*Universidad Politécnica de Madrid*
*Escuela Técnica Superior de Ingeniería de Sistemas Informáticos*

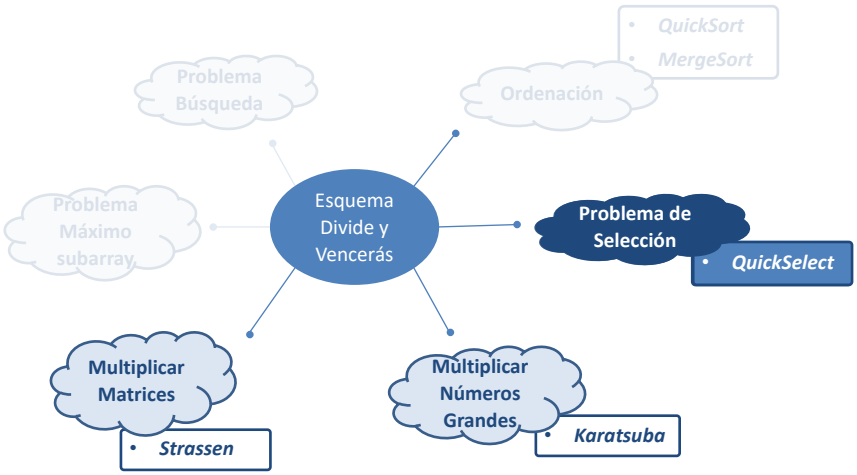
# Tema 7. Otros algoritmos basados en Divide y Vencerás

## Algorítmica y Complejidad

1

## Problema de Selección

### Algunos problemas planteados






2

## Problema de Selección

### Enunciado del Problema

Dado un *array*: un número *k*: **1**

| -2 | -21 | 3 | 4 | **12** | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

Devolver el *k*-ésimo número mayor del *array*

**12**



3

## Problema de Selección

### Enunciado del Problema

Dado un *array*: un número *k*: **3**

| -2 | -21 | 3 | 4 | 12 | **8** | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

Devolver el *k*-ésimo número mayor del *array*

**8**



4

## Problema de Selección

### Enunciado del Problema

Dado un *array*: un número *k*: **5**

| -2 | -21 | 3 | **4** | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

Devolver el *k*-ésimo número mayor del *array*

**4**



5

## Problema de Selección

**Actividad 7.1. Diseña e implementa un algoritmo que resuelva el problema de selección.**

Dado un *array*: un número *k*: **5**

| -2 | -21 | 3 | **4** | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

Devolver el *k*-ésimo número mayor del *array*

**4**



6

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

max: 10
j:2
j_max: 8
i:6
k: 5

| -2 | -21 | 3 | 4 | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max= Integer.MAX_VALUE;
   for (int j=0; j<k; j++) {
      int j_max = Integer.MIN_VALUE;
      for (int i=0; i<vector.length; i++)
             if (vector[i]< max && vector[i]> j_max)
                j_max = vector[i];
      max = j_max;
   }
   return max;
}
```

ERROR!!! No funciona si hay elementos repetidos

1. Problema de Selección

7

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:0

| -2 | -21 | 3 | 4 | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
             if (vector[i] > max) {
                j_max = i; max = vector[i];
             }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

1. Problema de Selección

8

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:0

| -2 | -21 | 3 | 4 | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
➡     for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Encuentra el elemento más grande del subvector

1. Problema de Selección

1 2 3

9

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:0 j_max:4

| -2 | -21 | 3 | 4 | 12 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
➡     int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Intercambiamos los elementos del vector

1. Problema de Selección

1 2 3

10

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:0 j_max:4

| 12 | -21 | 3 | 4 | -2 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

1. Problema de Selección

1 2 3

11

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:1

| 12 | -21 | 3 | 4 | -2 | 8 | 10 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
➡     for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Encuentra el elemento más grande del subvector

1. Problema de Selección

1 2 3

12

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

| | j:1 | | | | | j_max:6 | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | -21 | 3 | 4 | -2 | 8 | 10 | 5 | -6 | 1 |

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
➡     int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Intercambiamos los elementos del vector

1. Problema de Selección

1 2 3

13

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

| | j:1 | | | | | j_max:6 | | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 10 | 3 | 4 | -2 | 8 | -21 | 5 | -6 | 1 |

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

1. Problema de Selección

1 2 3

14

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:2

| 12 | 10 | 3 | 4 | -2 | 8 | -21 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
➡     for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Encuentra el elemento más grande del subvector

1. Problema de Selección

1 2 3

15

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:2 j_max:5

| 12 | 10 | 3 | 4 | -2 | 8 | -21 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
➡     int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Intercambiamos los elementos del vector

1. Problema de Selección

1 2 3

16

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:2 j_max:5

| 12 | 10 | 8 | 4 | -2 | 3 | -21 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```



17

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

j:3

| 12 | 10 | 8 | 4 | -2 | 3 | -21 | 5 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
➡     for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Encuentra el elemento más grande del subvector



18

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

| | | | j:3 | | | | j_max:7 | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 10 | 8 | 4 | -2 | 3 | -21 | 5 | -6 | 1 |

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
➡     int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Intercambiamos los elementos del vector

1. Problema de Selección

1 2 3

19

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5

| | | | j:3 | | | | j_max:7 | | |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 10 | 8 | 5 | -2 | 3 | -21 | 4 | -6 | 1 |

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

1. Problema de Selección

1 2 3

20

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.



k: 5

j:4

| 12 | 10 | 8 | 5 | -2 | 3 | -21 | 4 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
➡     for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Encuentra el elemento más grande del subvector

1. Problema de Selección

1 2 3

21

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.



k: 5

j:4 j_max:7

| 12 | 10 | 8 | 5 | -2 | 3 | -21 | 4 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
➡     int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Intercambiamos los elementos del vector

1. Problema de Selección

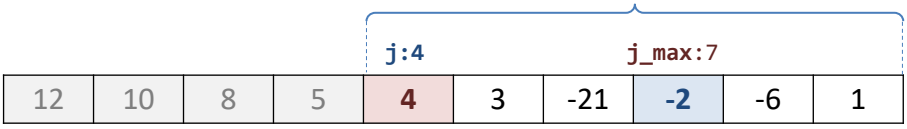
1 2 3

22

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5




```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
➡  for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

1. Problema de Selección

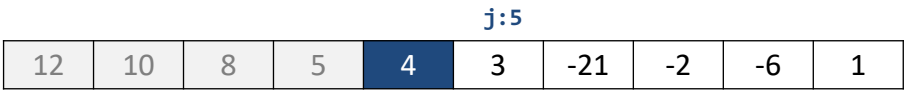
1 2 3

23

## Problema de Selección

**Actividad 7.1.** Diseña e implementa un algoritmo que resuelva el problema de selección.

k: 5




```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
➡  return max;
}
```

1. Problema de Selección

1 2 3

24

## Problema de Selección

**Actividad 7.1.** **Diseña e implementa un algoritmo que resuelva el problema de selección.**

k: 5

j:5

| 12 | 10 | 8 | 5 | 4 | 3 | -21 | -2 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Parecido a ordenación por selección.

Solo ordenamos los *k* primeros elementos del array

1. Problema de Selección

1 2 3

25

## Problema de Selección

### Algoritmo Simple

**Actividad 7.2.** **Calcula la complejidad del algoritmo.**

j:5

| 12 | 10 | 8 | 5 | 4 | 3 | -21 | -2 | -6 | 1 |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValor(int[] vector, int k){
   int max = Integer.MIN_VALUE;
   for (int j=0; j<k; j++) {
      max = vector[j];
      int j_max = j;
      for (int i=j; i<vector.length; i++)
            if (vector[i] > max) {
               j_max = i; max = vector[i];
            }
      int aux = vector[j]; vector[j] = vector[j_max]; vector[j_max]= aux;
   }
   return max;
}
```

Complejidad: $\Theta(N \cdot k)$

1. Problema de Selección

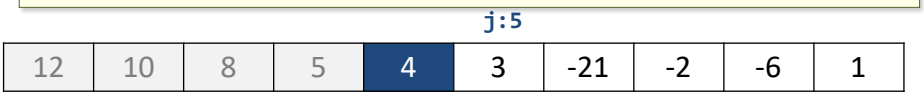
1 2 3

26

## Problema de Selección

### Algoritmo Simple

**Actividad 7.3.** **Diseña e implementa un algoritmo que tenga complejidad O(N·logN).**




