBA $(a, b)$                                                  cost           times
 1: if $n \leq 1$                                                   $\Theta(1)$    1
 2:     return $a \cdot b$                                          $\Theta(n)$    1
 3: $a_1 = \lfloor a/10^{n/2} \rfloor$                              $\Theta(n)$    1
 4: $a_2 = a \mod 10^{n/2}$                                         $\Theta(n)$    1
 5: $b_1 = \lfloor b/10^{n/2} \rfloor$                              $\Theta(n)$    1
 6: $b_2 = b \mod 10^{n/2}$                                         $\Theta(n)$    1
 7: $p = \text{KARATSUBA}(a_1, b_1)$                                $T(n/2)$       1
 8: $q = \text{KARATSUBA}(a_1 + a_2, b_1 + b_2)$                    $T(n/2)$       1
 9: $s = \text{KARATSUBA}(a_2, b_2)$                                $T(n/2)$       1
10: return $p \cdot 10^n + (q - p - s) \cdot 10^n + s$              $\Theta(n)$    1
```

3 Matrix multiplication

Simple Approach

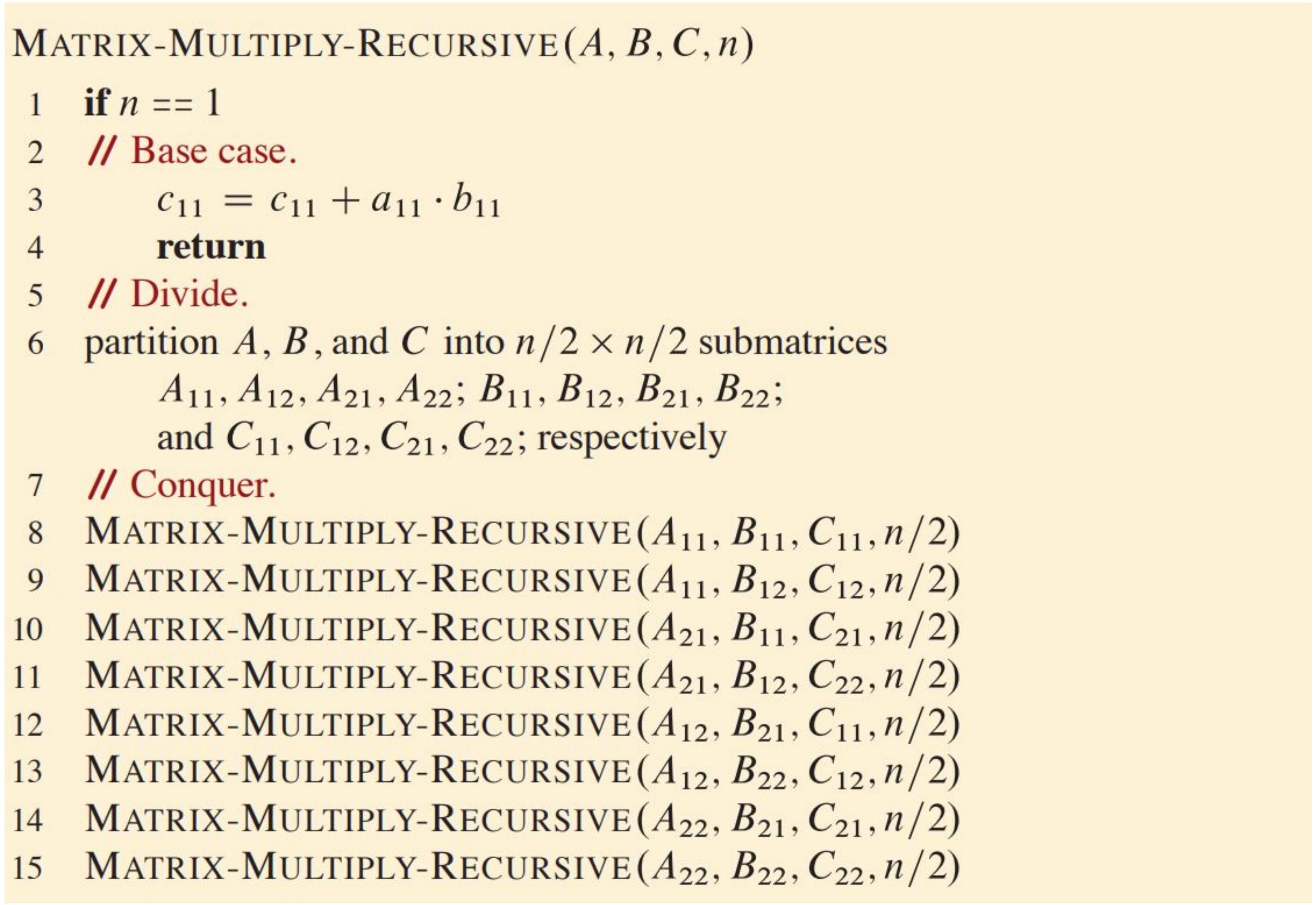
```