```
int k_esimoValor (int[] vector, int k){
      ordenarHeapSortMayorMenor(vector);
      return vector[k-1];
}
```

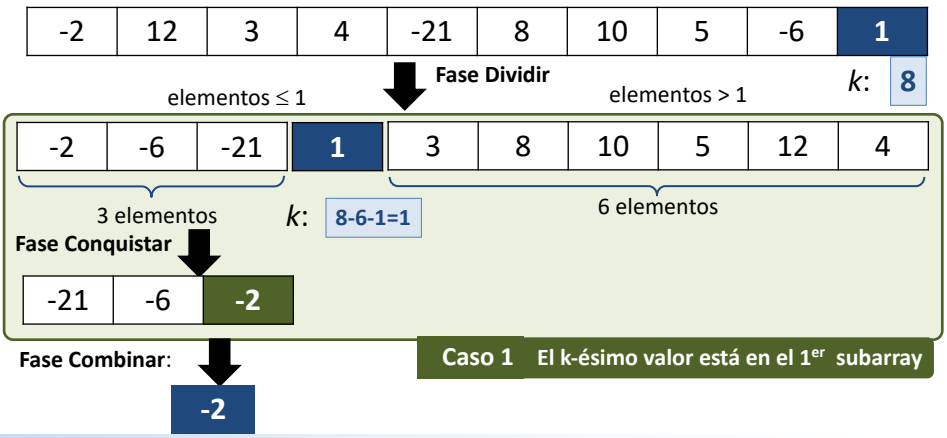
Complejidad: $\Theta(N \cdot \log N)$

1. Problema de Selección

27

## Problema de Selección

### QuickSelect: Algoritmo basado en Divide y Vencerás

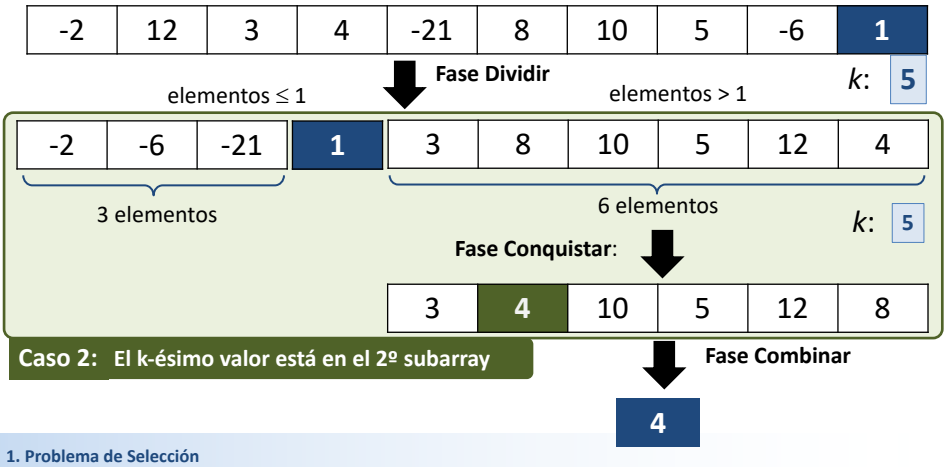



1. Problema de Selección

28

## Problema de Selección

### QuickSelect: Algoritmo basado en Divide y Vencerás




1. Problema de Selección

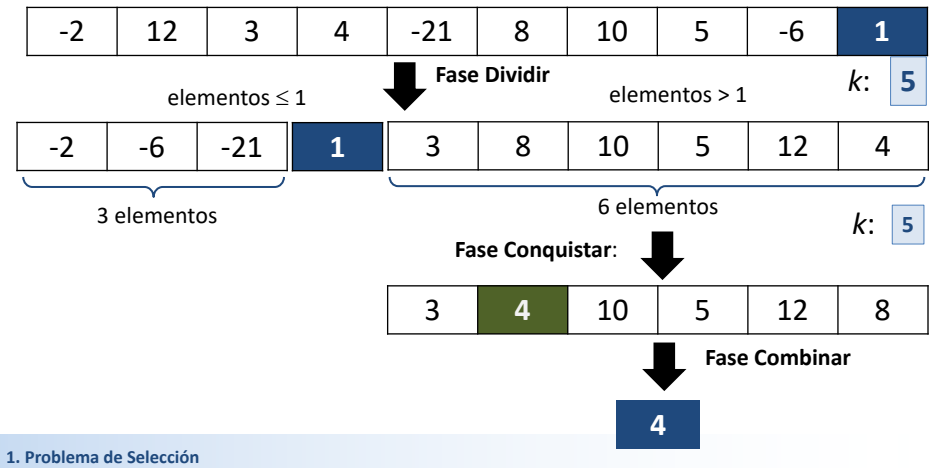
1 2 3

29

## Problema de Selección

### QuickSelect: Algoritmo basado en Divide y Vencerás

**Actividad 7.4.** Diseña e implementa un algoritmo basado en Divide y Vencerás




1. Problema de Selección

1 2 3

30

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación

| $x_0$ | ... | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | ... | $x_{50}$ |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

Utilizado en QuickSort

➡ Complejidad: $\Theta(N)$

1. Problema de Selección

1 — 2 — 3



## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación

i0: 7 iN: 12

| $x_0$ | ... | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | ... | $x_{50}$ |
|---|---|---|---|---|---|---|---|---|---|

```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

1. Problema de Selección

1 — 2 — 3

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación


```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

Caso Base

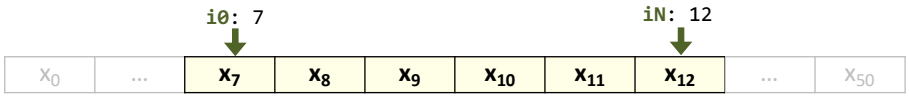
1. Problema de Selección

1 2 3

33

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación




```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

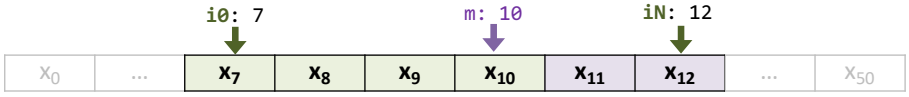
1. Problema de Selección

1 2 3

34

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación




```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

Dividir

**Utilizado en QuickSort**

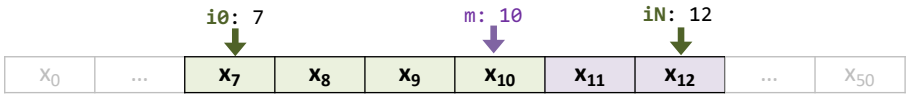
1. Problema de Selección

1 2 3

35

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación




```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

Conquistar

1. Problema de Selección

1 2 3

36

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación

**Actividad 7.5.** Calcula la complejidad en el mejor caso.

```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
int k_esimoValor(int[] vector, int k){
      return k_esimoValorAux(vector,0, vector.length-1, k);
}
```

**Complejidad: $\Theta(N)$**

1. Problema de Selección

1 2 3

37

## Problema de Selección

### Algoritmo basado en Divide y Vencerás: Implementación

**Actividad 7.5.** Calcula la complejidad en el peor caso.

```
int k_esimoValorAux (int[] vector, int i0, int iN, int k){
      if (i0 == iN)
          return vector[i0];
      else {
         int m = ordenarPorPivote(vector, i0, iN);
         if (k<=iN-m)
             return k_esimoValorAux(vector, m+1, iN, k);
         else if (k==iN-m+1)
             return vector[m];
         else
             return k_esimoValorAux(vector, i0, m-1, k-(iN-m+1));
      }
}
```

m=i0

Ecuación de Recurrencia

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ T(N-1) + \Theta(N) & N > 1 \end{cases}$$

**Complejidad: $\Theta(N^2)$**

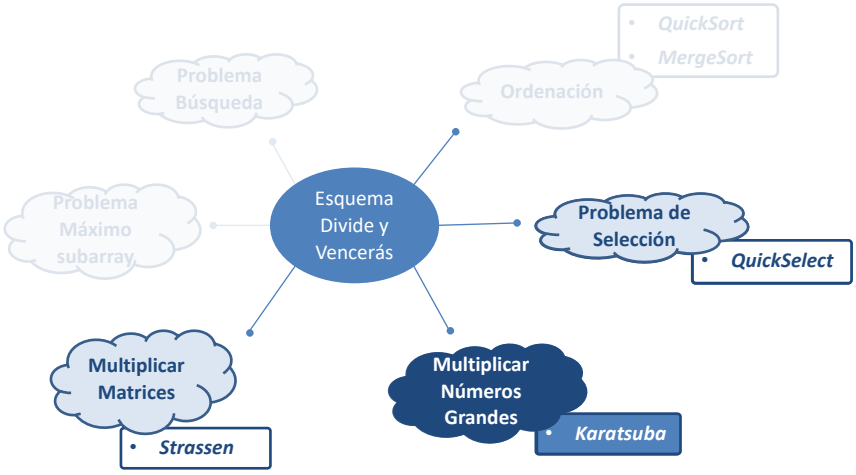
1. Problema de Selección

1 2 3

38

## Problema de Selección

### Algunos problemas planteados




1. Problema de Selección

1 2 3

39

## Multiplicar números grandes

### Enunciado del Problema

Dados dos números:

| a = | 1 | 2 | 3 | 4 |
|---|---|---|---|---|

| b= | 4 | 1 | 5 | 2 |
|---|---|---|---|---|

Multiplicarlos:

| c= a·b = | 5 | 1 | 2 | 3 | 5 | 6 | 8 |
|---|---|---|---|---|---|---|---|

2. Multiplicar Números Grandes

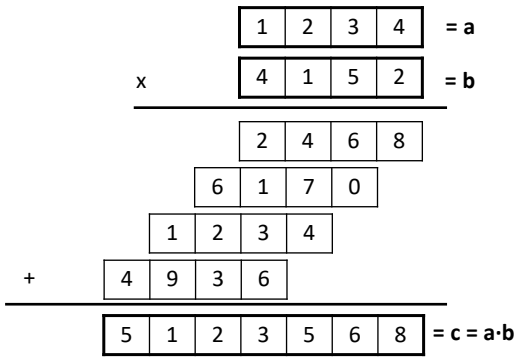
1 2 3

40 de 98

40

## Multiplicar números grandes

### Algoritmo Tradicional



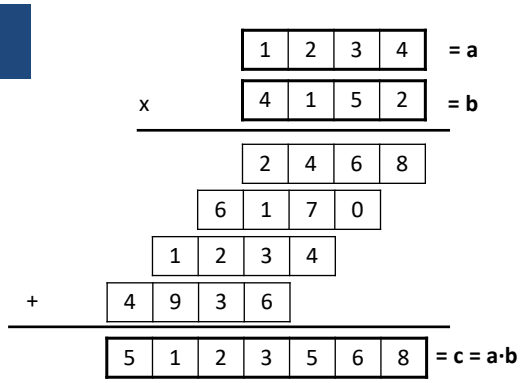



41

## Multiplicar números grandes

**Actividad 7.6.** Calcula la complejidad del Algoritmo de acuerdo con el número de dígitos, *N*, de los multiplicandos.

**Complejidad: $\Theta(N^2)$**

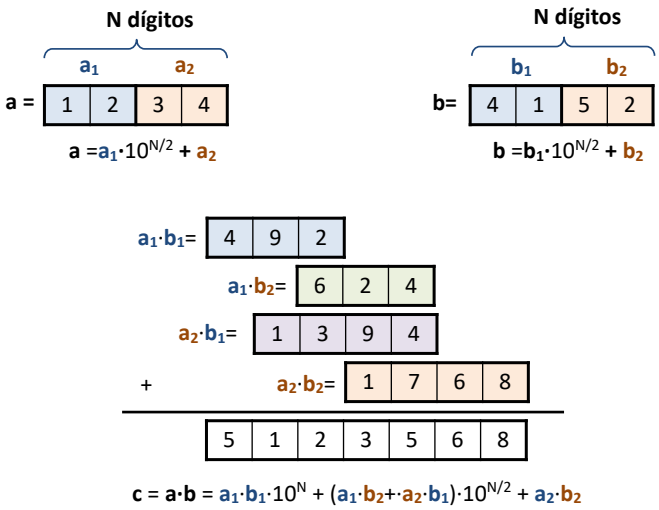





42

## Multiplicar números grandes

### Algoritmo inmediato basado en Divide y Vencerás

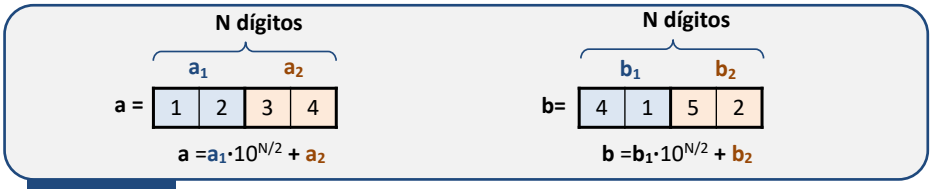





43

## Multiplicar números grandes

### Enunciado del Problema




**Fase Dividir**






44

## Multiplicar números grandes

### Enunciado del Problema

N dígitos

$a_1$ $a_2$

a = 1 2 3 4

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= 4 1 5 2

$b = b_1 \cdot 10^{N/2} + b_2$

$a_1 \cdot b_1$= 4 9 2

$a_1 \cdot b_2$= 6 2 4

$a_2 \cdot b_1$= 1 3 9 4

$a_2 \cdot b_2$= 1 7 6 8

Fase Conquistar

5 1 2 3 5 6 8

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$

2. Multiplicar Números Grandes

1 2 3

45 de 98

45

## Multiplicar números grandes

### Enunciado del Problema

N dígitos

$a_1$ $a_2$

a = 1 2 3 4

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= 4 1 5 2

$b = b_1 \cdot 10^{N/2} + b_2$

4 9 2

6 2 4

1 3 9 4

\+ 1 7 6 8

5 1 2 3 5 6 8

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$

Fase Combinar

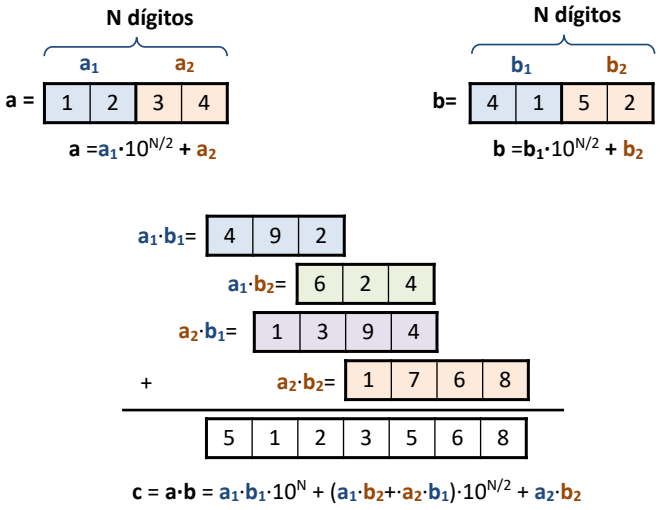
2. Multiplicar Números Grandes

1 2 3

46 de 98

46

**Actividad 7.7.** Calcula la complejidad de este algoritmo basado en divide y vencerás para multiplicar números




2. Multiplicar Números Grandes

1 2 3

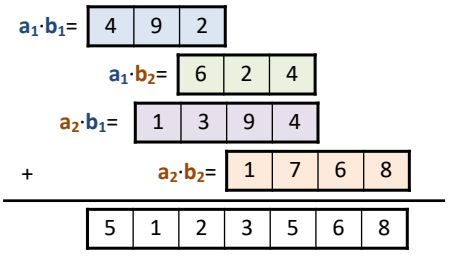
47 de 98

47

**Actividad 7.7.** Calcula la complejidad de este algoritmo basado en divide y vencerás para multiplicar números

**Fase Dividir**


**Fase Conquistar**




**Fase Combinar**

c = a·b = $a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 4 \cdot T\left(\frac{N}{2}\right) + \Theta(N) & N > 1 \end{cases}$$

48 de 98

48

**Actividad 7.7.** Calcula la complejidad de este algoritmo basado en divide y vencerás para multiplicar números

**Fase Dividir**

N dígitos

$a_1$ $a_2$

a = | 1 | 2 | 3 | 4 |

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= | 4 | 1 | 5 | 2 |

$b = b_1 \cdot 10^{N/2} + b_2$

**Fase Conquistar**

$a_1 \cdot b_1$ = | 4 | 9 | 2 |

$a_1 \cdot b_2$ = | 6 | 2 | 4 |

$a_2 \cdot b_1$ = | 1 | 3 | 9 | 4 |

\+ $a_2 \cdot b_2$ = | 1 | 7 | 6 | 8 |

**Fase Combinar**

| 5 | 1 | 2 | 3 | 5 | 6 | 8 |

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 4 \cdot T\left(\frac{N}{2}\right) + \Theta(N^1) & N > 1 \end{cases}$$

**Teorema Maestro**

(1er Caso)

$\text{Log}_2(4) = 2 > 1$

49 de 98

49

**Actividad 7.7.** Calcula la complejidad de este algoritmo basado en divide y vencerás para multiplicar números

**Fase Dividir**

N dígitos

$a_1$ $a_2$

a = | 1 | 2 | 3 | 4 |

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= | 4 | 1 | 5 | 2 |

$b = b_1 \cdot 10^{N/2} + b_2$

**Fase Conquistar**

$a_1 \cdot b_1$ = | 4 | 9 | 2 |

$a_1 \cdot b_2$ = | 6 | 2 | 4 |

$a_2 \cdot b_1$ = | 1 | 3 | 9 | 4 |

\+ $a_2 \cdot b_2$ = | 1 | 7 | 6 | 8 |

**Fase Combinar**

| 5 | 1 | 2 | 3 | 5 | 6 | 8 |

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 4 \cdot T\left(\frac{N}{2}\right) + \Theta(N^1) & N > 1 \end{cases}$$

**Teorema Maestro**

(1er Caso)

$\text{Log}_2(4) = 2 > 1$

NO MEJORA EL ALGORTIMO NATURAL

**Complejidad: $\Theta(N^2)$**

50

# Multiplicar números grandes

## Algoritmo de Karatsuba

N dígitos

$a_1$ $a_2$

a = | 1 | 2 | 3 | 4 |

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= | 4 | 1 | 5 | 2 |

$b = b_1 \cdot 10^{N/2} + b_2$

**4 multiplicaciones**

$a_1 \cdot b_1$= | 4 | 9 | 2 |

$a_1 \cdot b_2$= | 6 | 2 | 4 |

$a_2 \cdot b_1$= | 1 | 3 | 9 | 4 |

$a_2 \cdot b_2$= | 1 | 7 | 6 | 8 |

$$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$$



51

# Multiplicar números grandes

## Algoritmo de Karatsuba

N dígitos

$a_1$ $a_2$

a = | 1 | 2 | 3 | 4 |

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= | 4 | 1 | 5 | 2 |

$b = b_1 \cdot 10^{N/2} + b_2$

**3 multiplicaciones**

$a_1 \cdot b_1$= | 4 | 9 | 2 |

$p_3 = (a_1 + a_2) \cdot (b_1 + b_2)$= | 4 | 2 | 7 | 8 |

$a_2 \cdot b_2$= | 1 | 7 | 6 | 8 |

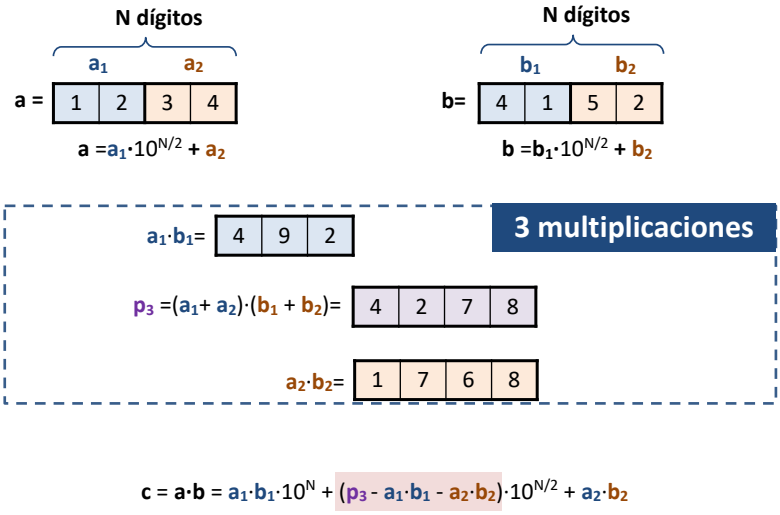
$$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot 10^{N/2} + a_2 \cdot b_2$$



52

## Multiplicar números grandes

### Algoritmo de Karatsuba



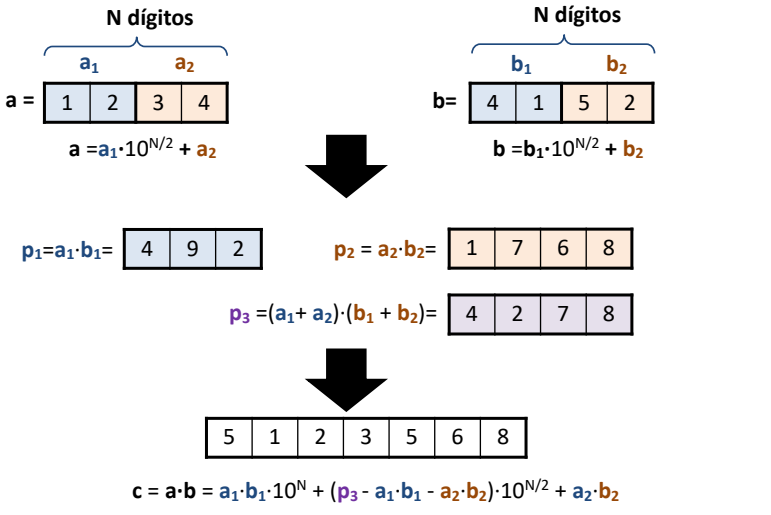

$$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (p_3 - a_1 \cdot b_1 - a_2 \cdot b_2) \cdot 10^{N/2} + a_2 \cdot b_2$$