MATRIX-MULTIPLY(A, B, C, n)
for $i = 1$ to $n$                              // compute entries in each of $n$ rows
    for $j = 1$ to $n$                          // compute $n$ entries in row $i$
        for $k = 1$ to $n$
            $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$    // add in another term of equation (4.1)
```

D&C Approach

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}.$$

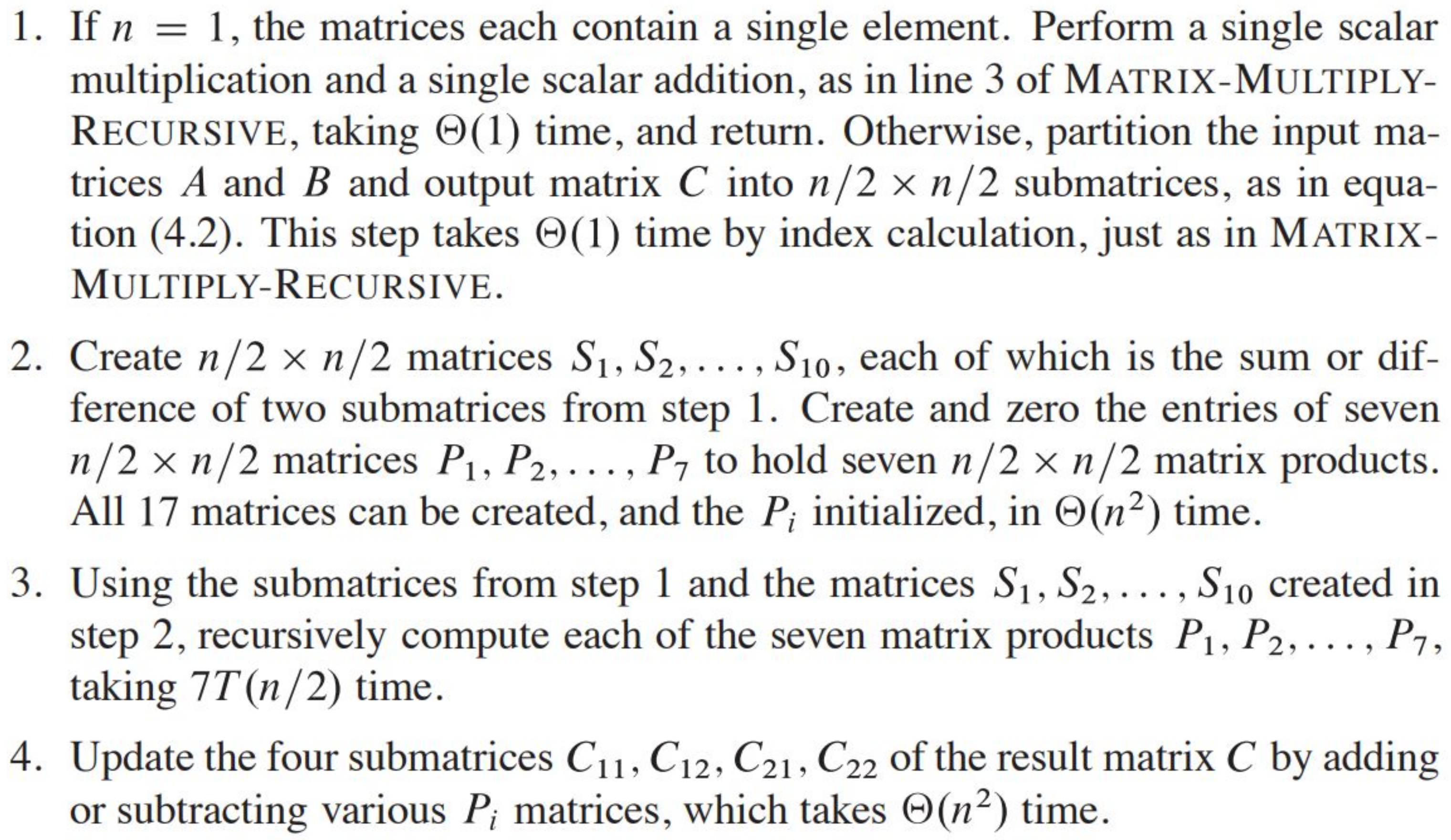
$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Simple D&C algorithm



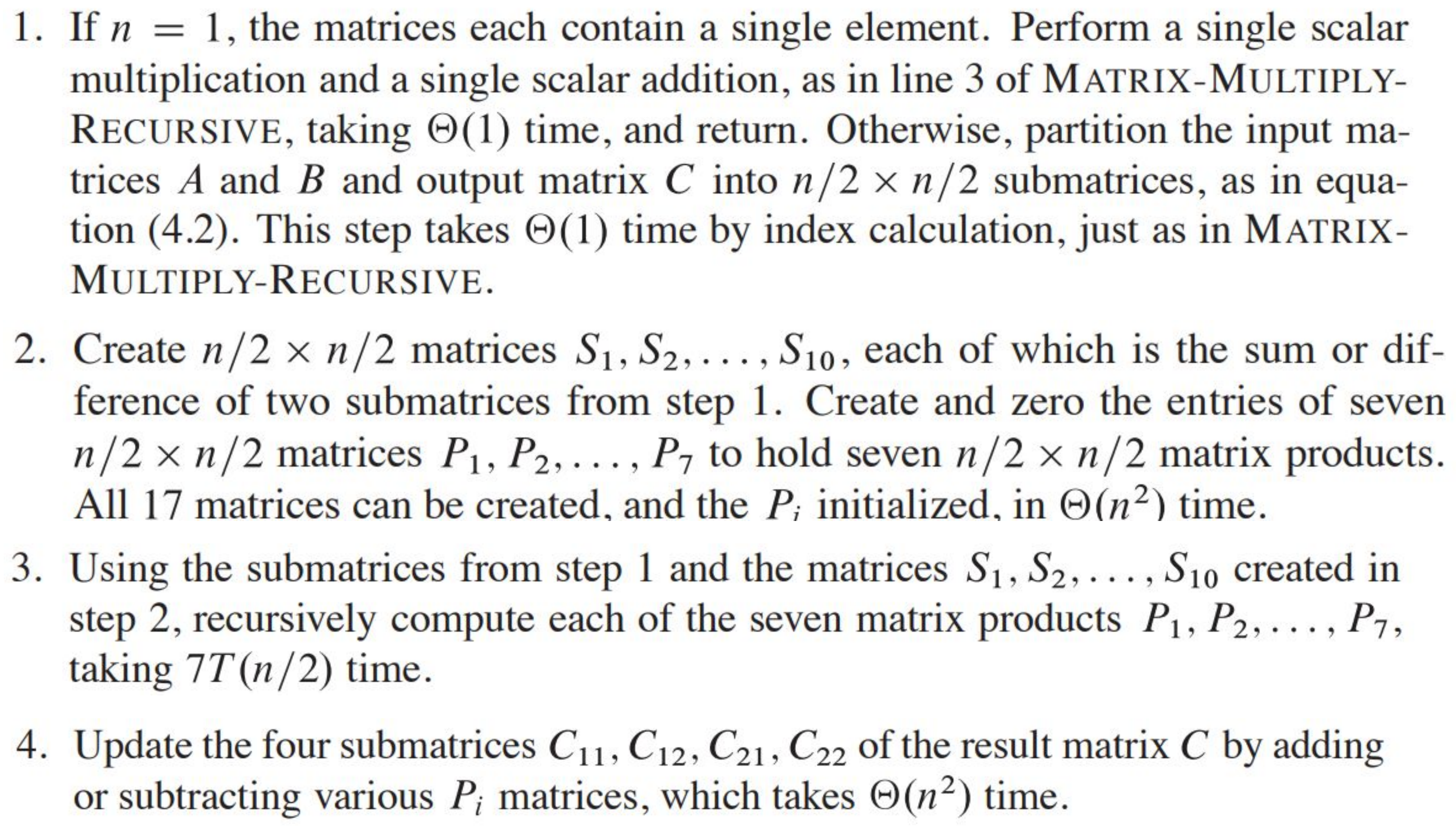
```
MATRIX-MULTIPLY-RECURSIVE(A, B, C, n)
if n == 1
// Base case.
    $c_{11} = c_{11} + a_{11} \cdot b_{11}$
    return
// Divide.
partition A, B, and C into $n/2 \times n/2$ submatrices
    $A_{11}, A_{12}, A_{21}, A_{22}$; $B_{11}, B_{12}, B_{21}, B_{22}$;
    and $C_{11}, C_{12}, C_{21}, C_{22}$; respectively
// Conquer.