53

## Multiplicar números grandes

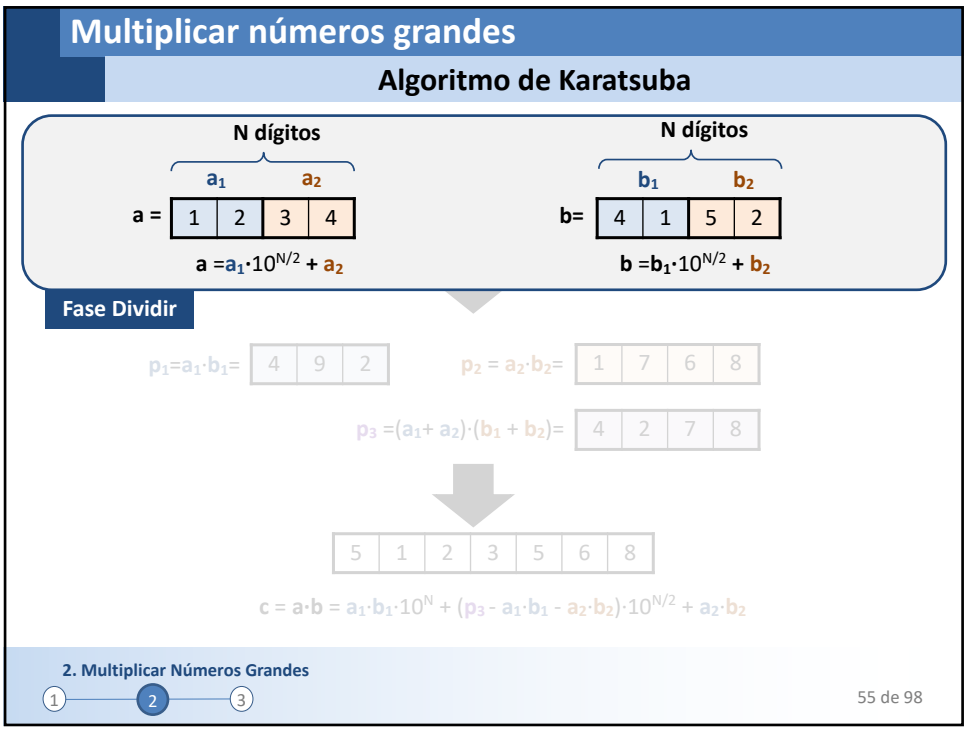
### Algoritmo de Karatsuba



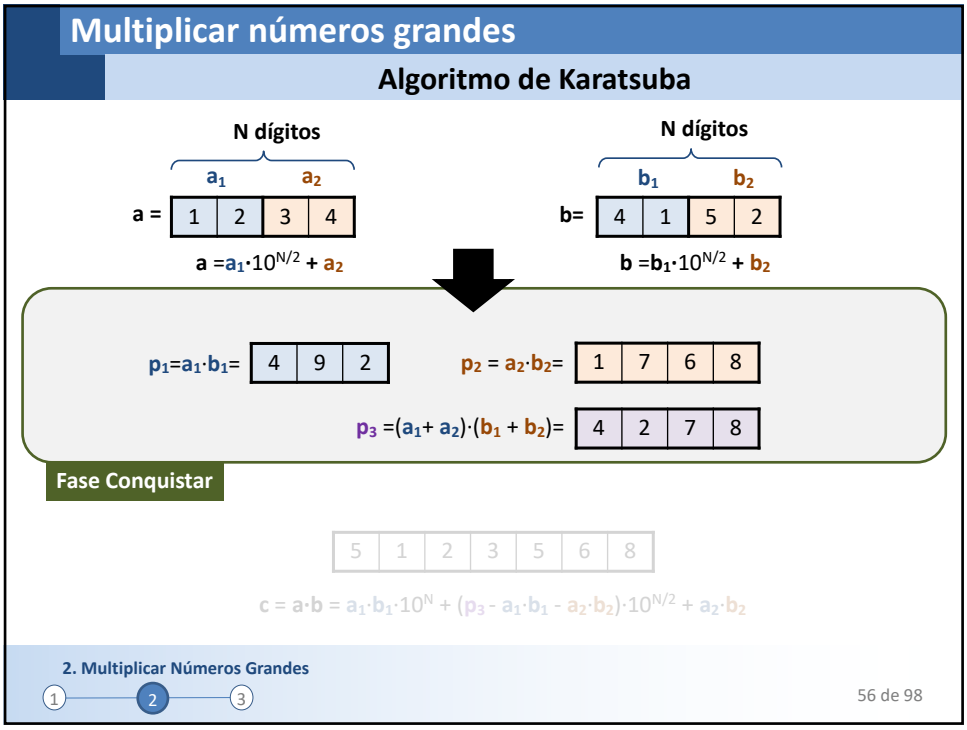

$$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (p_3 - a_1 \cdot b_1 - a_2 \cdot b_2) \cdot 10^{N/2} + a_2 \cdot b_2$$



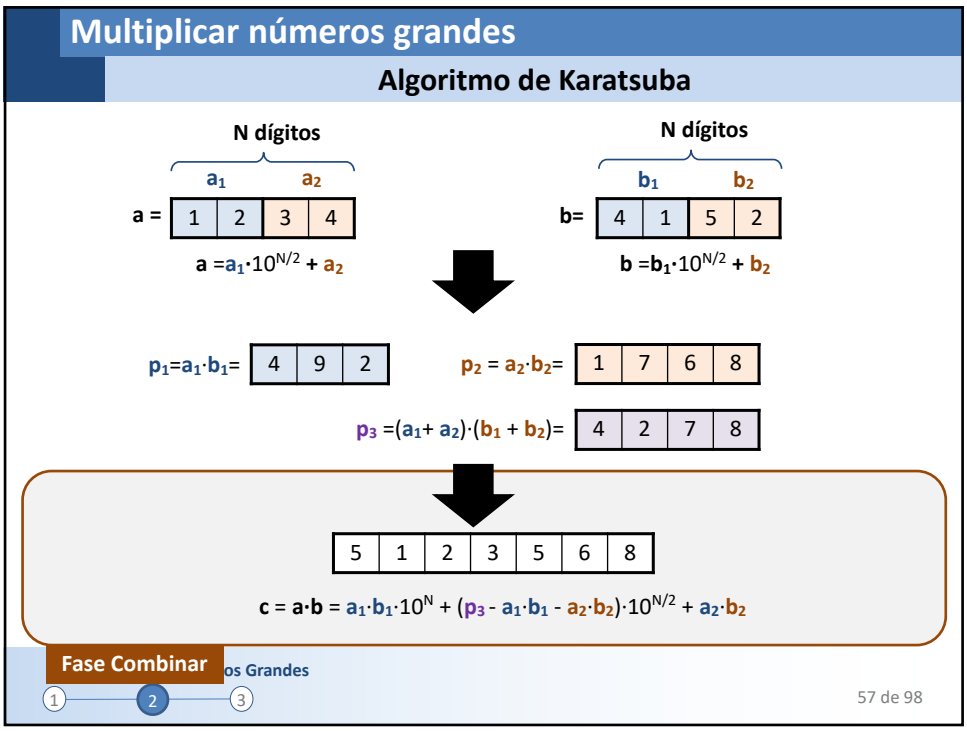
54


Multiplicar números grandes
Algoritmo de Karatsuba
N dígitos
N dígitos
a1 a2
b1 b2
a = 1 2 3 4
b= 4 1 5 2
a =a1·10N/2 + a2
b =b1·10N/2 + b2
Fase Dividir
p1=a1·b1= 4 9 2
p2 = a2·b2= 1 7 6 8
p3 =(a1+ a2)·(b1 + b2)= 4 2 7 8
5 1 2 3 5 6 8
c = a·b = a1·b1·10N + (p3 - a1·b1 - a2·b2)·10N/2 + a2·b2
2. Multiplicar Números Grandes
1 2 3
55 de 98

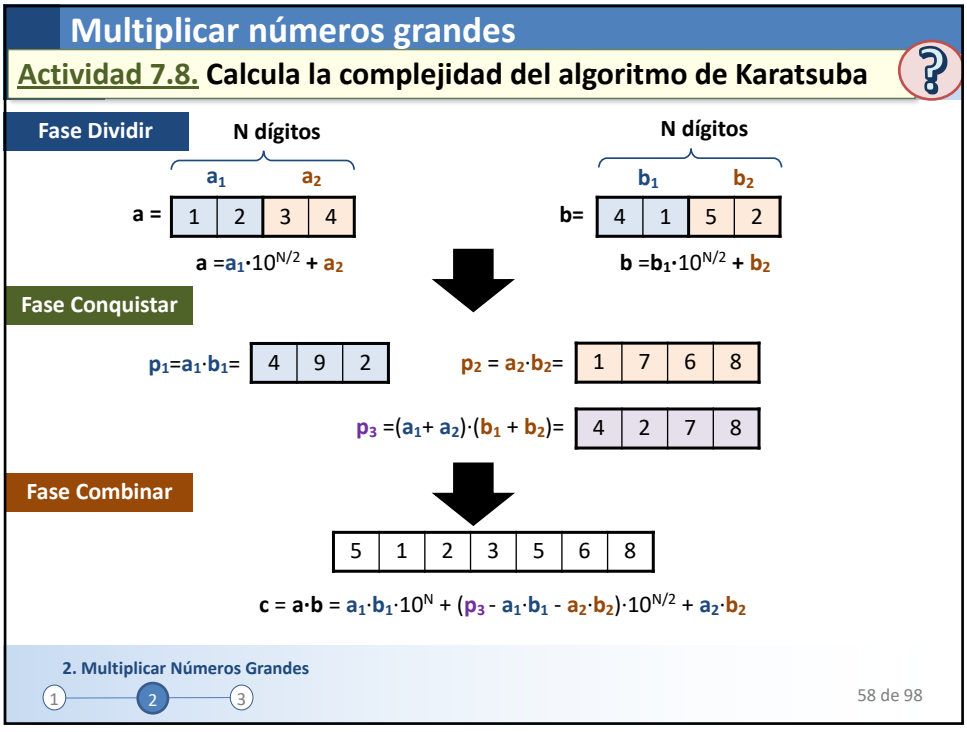
55


Multiplicar números grandes
Algoritmo de Karatsuba
N dígitos
N dígitos
a1 a2
b1 b2
a = 1 2 3 4
b= 4 1 5 2
a =a1·10N/2 + a2
b =b1·10N/2 + b2
p1=a1·b1= 4 9 2
p2 = a2·b2= 1 7 6 8
p3 =(a1+ a2)·(b1 + b2)= 4 2 7 8
Fase Conquistar
5 1 2 3 5 6 8
c = a·b = a1·b1·10N + (p3 - a1·b1 - a2·b2)·10N/2 + a2·b2
2. Multiplicar Números Grandes
1 2 3
56 de 98

56


Multiplicar números grandes
Algoritmo de Karatsuba
N dígitos
N dígitos
a = 1 2 3 4
b= 4 1 5 2
a =a1·10N/2 + a2
b =b1·10N/2 + b2
p1=a1·b1= 4 9 2
p2 = a2·b2= 1 7 6 8
p3 =(a1+ a2)·(b1 + b2)= 4 2 7 8
5 1 2 3 5 6 8
c = a·b = a1·b1·10N + (p3 - a1·b1 - a2·b2)·10N/2 + a2·b2
Fase Combinar
os Grandes
57 de 98

57


Multiplicar números grandes
Actividad 7.8. Calcula la complejidad del algoritmo de Karatsuba
Fase Dividir
N dígitos
N dígitos
a = 1 2 3 4
b= 4 1 5 2
a =a1·10N/2 + a2
b =b1·10N/2 + b2
Fase Conquistar
p1=a1·b1= 4 9 2
p2 = a2·b2= 1 7 6 8
p3 =(a1+ a2)·(b1 + b2)= 4 2 7 8
Fase Combinar
5 1 2 3 5 6 8
c = a·b = a1·b1·10N + (p3 - a1·b1 - a2·b2)·10N/2 + a2·b2
2. Multiplicar Números Grandes
58 de 98

58

## Multiplicar números grandes

**Actividad 7.8.** Calcula la complejidad del algoritmo de Karatsuba

**Fase Dividir**

N dígitos

$a_1$ $a_2$

a = 1 2 3 4

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= 4 1 5 2

$b = b_1 \cdot 10^{N/2} + b_2$

**Fase Conquistar**

$p_1 = a_1 \cdot b_1$ = 4 9 2

$p_2 = a_2 \cdot b_2$ = 1 7 6 8

$p_3 = (a_1 + a_2) \cdot (b_1 + b_2)$ = 4 2 7 8

**Fase Combinar**

5 1 2 3 5 6 8

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (p_3 - a_1 \cdot b_1 - a_2 \cdot b_2) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 3 \cdot T\left(\frac{N}{2}\right) + \Theta(N) & N > 1 \end{cases}$$