MATRIX-MULTIPLY-RECURSIVE($A_{11}, B_{11}, C_{11}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{11}, B_{12}, C_{12}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{21}, B_{11}, C_{21}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{21}, B_{12}, C_{22}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{12}, B_{21}, C_{11}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{12}, B_{22}, C_{12}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{22}, B_{21}, C_{21}, n/2$)
MATRIX-MULTIPLY-RECURSIVE($A_{22}, B_{22}, C_{22}, n/2$)
```

Strassen’s algorithm description



1. If $n = 1$, the matrices each contain a single element. Perform a single scalar multiplication and a single scalar addition, as in line 3 of MATRIX-MULTIPLY-RECURSIVE, taking $\Theta(1)$ time, and return. Otherwise, partition the input matrices A and B and output matrix C into $n/2 \times n/2$ submatrices, as in equation (4.2). This step takes $\Theta(1)$ time by index calculation, just as in MATRIX-MULTIPLY-RECURSIVE.
2. Create $n/2 \times n/2$ matrices $S_1, S_2, \ldots, S_{10}$, each of which is the sum or difference of two submatrices from step 1. Create and zero the entries of seven $n/2 \times n/2$ matrices $P_1, P_2, \ldots, P_7$ to hold seven $n/2 \times n/2$ matrix products. All 17 matrices can be created, and the P_i initialized, in $\Theta(n^2)$ time.
3. Using the submatrices from step 1 and the matrices $S_1, S_2, \ldots, S_{10}$ created in step 2, recursively compute each of the seven matrix products $P_1, P_2, \ldots, P_7$, taking $7T(n/2)$ time.
4. Update the four submatrices $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C by adding or subtracting various P_i matrices, which takes $\Theta(n^2)$ time.

Strassen’s algorithm description



1. If $n = 1$, the matrices each contain a single element. Perform a single scalar multiplication and a single scalar addition, as in line 3 of MATRIX-MULTIPLY-RECURSIVE, taking $\Theta(1)$ time, and return. Otherwise, partition the input matrices A and B and output matrix C into $n/2 \times n/2$ submatrices, as in equation (4.2). This step takes $\Theta(1)$ time by index calculation, just as in MATRIX-MULTIPLY-RECURSIVE.

2. Create $n/2 \times n/2$ matrices $S_1, S_2, \ldots, S_{10}$, each of which is the sum or difference of two submatrices from step 1. Create and zero the entries of seven $n/2 \times n/2$ matrices $P_1, P_2, \ldots, P_7$ to hold seven $n/2 \times n/2$ matrix products. All 17 matrices can be created, and the P_i initialized, in $\Theta(n^2)$ time.

3. Using the submatrices from step 1 and the matrices $S_1, S_2, \ldots, S_{10}$ created in step 2, recursively compute each of the seven matrix products $P_1, P_2, \ldots, P_7$, taking $7T(n/2)$ time.

4. Update the four submatrices $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C by adding or subtracting various P_i matrices, which takes $\Theta(n^2)$ time.

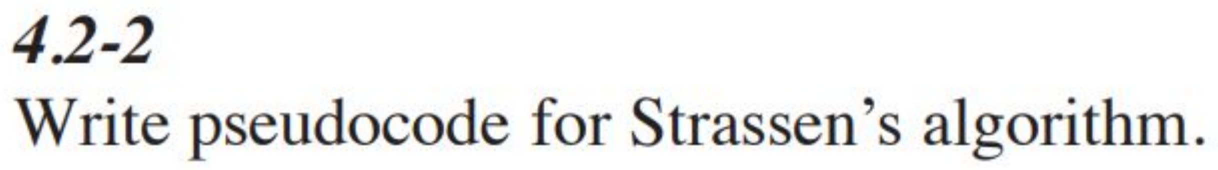
Step 2

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} \,, \\
S_2 &= A_{11} + A_{12} \,, \\
S_3 &= A_{21} + A_{22} \,, \\
S_4 &= B_{21} - B_{11} \,, \\
S_5 &= A_{11} + A_{22} \,, \\
S_6 &= B_{11} + B_{22} \,, \\
S_7 &= A_{12} - A_{22} \,, \\
S_8 &= B_{21} + B_{22} \,, \\
S_9 &= A_{11} - A_{21} \,, \\
S_{10} &= B_{11} + B_{12} \,.
\end{aligned}
$$

Step 2

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 & (&= A_{11} \cdot B_{12} - A_{11} \cdot B_{22}) \,, \\
P_2 &= S_2 \cdot B_{22} & (&= A_{11} \cdot B_{22} + A_{12} \cdot B_{22}) \,, \\
P_3 &= S_3 \cdot B_{11} & (&= A_{21} \cdot B_{11} + A_{22} \cdot B_{11}) \,, \\
P_4 &= A_{22} \cdot S_4 & (&= A_{22} \cdot B_{21} - A_{22} \cdot B_{11}) \,, \\
P_5 &= S_5 \cdot S_6 & (&= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}) \,, \\
P_6 &= S_7 \cdot S_8 & (&= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}) \,, \\
P_7 &= S_9 \cdot S_{10} & (&= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}) \,.
\end{aligned}
$$

Pseudocode



4.2-2

Write pseudocode for Strassen's algorithm.