59 de 98

59

## Multiplicar números grandes

**Actividad 7.8.** Calcula la complejidad del algoritmo de Karatsuba

**Fase Dividir**

N dígitos

$a_1$ $a_2$

a = 1 2 3 4

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= 4 1 5 2

$b = b_1 \cdot 10^{N/2} + b_2$

**Fase Conquistar**

$p_1 = a_1 \cdot b_1$ = 4 9 2

$p_2 = a_2 \cdot b_2$ = 1 7 6 8

$p_3 = (a_1 + a_2) \cdot (b_1 + b_2)$ = 4 2 7 8

**Fase Combinar**

5 1 2 3 5 6 8

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (p_3 - a_1 \cdot b_1 - a_2 \cdot b_2) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 3 \cdot T\left(\frac{N}{2}\right) + \Theta(N^1) & N > 1 \end{cases}$$

**Teorema Maestro**

(1er Caso)

$\text{Log}_2(3) = 1.58 > 1$

60 de 98

60

## Multiplicar números grandes

**Actividad 7.8.** Calcula la complejidad del algoritmo de Karatsuba

**Fase Dividir**

N dígitos

$a_1$ $a_2$

a = | 1 | 2 | 3 | 4 |

$a = a_1 \cdot 10^{N/2} + a_2$

N dígitos

$b_1$ $b_2$

b= | 4 | 1 | 5 | 2 |

$b = b_1 \cdot 10^{N/2} + b_2$

**Fase Conquistar**

$p_1 = a_1 \cdot b_1$ = | 4 | 9 | 2 |

$p_2 = a_2 \cdot b_2$ = | 1 | 7 | 6 | 8 |

$p_3 = (a_1 + a_2) \cdot (b_1 + b_2)$ = | 4 | 2 | 7 | 8 |

**Fase Combinar**

| 5 | 1 | 2 | 3 | 5 | 6 | 8 |

$c = a \cdot b = a_1 \cdot b_1 \cdot 10^N + (p_3 - a_1 \cdot b_1 - a_2 \cdot b_2) \cdot 10^{N/2} + a_2 \cdot b_2$

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 3 \cdot T\left(\frac{N}{2}\right) + \Theta(N^1) & N > 1 \end{cases}$$

**Teorema Maestro**

(1er Caso)

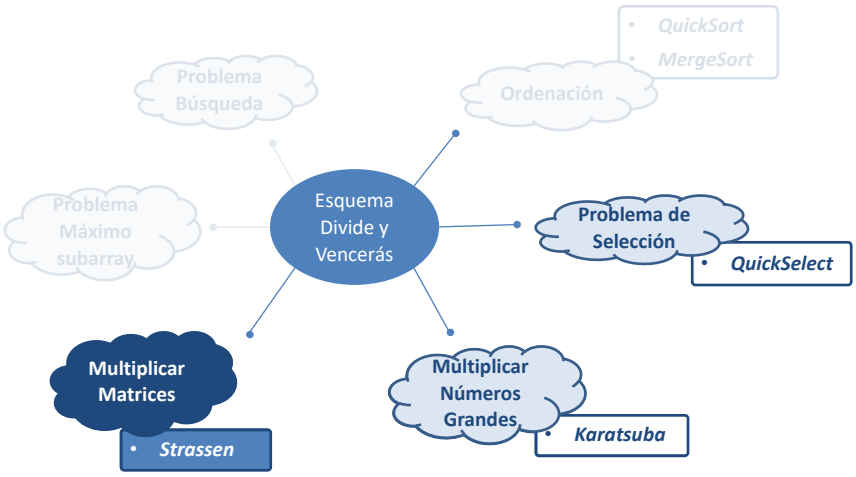
$\text{Log}_2(3) = 1.58 > 1$

**Complejidad:** $\Theta(N^{1.58})$

61

## Multiplicar números grandes

### Algunos problemas planteados




2. Multiplicar Números Grandes

1 2 3

62 de 98

62

## Multiplicar Matrices

### Enunciado del Problema

Dadas dos matrices cuadradas:

**A =**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 3 | 7 | 5 | 2 | 7 |
| 2 | 3 | 3 | 4 | 5 |
| 4 | 1 | 6 | 8 | 0 |
| 1 | 0 | 3 | 4 | 3 |

**B =**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 0 | 3 | 4 |
| 5 | 2 | 8 | 1 | 2 |
| 3 | 3 | 3 | 8 | 1 |
| 4 | 0 | 5 | 4 | 5 |
| 2 | 6 | 2 | 2 | 3 |

Multiplicarlas:

**A·B =**

| | | | | |
|---|---|---|---|---|
| 46 | 44 | 55 | 55 | 46 |
| 75 | 74 | 95 | 78 | 62 |
| 52 | 47 | 63 | 59 | 52 |
| 59 | 24 | 66 | 93 | 64 |
| 32 | 28 | 35 | 49 | 36 |



63

## Multiplicar Matrices

### Algoritmo Natural

**Actividad 7.9. Diseña e implementa un algoritmo que multiplique dos matrices**

```
int[][] multiplicar(int[][] A, int[][] B){
  int[][] resul= new int[A.length][A[0].length];
  for (int f=0; f<A.length; f++)
    for (int c=0; c<A[0].length; c++) {
      resul[f][c]=0;
      for (int i=0; i<A[0].length; i++)
          resul[f][c]+=A[f][i]*B[i][c];
    }
  return resul;
}
```



64

## Multiplicar Matrices

### Algoritmo Natural

**Actividad 7.10.** Calcula la complejidad del Algoritmo

```
int[][] multiplicar(int[][] A, int[][] B){
  int[][] resul= new int[A.length][A[0].length];
  for (int f=0; f<A.length; f++)
    for (int c=0; c<A[0].length; c++) {
      resul[f][c]=0;
      for (int i=0; i<A[0].length; i++)
          resul[f][c]+=A[f][i]*B[i][c];
    }
  return resul;
}
```

**Complejidad:** $\Theta(N^3)$

3. Multiplicar Matrices

1 2 3

65

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

**A**

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

**B**

**Vamos a considerar una matriz cuadrada de dimensión $2^k$**

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

**C = A·B**

3. Multiplicar Matrices

1 2 3

66

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

C = A·B



67

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

| | | | | |
|---|---|---|---|---|
| $A_{11}$ | 1 | 2 | 3 | 5 |
| | 3 | 7 | 5 | 7 |
| $A_{21}$ | 2 | 3 | 3 | 5 |
| | 1 | 0 | 3 | 3 |

$A_{12}$ $A_{22}$

A

| | | | | |
|---|---|---|---|---|
| $B_{11}$ | 1 | 1 | 0 | 4 |
| | 5 | 2 | 8 | 2 |
| $B_{21}$ | 3 | 3 | 3 | 1 |
| | 2 | 6 | 2 | 3 |

$B_{12}$ $B_{22}$

B

$C_{11}=A_{11}\cdot B_{11} + A_{12}\cdot B_{21}$ $C_{12} = A_{11}\cdot B_{12} + A_{12}\cdot B_{22}$

$C_{21}= A_{21}\cdot B_{11} + A_{22}\cdot B_{21}$ $C_{22} = A_{21}\cdot B_{12} + A_{22}\cdot B_{22}$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

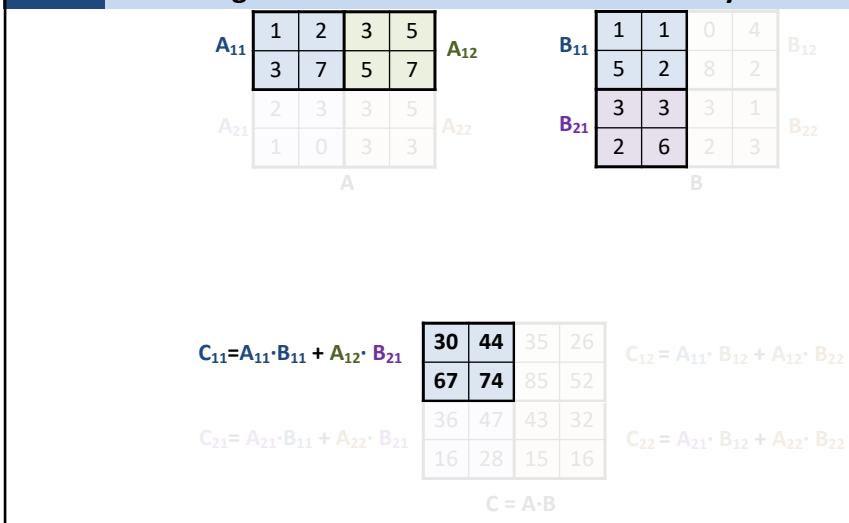
C = A·B



68

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

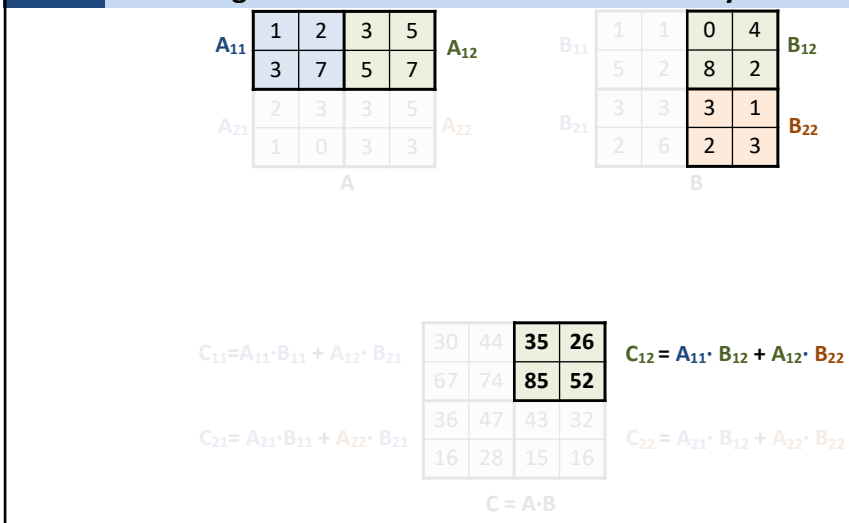





69

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

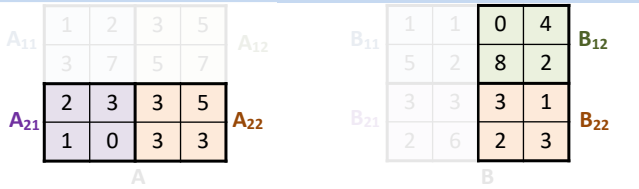





70

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás




$C_{11}=A_{11}\cdot B_{11} + A_{12}\cdot B_{21}$

$C_{12} = A_{11}\cdot B_{12} + A_{12}\cdot B_{22}$

$C_{21} = A_{21}\cdot B_{11} + A_{22}\cdot B_{21}$

$C_{22} = A_{21}\cdot B_{12} + A_{22}\cdot B_{22}$

| 30 | 44 | 35 | 26 |
|---|---|---|---|
| 67 | 74 | 85 | 52 |
| 36 | 47 | **43** | **32** |
| 16 | 28 | **15** | **16** |

C = A·B

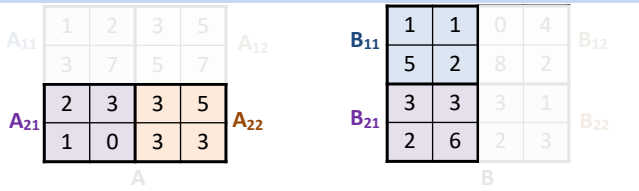
3. Multiplicar Matrices

1 — 2 — 3

71

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás




$C_{11}=A_{11}\cdot B_{11} + A_{12}\cdot B_{21}$

$C_{12} = A_{11}\cdot B_{12} + A_{12}\cdot B_{22}$

$C_{21} = A_{21}\cdot B_{11} + A_{22}\cdot B_{21}$

$C_{22} = A_{21}\cdot B_{12} + A_{22}\cdot B_{22}$

| 30 | 44 | 35 | 26 |
|---|---|---|---|
| 67 | 74 | 85 | 52 |
| **36** | **47** | 43 | 32 |
| **16** | **28** | 15 | 16 |

C = A·B

3. Multiplicar Matrices

1 — 2 — 3

72

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

$A_{11}$ $A_{12}$ $A_{21}$ $A_{22}$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

$B_{11}$ $B_{12}$ $B_{21}$ $B_{22}$

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

$C_{11}=A_{11}\cdot B_{11} + A_{12}\cdot B_{21}$

$C_{12} = A_{11}\cdot B_{12} + A_{12}\cdot B_{22}$

$C_{21}= A_{21}\cdot B_{11} + A_{22}\cdot B_{21}$

$C_{22} = A_{21}\cdot B_{12} + A_{22}\cdot B_{22}$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

C = A·B



73

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

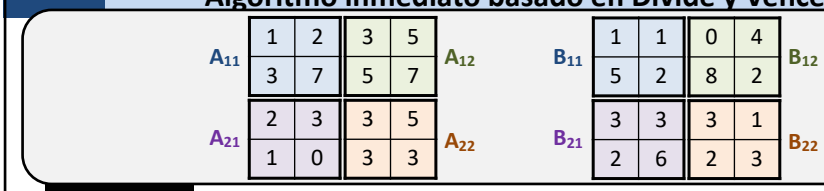
B



74

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás



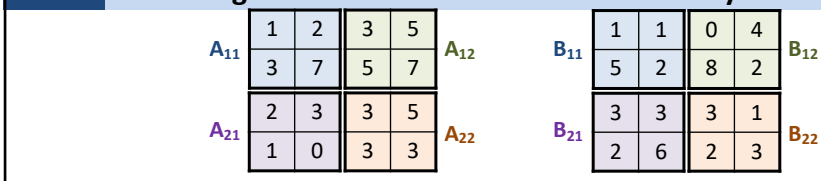

**Fase Dividir**



75

## Multiplicar Matrices

### Algoritmo inmediato basado en Divide y Vencerás



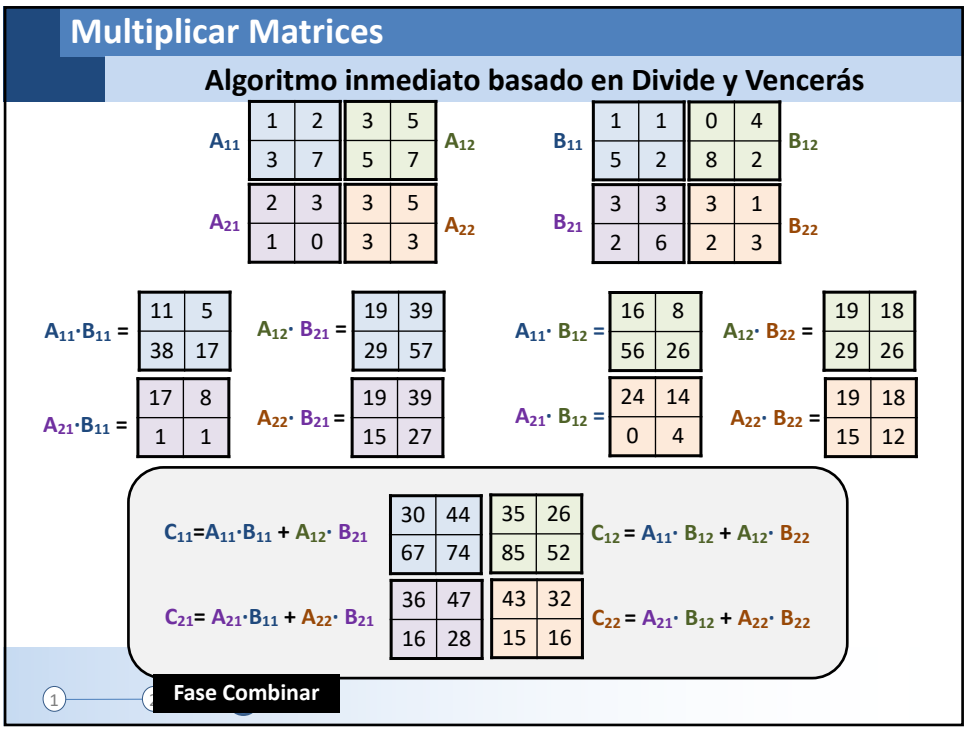

$A_{11} \cdot B_{11} = \begin{bmatrix} 11 & 5 \\ 38 & 17 \end{bmatrix}$ $A_{12} \cdot B_{21} = \begin{bmatrix} 19 & 39 \\ 29 & 57 \end{bmatrix}$ $A_{11} \cdot B_{12} = \begin{bmatrix} 16 & 8 \\ 56 & 26 \end{bmatrix}$ $A_{12} \cdot B_{22} = \begin{bmatrix} 19 & 18 \\ 29 & 26 \end{bmatrix}$

$A_{21} \cdot B_{11} = \begin{bmatrix} 17 & 8 \\ 1 & 1 \end{bmatrix}$ $A_{22} \cdot B_{21} = \begin{bmatrix} 19 & 39 \\ 15 & 27 \end{bmatrix}$ $A_{21} \cdot B_{12} = \begin{bmatrix} 24 & 14 \\ 0 & 4 \end{bmatrix}$ $A_{22} \cdot B_{22} = \begin{bmatrix} 19 & 18 \\ 15 & 12 \end{bmatrix}$

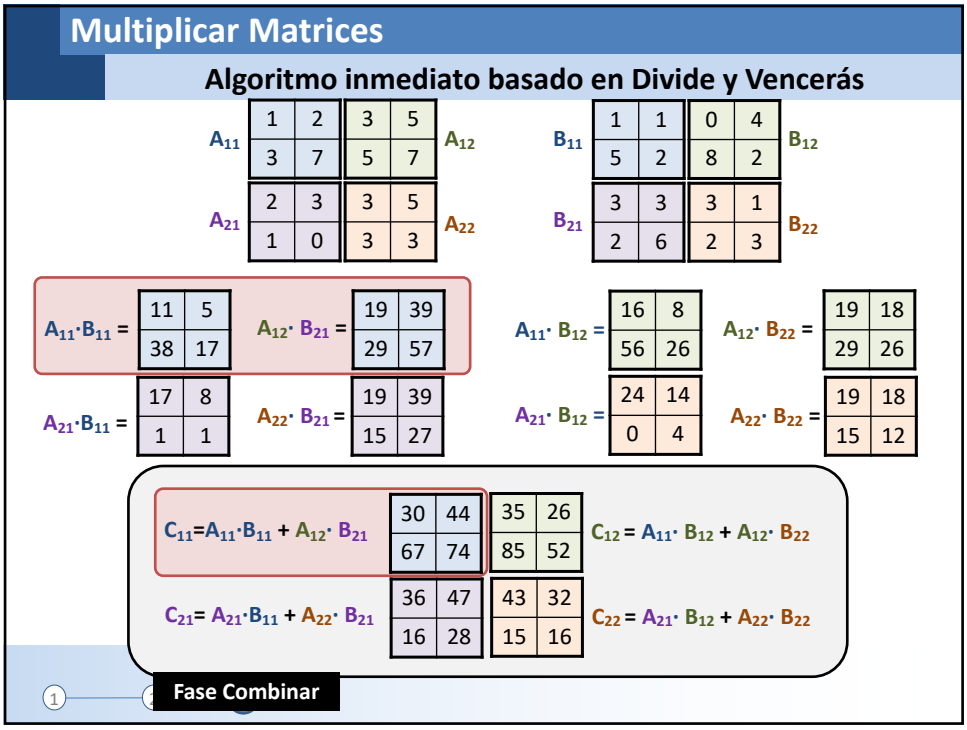
**Fase Conquistar**



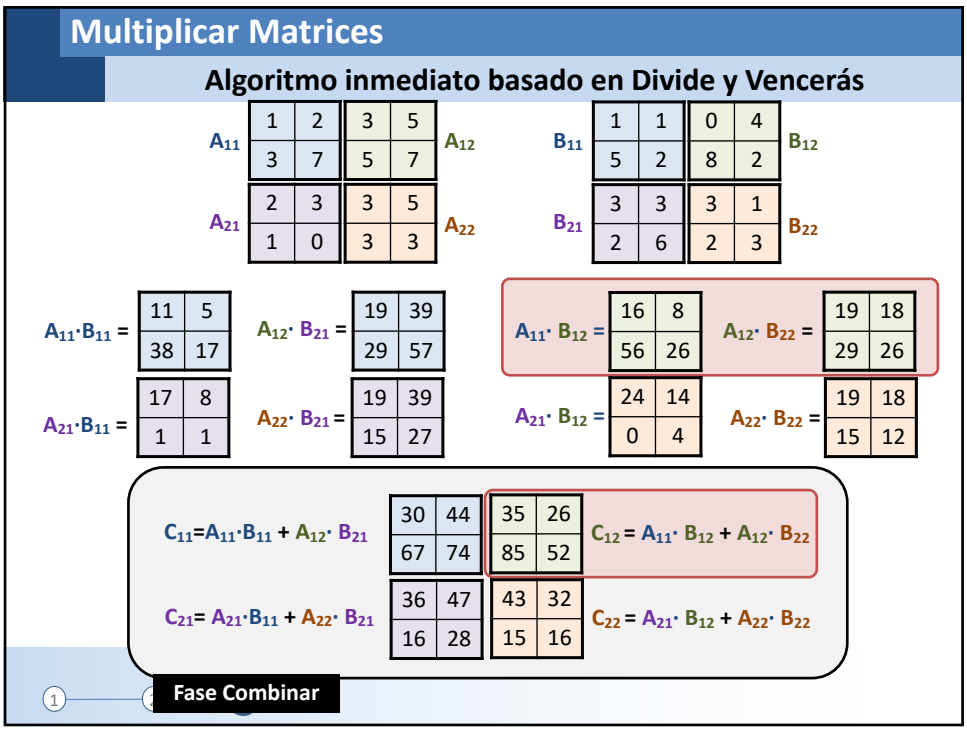
76


Multiplicar Matrices
Algoritmo inmediato basado en Divide y Vencerás
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
Fase Combinar

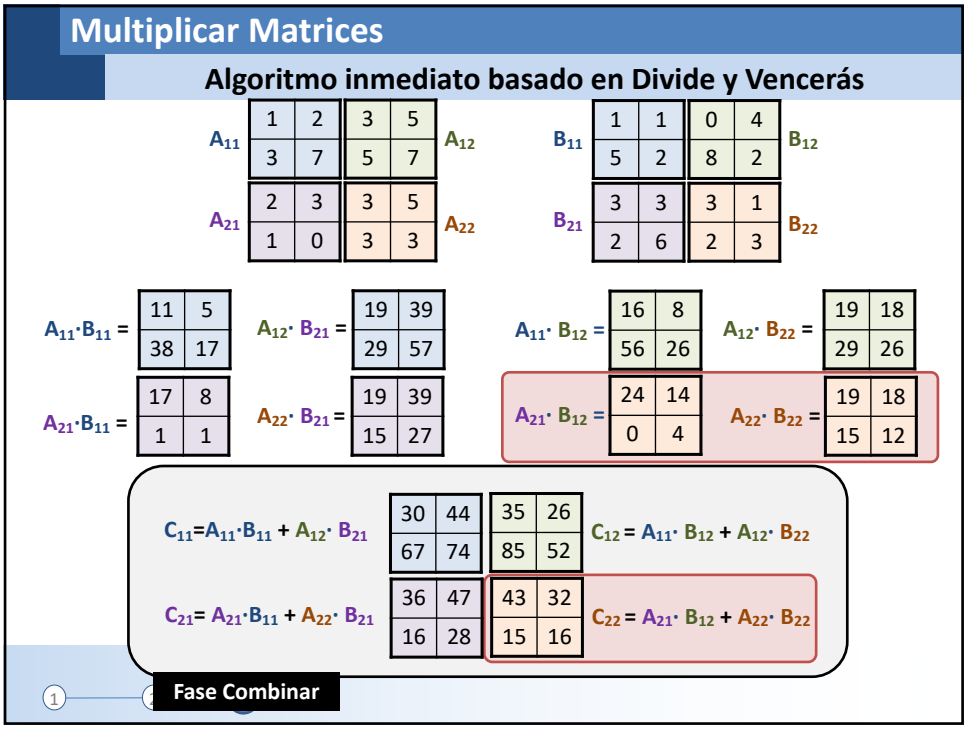
77


Multiplicar Matrices
Algoritmo inmediato basado en Divide y Vencerás
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
Fase Combinar

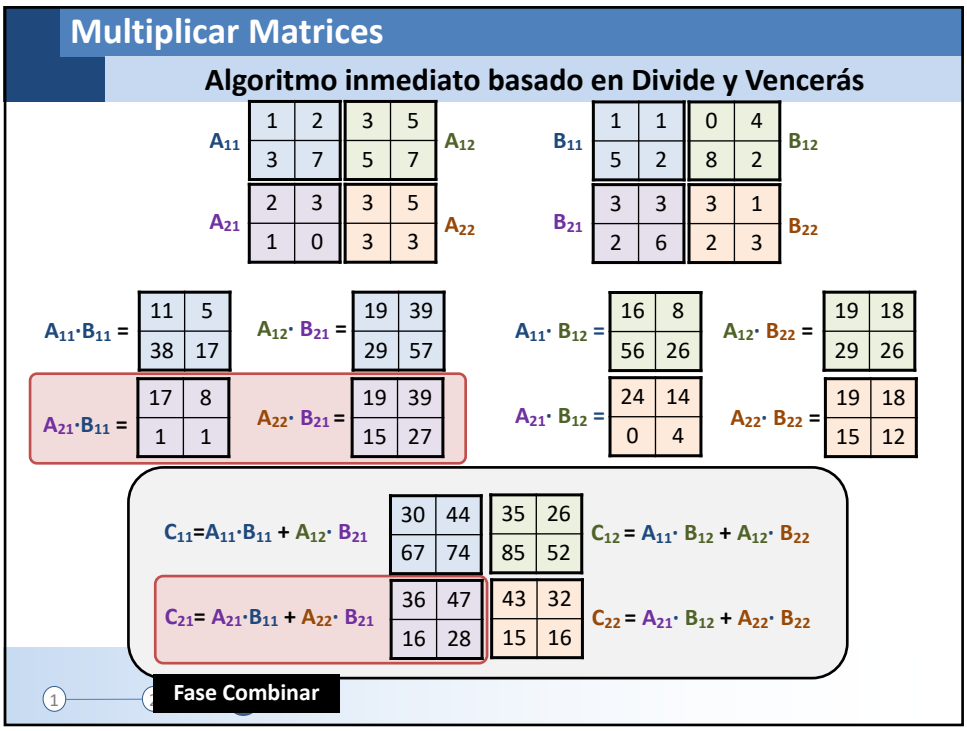
78


Multiplicar Matrices
Algoritmo inmediato basado en Divide y Vencerás
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
Fase Combinar

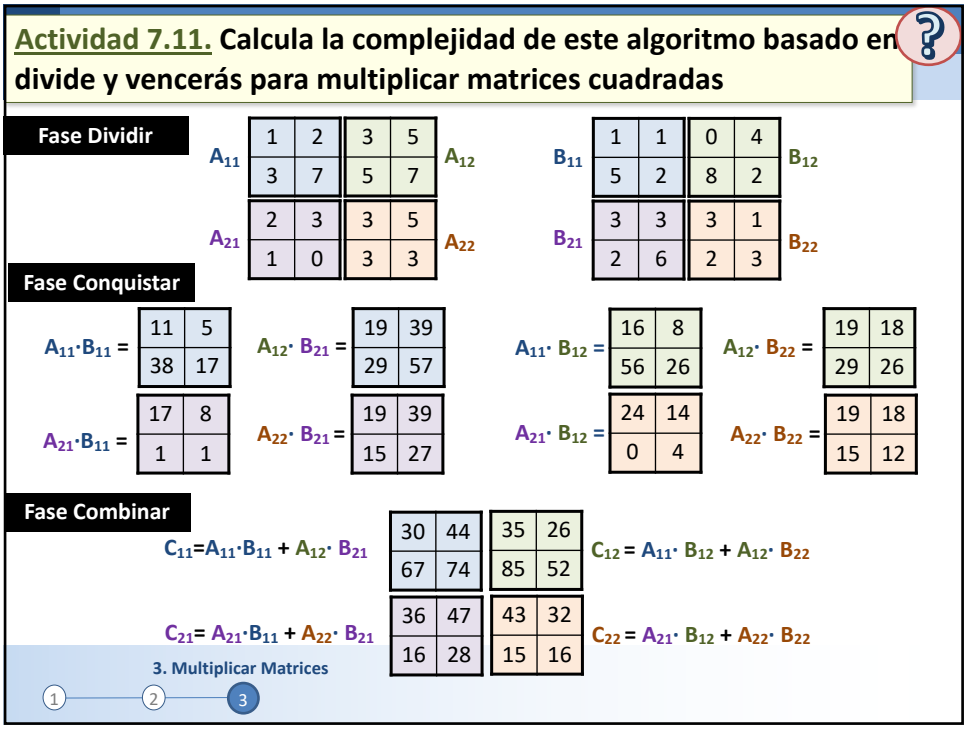
79


Multiplicar Matrices
Algoritmo inmediato basado en Divide y Vencerás
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
Fase Combinar

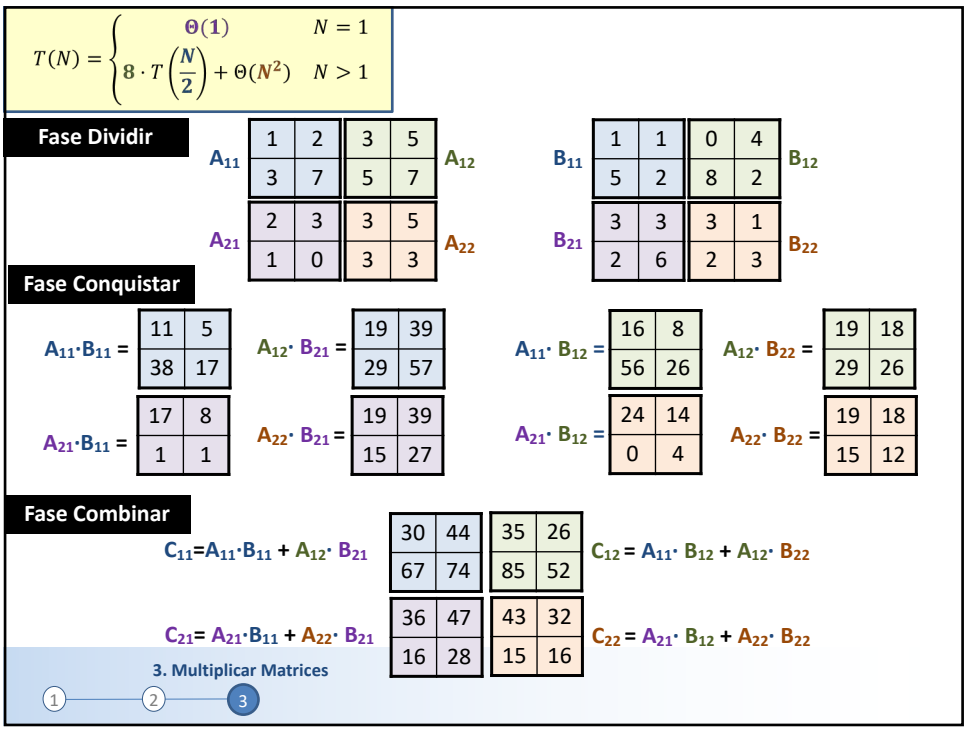
80


Multiplicar Matrices
Algoritmo inmediato basado en Divide y Vencerás
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
Fase Combinar
1
2

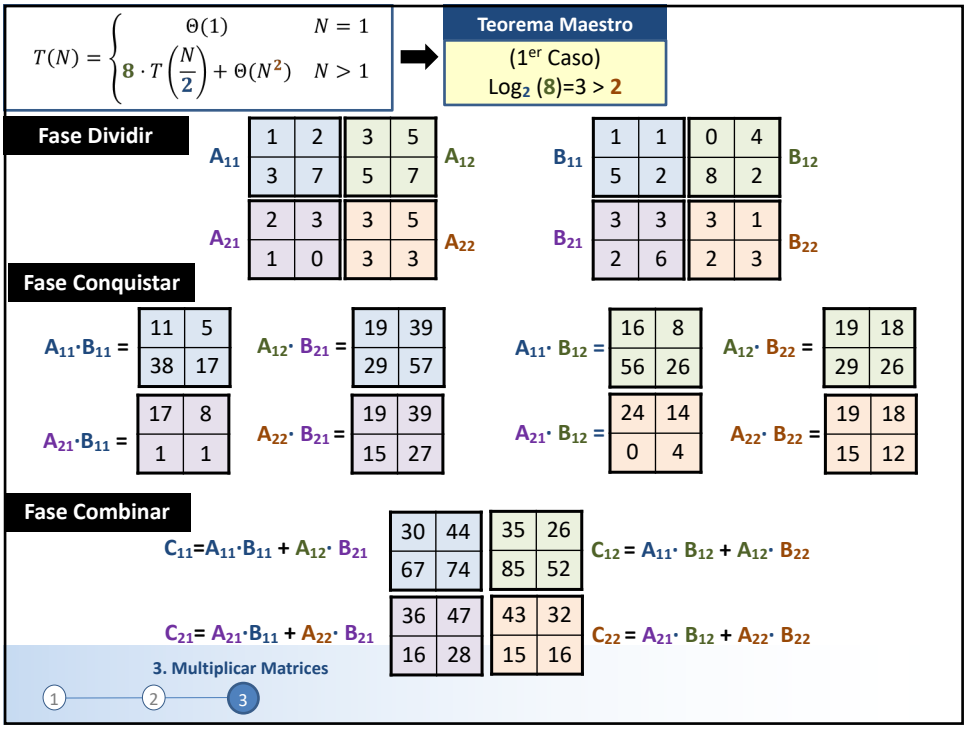
81


Actividad 7.11. Calcula la complejidad de este algoritmo basado en divide y vencerás para multiplicar matrices cuadradas
Fase Dividir
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
Fase Conquistar
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
Fase Combinar
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
3. Multiplicar Matrices
1
2
3

82


$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 8 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$
Fase Dividir
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
Fase Conquistar
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
Fase Combinar
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
3. Multiplicar Matrices
1 2 3

83


$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 8 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$
Teorema Maestro
(1er Caso)
Log2 (8)=3 > 2
Fase Dividir
A11 1 2 3 7
A12 3 5 5 7
A21 2 3 1 0
A22 3 5 3 3
B11 1 1 5 2
B12 0 4 8 2
B21 3 3 2 6
B22 3 1 2 3
Fase Conquistar
A11·B11 = 11 5 38 17
A12· B21 = 19 39 29 57
A11· B12 = 16 8 56 26
A12· B22 = 19 18 29 26
A21·B11 = 17 8 1 1
A22· B21 = 19 39 15 27
A21· B12 = 24 14 0 4
A22· B22 = 19 18 15 12
Fase Combinar
C11=A11·B11 + A12· B21 30 44 67 74
C12 = A11· B12 + A12· B22 35 26 85 52
C21= A21·B11 + A22· B21 36 47 16 28
C22 = A21· B12 + A22· B22 43 32 15 16
3. Multiplicar Matrices
1 2 3

84

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 8 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$$

**Teorema Maestro**
(1er Caso)
$\text{Log}_2(8)=3 > 2$

**Complejidad: $\Theta(N^3)$**

NO MEJORA EL ALGORTIMO NATURAL

**Fase Dividir**

$A_{11}$ | 1 | 2 | 3 | 5 |
|---|---|---|---|---|
| | 3 | 7 | 5 | 7 |

$A_{12}$

$A_{21}$ | 2 | 3 | 3 | 5 |
|---|---|---|---|---|
| | 1 | 0 | 3 | 3 |

$A_{22}$

$B_{11}$ | 1 | 1 |
|---|---|---|
| | 5 | 2 |

$B_{12}$

$B_{21}$ | 3 | 3 | 3 | 1 |
|---|---|---|---|---|
| | 2 | 6 | 2 | 3 |

$B_{22}$

**Fase Conquistar**

$A_{11} \cdot B_{11} =$ | 11 | 5 |
|---|---|---|
| | 38 | 17 |

$A_{12} \cdot B_{21} =$ | 19 | 39 |
|---|---|---|
| | 29 | 57 |

$A_{11} \cdot B_{12} =$ | 16 | 8 |
|---|---|---|
| | 56 | 26 |

$A_{12} \cdot B_{22} =$ | 19 | 18 |
|---|---|---|
| | 29 | 26 |

$A_{21} \cdot B_{11} =$ | 17 | 8 |
|---|---|---|
| | 1 | 1 |

$A_{22} \cdot B_{21} =$ | 19 | 39 |
|---|---|---|
| | 15 | 27 |

$A_{21} \cdot B_{12} =$ | 24 | 14 |
|---|---|---|
| | 0 | 4 |

$A_{22} \cdot B_{22} =$ | 19 | 18 |
|---|---|---|
| | 15 | 12 |

**Fase Combinar**

$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$ | 30 | 44 |
|---|---|---|
| | 67 | 74 |

| 35 | 26 |
|---|---|
| 85 | 52 |

$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$

$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$ | 36 | 47 |
|---|---|---|
| | 16 | 28 |

| 43 | 32 |
|---|---|
| 15 | 16 |

$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$

3. Multiplicar Matrices

1 — 2 — 3

85

## Multiplicar Matrices

**Fase Dividir**

$A_{11}$ | 1 | 2 | 3 | 5 |
|---|---|---|---|---|
| | 3 | 7 | 5 | 7 |

$A_{12}$

$A_{21}$ | 2 | 3 | 3 | 5 |
|---|---|---|---|---|
| | 1 | 0 | 3 | 3 |

$A_{22}$

$B_{11}$ | 1 | 1 | 0 | 4 |
|---|---|---|---|---|
| | 5 | 2 | 8 | 2 |

$B_{12}$

$B_{21}$ | 3 | 3 | 3 | 1 |
|---|---|---|---|---|
| | 2 | 6 | 2 | 3 |

$B_{22}$

**Fase Conquistar**

$A_{11} \cdot B_{11} =$ | 11 | 5 |
|---|---|---|
| | 38 | 17 |

$A_{12} \cdot B_{21} =$ | 19 | 39 |
|---|---|---|
| | 29 | 57 |

$A_{11} \cdot B_{12} =$ | 16 | 8 |
|---|---|---|
| | 56 | 26 |

$A_{12} \cdot B_{22} =$ | 19 | 18 |
|---|---|---|
| | 29 | 26 |

$A_{21} \cdot B_{11} =$ | 17 | 8 |
|---|---|---|
| | 1 | 1 |

$A_{22} \cdot B_{21} =$ | 19 | 39 |
|---|---|---|
| | 15 | 27 |

$A_{21} \cdot B_{12} =$ | 24 | 14 |
|---|---|---|
| | 0 | 4 |

$A_{22} \cdot B_{22} =$ | 19 | 18 |
|---|---|---|
| | 15 | 12 |

8 Multiplicaciones

**Fase Combinar**

$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$ | 30 | 44 |
|---|---|---|
| | 67 | 74 |

| 35 | 26 |
|---|---|
| 85 | 52 |

$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$

$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$ | 36 | 47 |
|---|---|---|
| | 16 | 28 |

| 43 | 32 |
|---|---|
| 15 | 16 |

$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$

3. Multiplicar Matrices

1 — 2 — 3

86

## Multiplicar Matrices

### Algoritmo Strassen

**Fase Dividir**

$A_{11}$ | $A_{12}$ | $B_{11}$ | $B_{12}$ | $A_{21}$ | $A_{22}$ | $B_{21}$ | $B_{22}$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

**Fase Conquistar**

$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \cdot B_{11}$

$M_3 = A_{11} \cdot (B_{12} - B_{22})$

$M_4 = A_{22} \cdot (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \cdot B_{22}$

$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

7 Multiplicaciones

**Fase Combinar**

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

3. Multiplicar Matrices

1 — 2 — 3

87

## Multiplicar Matrices

**Actividad 7.12.** Calcula la complejidad del algoritmo de Strassen

**Fase Dividir**

$A_{11}$ | $A_{12}$ | $B_{11}$ | $B_{12}$ | $A_{21}$ | $A_{22}$ | $B_{21}$ | $B_{22}$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

**Fase Conquistar**

$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \cdot B_{11}$

$M_3 = A_{11} \cdot (B_{12} - B_{22})$

$M_4 = A_{22} \cdot (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \cdot B_{22}$

$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

**Fase Combinar**

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

3. Multiplicar Matrices

1 — 2 — 3

88

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 7 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$$

**Fase Dividir**

$A_{11}$ | $A_{12}$ | $A_{21}$ | $A_{22}$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

$B_{11}$ | $B_{12}$ | $B_{21}$ | $B_{22}$

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

**Fase Conquistar**

$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \cdot B_{11}$

$M_3 = A_{11} \cdot (B_{12} - B_{22})$

$M_4 = A_{22} \cdot (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \cdot B_{22}$

$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

**Fase Combinar**

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

3. Multiplicar Matrices

1 — 2 — 3

89

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 7 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$$

**Teorema del Maestro**

($1^{er}$ Caso)

$\text{Log}_2(7) = 2.80 > 2$

**Fase Dividir**

$A_{11}$ | $A_{12}$ | $A_{21}$ | $A_{22}$

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

A

$B_{11}$ | $B_{12}$ | $B_{21}$ | $B_{22}$

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

B

**Fase Conquistar**

$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \cdot B_{11}$

$M_3 = A_{11} \cdot (B_{12} - B_{22})$

$M_4 = A_{22} \cdot (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \cdot B_{22}$

$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

**Fase Combinar**

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

3. Multiplicar Matrices

1 — 2 — 3

90

$$T(N) = \begin{cases} \Theta(1) & N = 1 \\ 7 \cdot T\left(\frac{N}{2}\right) + \Theta(N^2) & N > 1 \end{cases}$$

**Teorema del Maestro**

(1er Caso)

$\text{Log}_2(7) = 2.80 > 2$

**Complejidad: $\Theta(N^{2.80})$**

**Fase Dividir**

$A_{11}$ | $A_{12}$ | $A_{21}$ | $A_{22}$ — A

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 3 | 7 | 5 | 7 |
| 2 | 3 | 3 | 5 |
| 1 | 0 | 3 | 3 |

$B_{11}$ | $B_{12}$ | $B_{21}$ | $B_{22}$ — B

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 4 |
| 5 | 2 | 8 | 2 |
| 3 | 3 | 3 | 1 |
| 2 | 6 | 2 | 3 |

**Fase Conquistar**

$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) \cdot B_{11}$

$M_3 = A_{11} \cdot (B_{12} - B_{22})$

$M_4 = A_{22} \cdot (B_{21} - B_{11})$

$M_5 = (A_{11} + A_{12}) \cdot B_{22}$

$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

**Fase Combinar**

$C_{11} = M_1 + M_4 - M_5 + M_7$

$C_{12} = M_3 + M_5$

$C_{21} = M_2 + M_4$

$C_{22} = M_1 - M_2 + M_3 + M_6$

| | | | |
|---|---|---|---|
| 30 | 44 | 35 | 26 |
| 67 | 74 | 85 | 52 |
| 36 | 47 | 43 | 32 |
| 16 | 28 | 15 | 16 |

3. Multiplicar Matrices

1 — 2 — 3

91