4 Counting Inversions

Problem

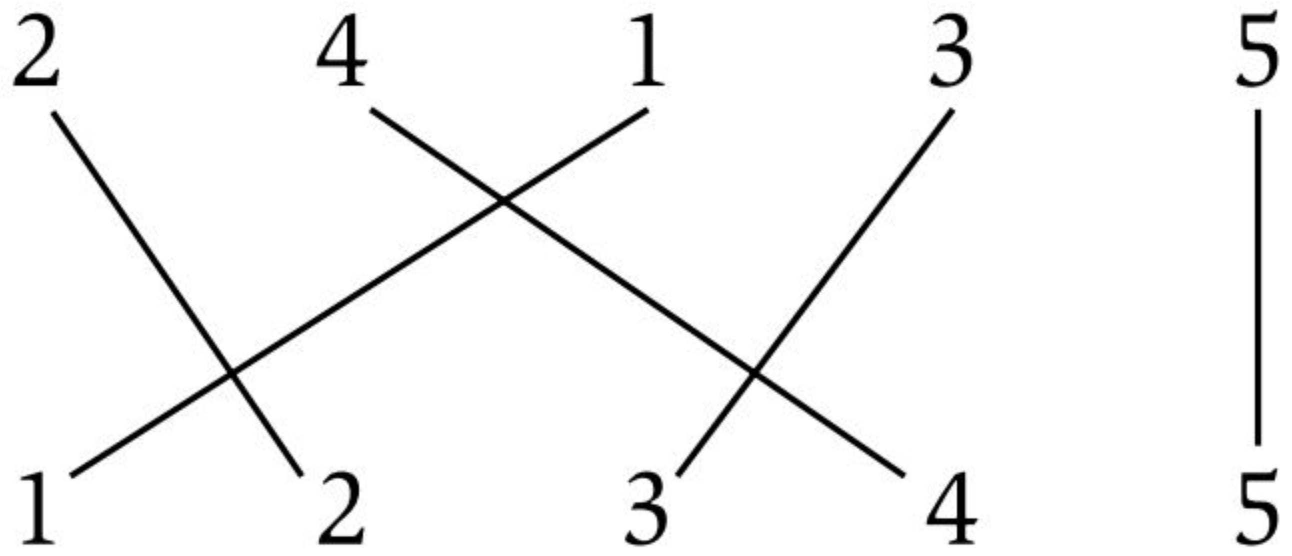


Figure 1: Kleinberg - Tardos, Algorithm Design

Simple Approach

```
INVERSIONES-INGENUO($A, n$)                     cost     times
1: total $= 0$                                  $c_1$    $1$
2: for $i = 1$ to $n - 1$                       $c_2$    $n$
3:     for $j = i + 1$ to $n$                   $c_3$    $\sum_{i=1}^{n-1} n - i + 1$
4:         if $A[i] > A[j]$                     $c_4$    $\sum_{i=1}^{n-1} n - i$
5:             total $=$ total $+ 1$            $c_5$    $\sum_{i=1}^{n-1} n - i$
6: return total                                 $c_6$    $1$
```

D&C approach

Input: An array of distinct integers $A[p..r]$
Output: The number of inversions in A.

```
INVERSIONS-DC($A, p, r$)                                   cost                 times
1: if ($p == r$)                                           $c_1$                1
2:     return 0                                            $c_2$                0
3: $q = \lfloor \frac{r-p+1}{2} \rfloor$                   $c_3$                1
4: $total_1$ = INVERSIONS-DC($A, p, q$)                    $T(\lfloor n/2 \rfloor)$   1
5: $total_2$ = INVERSIONS-DC($A, q+1, r$)                  $T(\lceil n/2 \rceil)$     1
6: $total_3$ = CENTRAL-INVERSIONS($A, p, q, r$)            $kn$                 1
7: return $total_1 + total_2 + total_3$                    $c_5$                1
```

Note that $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + kn$. Then, by the Master Theorem, $T(n) = \Theta(n \lg n)$.

Simple Approach

```
CENTRAL-INVERSIONS($A, p, q, r$)
 1: $n_1 = q - p + 1$
 2: $n_2 = r - q$
 3: Let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
 4: for $i = 1$ to $n_1$
 5:     $L[i] = A[p + i - 1]$
 6: for $j = 1$ to $n_2$
 7:     $R[j] = A[q + j]$
 8: $L[n_1 + 1] = \infty$
 9: $L[n_2 + 1] = \infty$
10: $i = 1$
11: $j = 1$
12: $total = 0$
13: for $k = p$ to $r$
14:     if $L[i] > R[j]$
15:         $A[k] = R[j]$
16:         $total = total + (n_1 + 1 - i)$
17:         $j = j + 1$
18:     else
19:         $A[k] = L[i]$
20:         $i = i + 1$
21: return total
```

Gracias

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA