

DIVIDE Y VENCERÁS

Aránzazu Jurío

ALGORITMIA

2018/2019

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Técnica Divide y Vencerás

- Resolver un problema a partir de la solución de subproblemas del mismo tipo, pero de menor tamaño.
- Si los subproblemas son todavía relativamente grandes, volver a aplicar la técnica hasta que los subproblemas se puedan resolver directamente



RECURSIVIDAD

Técnica Divide y Vencerás

- Pasos para solucionar un problema:
 - Plantear el problema de forma que pueda **descomponerse** en k subproblemas del mismo tipo pero de menor tamaño ($0 \leq n_k \leq n$)
 - **Resolver independientemente** todos los subproblemas, bien directamente si son elementales o de forma recursiva. El hecho de que el tamaño de los subproblemas sea estrictamente menor que el tamaño original, garantiza la convergencia hacia el caso base
 - **Combinar** las soluciones obtenidas para construir la solución del problema original

Técnica Divide y Vencerás

algoritmo divideYVenceras(ENT x:problema; SAL z:resultado)

variables

y : problema

i, k : entero

principio

si esCasoBase(x) **entonces**

z ← resolveCasoBase(x)

si no

subproblemas ← divide(x,k)

para i = 1 **a** k **hacer**

subsoluciones[i] ← divideYVenceras(subproblemas[i])

fin para

z ← combina(subsoluciones)

fin si

devolver z

fin

Técnica Divide y Vencerás

- Los subproblemas deben ser independientes (que no exista solapamiento entre ellos), para que el tiempo de ejecución no sea exponencial
- Contraejemplo Fibonacci

Técnica Divide y Vencerás

- El número k de subproblemas debe ser pequeño, e independiente de la entrada
- Cuando $k = 1$, los llamamos **algoritmos de simplificación**
 - Tiempos de ejecución muy buenos (orden logarítmico o lineal)
 - Puede sustituirse la recursión por un bucle iterativo
 - Menor complejidad espacial (no utiliza pila de recursión)
 - Menor legibilidad y sencillez de código

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Multiplicación de grandes enteros

- Queremos multiplicar dos números de n cifras cada uno de ellos, siendo n potencia de 2
- La multiplicación tradicional tiene orden $O(n^2)$
- ¿Se puede mejorar?

Multiplicación de grandes enteros

- Queremos multiplicar uv
- Descomponemos cada número en dos partes
 - $u = a10^{n/2} + b$
 - $v = c10^{n/2} + d$
- La multiplicación queda
 - $uv = \left(a10^{n/2} + b\right)\left(c10^{n/2} + d\right) = ac10^n + (ad + bc)10^{n/2} + bd$
 - Esta complejidad sigue siendo $O(n^2)$
- Se puede mejorar
 - $ad + bc = (a + b)(c + d) - ac - bd$

Multiplicación de grandes enteros

- Para calcular un producto de n cifras, necesitamos tres productos de $n/2$ cifras
 - $T(n) = 3T\left(\frac{n}{2}\right) + kn$
- Complejidad $O(n^{\log_2 3}) = O(n^{1,59})$

Multiplicación de grandes enteros

- Ejemplo: $1963 * 3521$
 - $u = 1963 = 19 * 10^2 + 63$
 - $v = 3521 = 35 * 10^2 + 21$
 - $ac = 19 * 35$
 - $(a + b)(c + d) = 82 * 56$
 - $bd = 63 * 21$
- Recursivamente, obtenemos la solución

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Búsqueda binaria

- Dado un vector ordenado de números enteros no repetidos, queremos conocer si un elemento x está en él o no. Además, si está, queremos conocer su posición

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

- ¿Está el 10? → Posición 5
- ¿Está el 8? → No

Búsqueda binaria

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

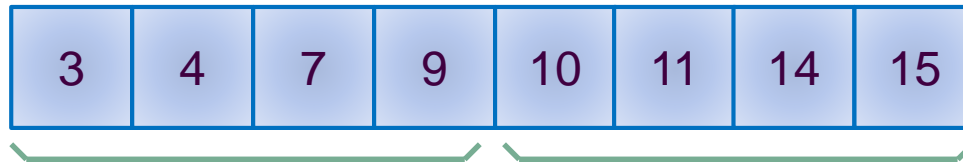
- Posible solución
 - Recorrer el vector desde la primera posición
 - hasta que encontremos el elemento
 - hasta encontrar un elemento mayor que el buscado
 - hasta finalizar el vector
 - Complejidad $O(n)$ siendo n el tamaño del vector
- ¿Hay alguna solución mejor?

Búsqueda binaria

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

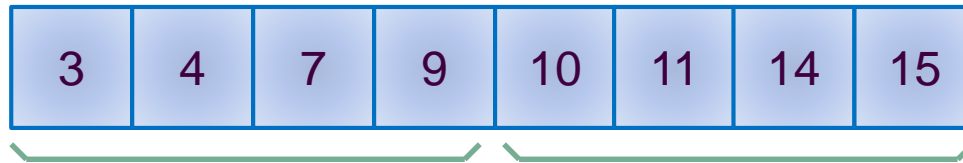
- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Dividir el vector de entrada en dos partes iguales?
 - ¿Dividir el vector de entrada en dos partes desiguales?
 - ¿Dividir el vector de entrada en tres partes iguales?
 - ...

Búsqueda binaria



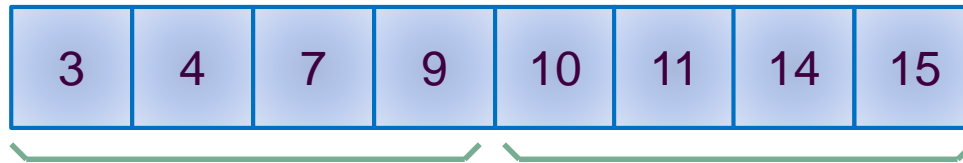
- Paso 2 – Divide y vencerás: resolver cada uno de los subproblemas
 - Buscar el elemento en la primera mitad del vector
 - Buscar el elemento en la segunda mitad del vector
- Paso 3 – Divide y vencerás: combinar las soluciones de los subproblemas
 - Si el elemento está en la primera mitad, devolver su posición
 - Si el elemento está en la segunda mitad, devolver su posición
 - Si el elemento no está en la primera mitad ni en la segunda mitad, devolver “no está”

Búsqueda binaria



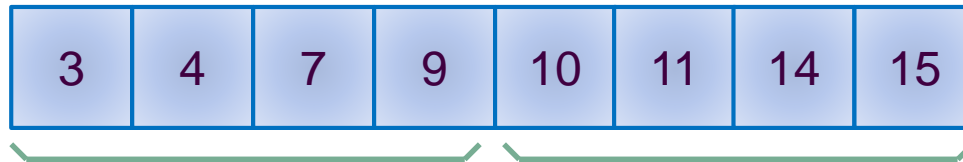
- ¿Se puede mejorar?

Búsqueda binaria



- El elemento sólo puede estar en una de las dos mitades
- No es necesario ejecutar los dos subproblemas
- ¿Cómo saber en qué mitad hay que buscar?

Búsqueda binaria



- El elemento sólo puede estar en una de las dos mitades
- No es necesario ejecutar los dos subproblemas
- ¿Cómo saber en qué mitad hay que buscar?
 - Si el elemento a buscar es menor que el central, buscar en la primera mitad
 - Si el elemento a buscar es mayor que el central, buscar en la segunda mitad
- Como en nuestra función sólo ejecutamos un subproblema, la búsqueda binaria es un ejemplo de problema de simplificación

Búsqueda binaria

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

- ¿Cuál es el caso base, con el que terminar la recursividad?

Búsqueda binaria

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

- ¿Cuál es el caso base, con el que terminar la recursividad?
 - Buscamos en una tabla de un solo elemento
 - El elemento a buscar está en la posición central de la tabla

Búsqueda binaria

Busco el 10

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

3	4	7	9	10	11	14	15
---	---	---	---	----	----	----	----

Búsqueda binaria

- Complejidad $O(\log n)$
- ¿Mejora si dividimos el vector en tres subproblemas?
 - $O(\log n)$

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Mediana de dos vectores

- Sean X e Y dos vectores de enteros ordenados, cada uno de n elementos. Queremos calcular la mediana de los $2n$ elementos que contienen X e Y

X	2	6	7	11	13	15	16	17	21
Y	1	2	8	9	10	14	15	18	20

- Mediana $\rightarrow 11$ (si son pares seleccionamos el elemento que deja $n/2-1$ elementos menores y $n/2$ elementos mayores)

Mediana de dos vectores

- Posible solución: ordenar todos los elementos en un único vector, y seleccionar su elemento central

X	2	6	7	11	13	15	16	17	21
---	---	---	---	----	----	----	----	----	----

Y	1	2	8	9	10	14	15	18	20
---	---	---	---	---	----	----	----	----	----

1	2	2	6	7	8	9	10	11	13	14	15	15	16	17	18	20	21
---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----



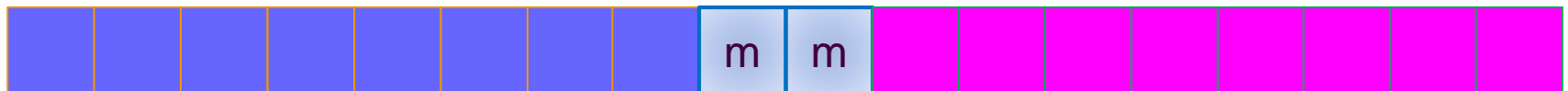
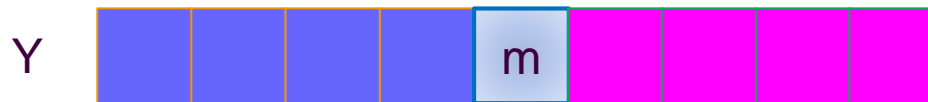
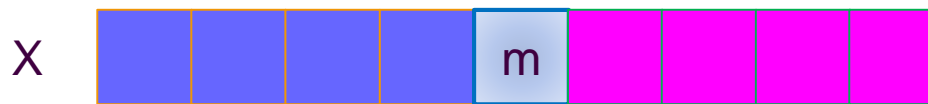
- Aunque sólo ordenemos los n primeros elementos, la complejidad es $O(n)$
- ¿Hay alguna solución mejor?

Mediana de dos vectores

- Divide y vencerás. Idea basada en el problema de búsqueda binaria
 - ¿Cuál es el caso base?

Mediana de dos vectores

- Divide y vencerás. Idea basada en el problema de búsqueda binaria
 - ¿Cuál es el caso base?
 - Si los vectores tienen un solo elemento, seleccionamos como mediana el mínimo de los dos
 - Si la mediana de los dos vectores es igual, coincide con la mediana del vector que incluye todos los elementos

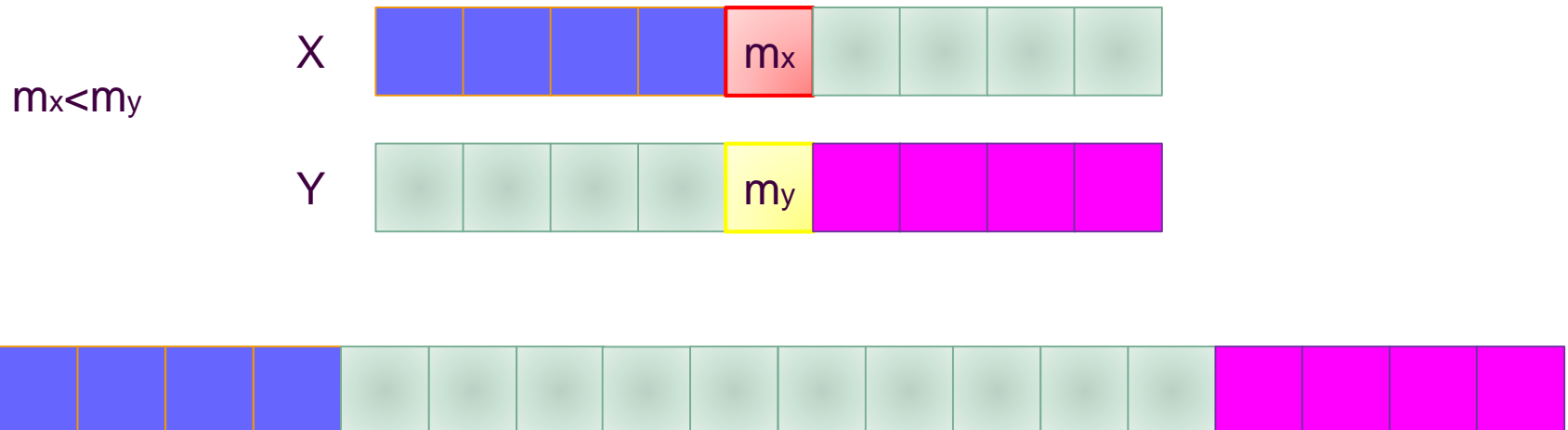


Mediana de dos vectores

- Divide y vencerás. Idea basada en el problema de búsqueda binaria
 - ¿Cómo podemos dividir las tablas para calcular los subproblemas?
 - Dividimos cada vector a la mitad
 - ¿Es necesario ejecutar todos los subproblemas? (Estudiar las dos mitades de cada vector)

Mediana de dos vectores

- Divide y vencerás.
 - ¿Cómo podemos dividir las tablas para calcular los subproblemas?
 - Dividimos cada vector a la mitad
 - ¿Es necesario ejecutar todos los subproblemas? (Estudiar las dos mitades de cada vector)

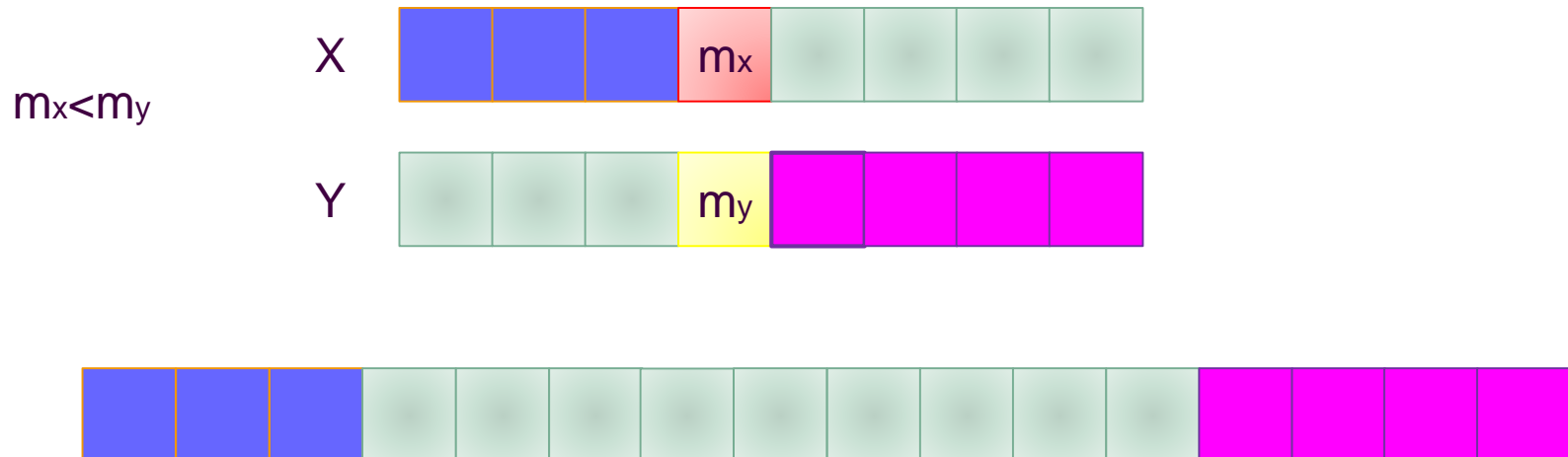


Mediana de dos vectores

- Divide y vencerás.
 - ¿Cómo podemos dividir las tablas para calcular los subproblemas?
 - Si la mediana del vector X (m_x) es menor que la mediana del vector Y (m_y), entonces la mediana global va a estar en $X[m_x..fin]$ o en $Y[principio..m_y]$
 - Si la mediana del vector X (m_x) es mayor que la mediana del vector Y (m_y), entonces la mediana global va a estar en $X[principio..m_x]$ o en $Y[m_y..fin]$

Mediana de dos vectores

- Divide y vencerás
 - Los dos vectores tienen que tener siempre el mismo número de elementos
 - ¿Pero y si el número de elementos es par?



Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 0, 7, Y, 0, 7)`

- `posmedx = 3` `X[posmedx] = 13`
- `posmedy = 3` `Y[posmedy] = 10`
- `numelem = ?`

Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 0, 7, Y, 0, 7)`

- `posmedx = 3` `X[posmedx] = 13`
- `posmedy = 3` `Y[posmedy] = 10`
- `numelem = 4`

Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 0, 3, Y, 4, 7)`

- `posmedx = 1` `X[posmedx] = 3`
- `posmedy = 5` `Y[posmedy] = 17`
- `numelem = 2`

Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 0, 3, Y, 4, 7)`

- `posmedx = 1` `X[posmedx] = 3`
- `posmedy = 5` `Y[posmedy] = 17`
- `numelem = 2`

Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 2, 3, Y, 4, 5)`

- `posmedx = 2` `X[posmedx] = 5`
- `posmedy = 4` `Y[posmedy] = 11`
- `numelem = 1`


Mediana de dos vectores

- Ejemplo

	0	1	2	3	4	5	6	7
X	2	3	5	13	14	16	18	23
Y	4	6	8	10	11	17	20	21

- `mediana(X, 3, 3, Y, 4, 4)`
 - caso base
 - $X[3] > Y[4] \rightarrow \text{mediana} = 11$

Mediana de dos vectores

- Complejidad
 - $T(2n) = T(n) + cte$
- 
- $T(n) \in O(\log n)$

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Buscar el elemento en su posición

- Sea $a[1..n]$ un vector ordenado de enteros todos distintos. Encontrar un índice i tal que $1 \leq i \leq n$ y $a[i] = i$

-1	1	3	5	6	8	9	10
----	---	---	---	---	---	---	----

- $a[3] = 3$

Buscar el elemento en su posición

- Sea $a[1..n]$ un vector ordenado de enteros todos distintos. Encontrar un índice i tal que $1 \leq i \leq n$ y $a[i] = i$

-1	1	3	5	6	8	9	10
----	---	---	---	---	---	---	----

- $a[3] = 3$
- Posible solución
 - Recorrer el vector desde la primera posición
 - hasta que encontremos el elemento en su posición
 - hasta encontrar un elemento mayor que su posición
 - hasta finalizar el vector
 - Complejidad $O(n)$ siendo n el tamaño del vector

Buscar el elemento en su posición

- Sea $a[1..n]$ un vector ordenado de enteros todos distintos. Encontrar un índice i tal que $1 \leq i \leq n$ y $a[i] = i$

-1	1	3	5	6	8	9	10
----	---	---	---	---	---	---	----

- $a[3] = 3$
- Solución utilizando divide y vencerás

Buscar el elemento en su posición

1	2	3	4	5	6	7	8
-1	1	3	5	6	8	9	10
-1	1	3	5	6	8	9	10
-1	1	3	5	6	8	9	10
-1	1	3	5	6	8	9	10

Índice

- Técnica Divide y vencerás
- **Problemas**
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - **Matriz cuadrada transpuesta**
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Matriz cuadrada transpuesta

- Dada una matriz cuadrada de dimensiones $n \times n$ (siendo n potencia de 2), diseñar un algoritmo para obtener su matriz transpuesta.

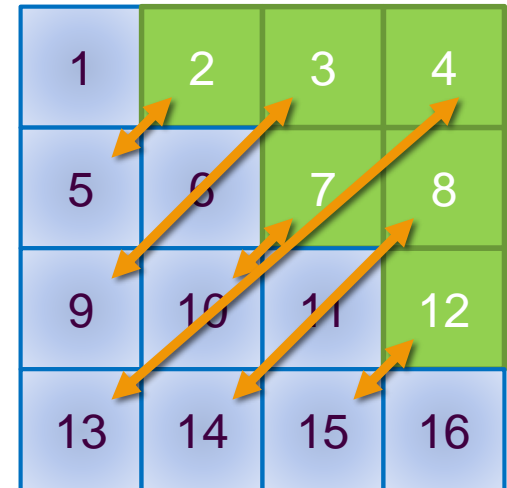
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Matriz cuadrada transpuesta

- Primera solución
 - Recorrer la mitad superior (o inferior)
 - Intercambiar cada elemento por su correspondiente en la otra mitad

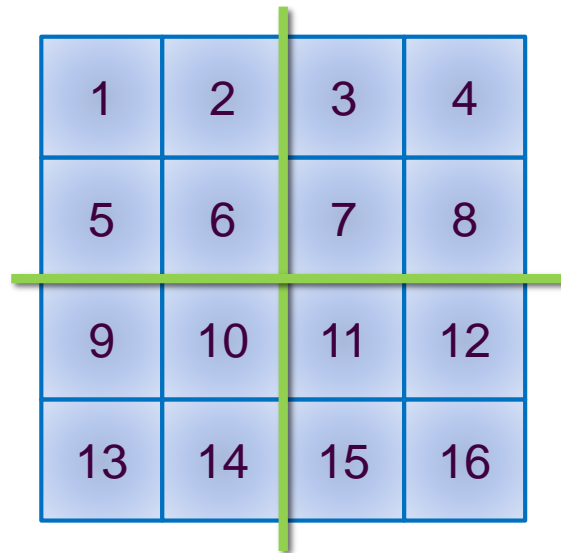


Matriz cuadrada transpuesta

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cuál es la mejor división? Pocos subproblemas y del mismo tamaño todos ellos

Matriz cuadrada transpuesta

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cuál es la mejor división? Pocos subproblemas y del mismo tamaño todos ellos

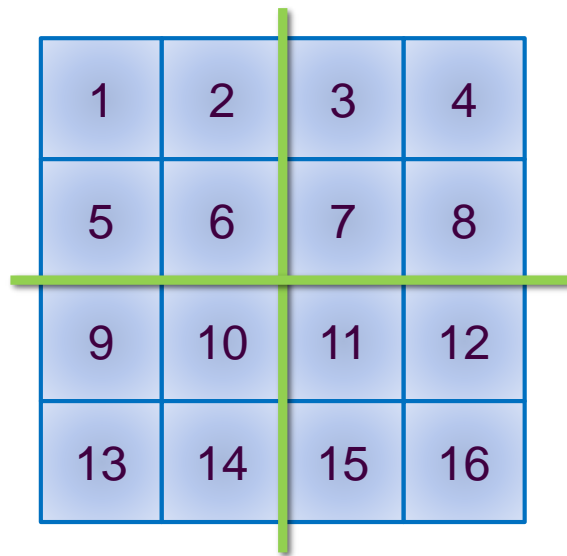


1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

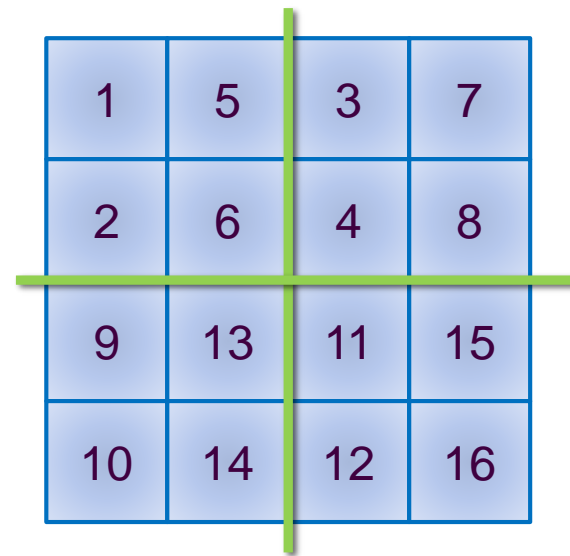
- ¿Cuál es el caso base? ¿Cuándo una matriz es igual a su transpuesta?

Matriz cuadrada transpuesta

- Paso 2 – Divide y vencerás: resolver cada uno de los subproblemas
 - Como la aplicación es recursiva, damos por hecho que en este paso cada una de las submatrices pasa a estar transpuesta



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



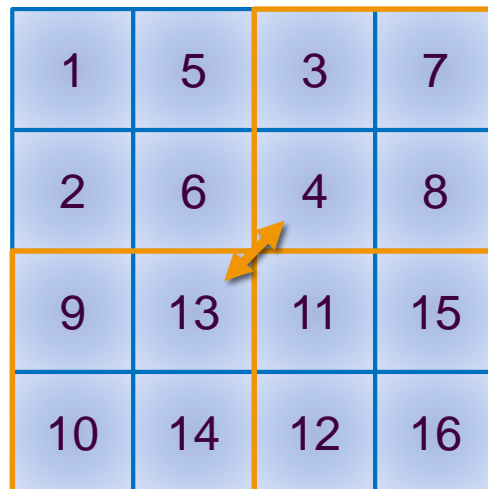
1	5	3	7
2	6	4	8
9	13	11	15
10	14	12	16

Matriz cuadrada transpuesta

- Paso 3 – Divide y vencerás: combinar las soluciones de los subproblemas
 - ¿Qué nos falta hacer para obtener la solución al problema original?

Matriz cuadrada transpuesta

- Paso 3 – Divide y vencerás: combinar las soluciones de los subproblemas
 - ¿Qué nos falta hacer para obtener la solución al problema original?
 - ¡¡Intercambiar los cuadrantes 2 y 3!!



1	5	3	7
2	6	4	8
9	13	11	15
10	14	12	16

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Subsecuencia de suma máxima

- Dado un vector de números, el problema de la subsecuencia de suma máxima consiste en encontrar una porción del vector cuya suma sea máxima dentro de un vector

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

5	-1	4	3
---	----	---	---

=11

6	-2	5
---	----	---

=9

6	-2	5	-1	4	3
---	----	---	----	---	---

=15

Subsecuencia de suma máxima

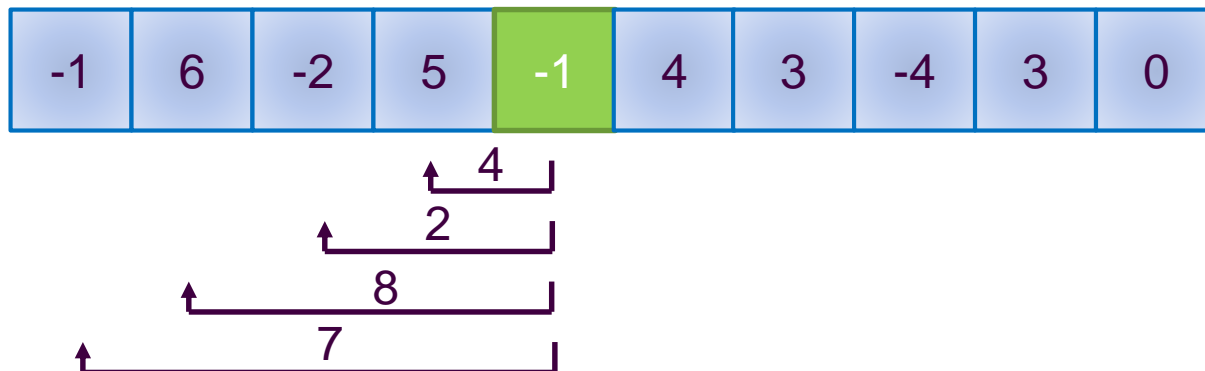
- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - Por elementos de la mitad derecha
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central

Subsecuencia de suma máxima

- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad derecha
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central

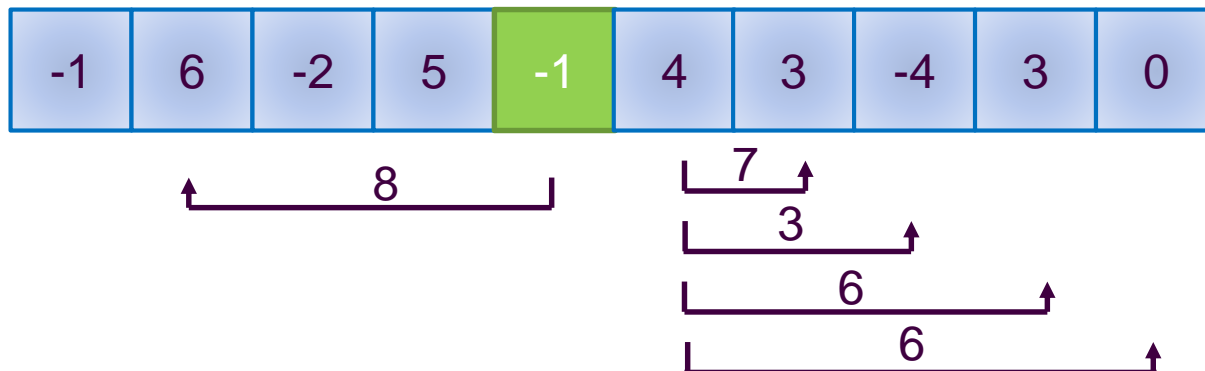
Subsecuencia de suma máxima

- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad derecha
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central



Subsecuencia de suma máxima

- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad derecha
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central



Subsecuencia de suma máxima

- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad derecha
 - Solucionar recursivamente el problema con este nuevo vector
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central



Subsecuencia de suma máxima

- Divido el vector en dos mitades
- La subsecuencia de suma máxima puede estar formada:
 - Por elementos de la mitad izquierda
 - 9
 - Por elementos de la mitad derecha
 - 7
 - Por elementos de la mitad izquierda y la derecha, incluyendo el elemento central
 - $8+7 = 15$

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

- Selecciono el máximo de las 3: 15

Subsecuencia de suma máxima

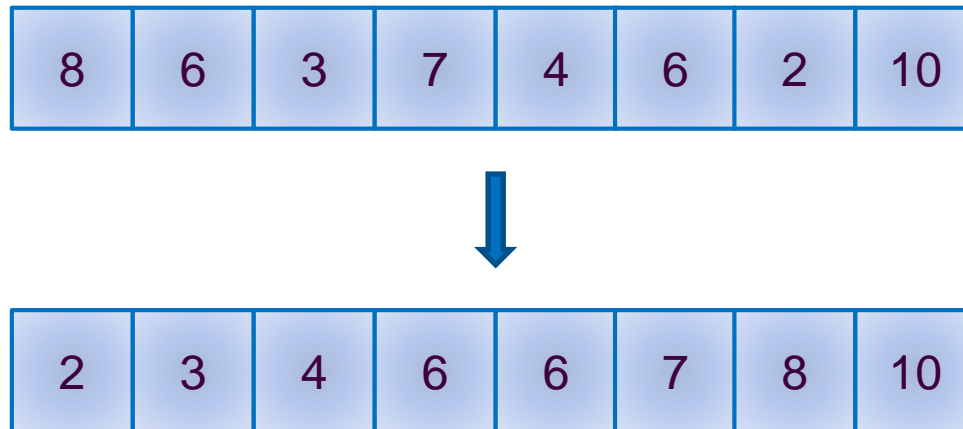
- Complejidad
 - Buscar la subsecuencia máxima de la mitad izquierda: $T\left(\frac{n}{2}\right)$
 - Buscar la subsecuencia máxima de la mitad derecha: $T\left(\frac{n}{2}\right)$
 - Buscar la subsecuencia máxima que incluye el elemento central: $\Theta(n)$
- Complejidad total: $\Theta(n \log n)$

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
- Ordenación
 - Mergesort
 - Quicksort
- Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Ordenación

- Dado un vector de n números enteros, el problema de ordenación consiste en encontrar una permutación de esos números ordenada de forma creciente.



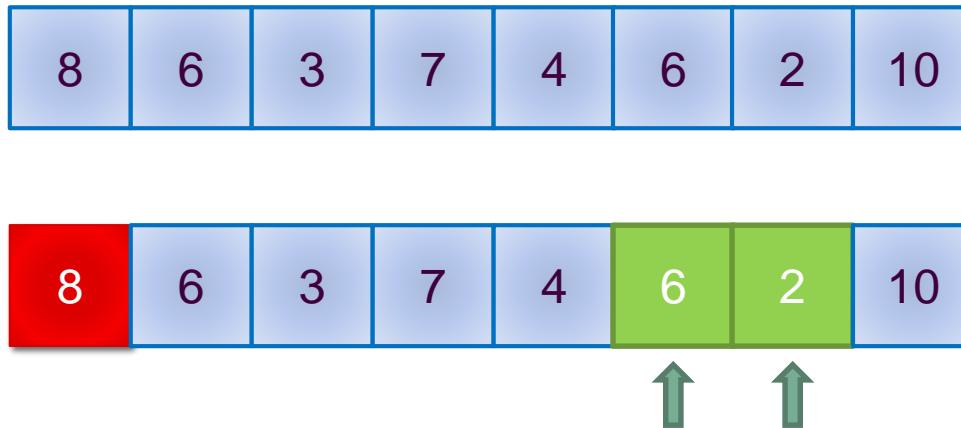
Ordenación

- Método de la burbuja



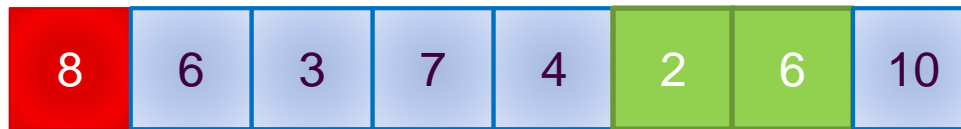
Ordenación

- Método de la burbuja



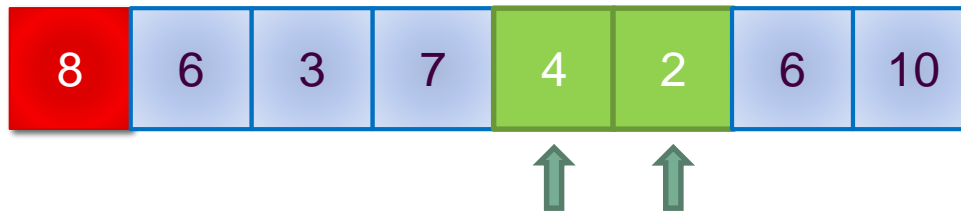
Ordenación

- Método de la burbuja



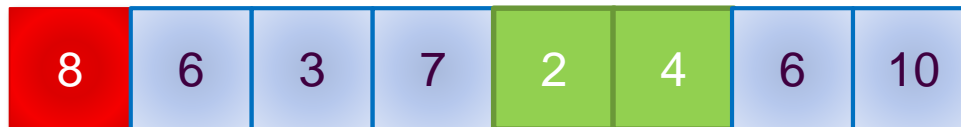
Ordenación

- Método de la burbuja



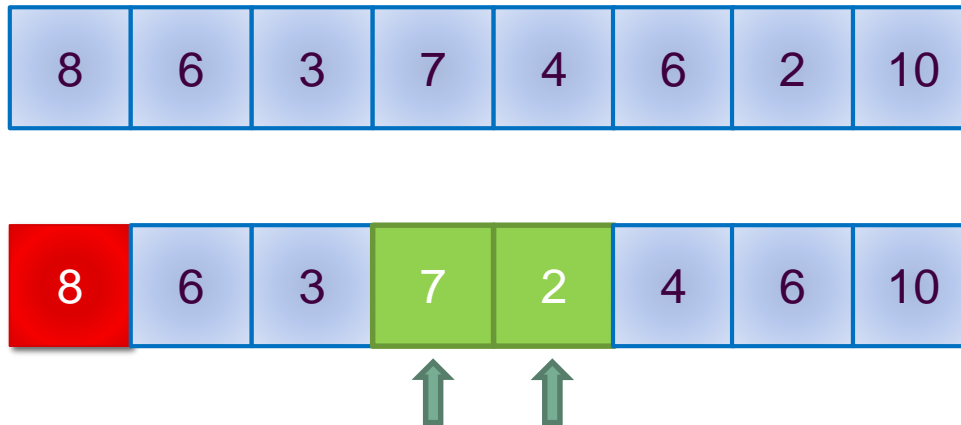
Ordenación

- Método de la burbuja



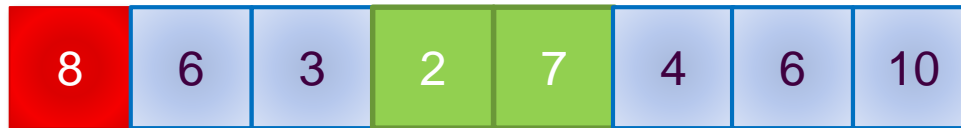
Ordenación

- Método de la burbuja



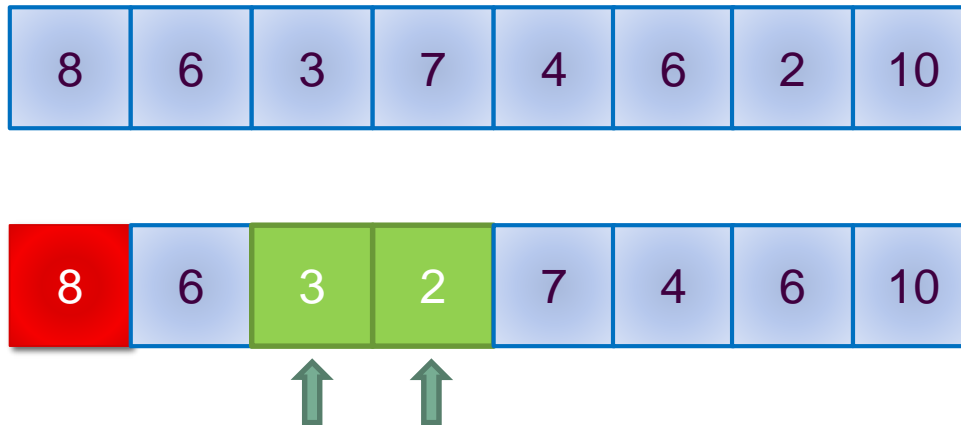
Ordenación

- Método de la burbuja



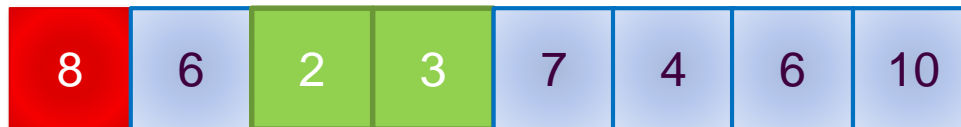
Ordenación

- Método de la burbuja



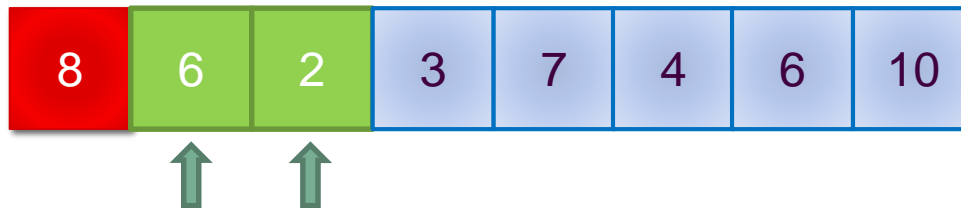
Ordenación

- Método de la burbuja



Ordenación

- Método de la burbuja



Ordenación

- Método de la burbuja



Ordenación

- Método de la burbuja



Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	8	6	3	7	4	6	10
---	---	---	---	---	---	---	----



Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	8	6	3	7	4	6	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	8	6	4	7	6	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	4	8	6	6	7	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	4	6	8	6	7	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	4	6	6	8	7	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	4	6	6	7	8	10
---	---	---	---	---	---	---	----

Ordenación

- Método de la burbuja

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

2	3	4	6	6	7	8	10
---	---	---	---	---	---	---	----

Ordenación

algoritmo burbuja (ENT prim, ult: 1..n; ENT-SAL a:tabla[1..n])

variables

i,j: 1..n; aux: entero

principio

para i=prim **hasta** ult-1 **hacer**

para j=ult **hasta** i+1 **hacer**

si a[j-1] > a[j] **entonces**

aux = a[j-1]

a[j-1] = a[j]

a[j] = aux

fin si

fin para

fin para

fin

$O(n^2)$

Ordenación

- Vamos a estudiar dos algoritmos más eficientes, utilizando la técnica de divide y vencerás

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
- Ordenación
 - Mergesort
 - Quicksort
- Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Mergesort

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - Dividimos el vector que queremos ordenar en dos vectores del mismo tamaño

8	6	3	7
---	---	---	---

4	6	2	10
---	---	---	----

Mergesort

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

- Paso 2 – Divide y vencerás: resolver cada uno de los subproblemas
 - Ordenamos cada uno de los subvectores por separado

3	6	7	8
---	---	---	---

2	4	6	10
---	---	---	----

Mergesort

8	6	3	7	4	6	2	10
---	---	---	---	---	---	---	----

- Paso 3 – Divide y vencerás: combinar las soluciones de los subproblemas
 - Unimos los dos vectores en el orden adecuado

3	6	7	8
---	---	---	---

2	4	6	10
---	---	---	----

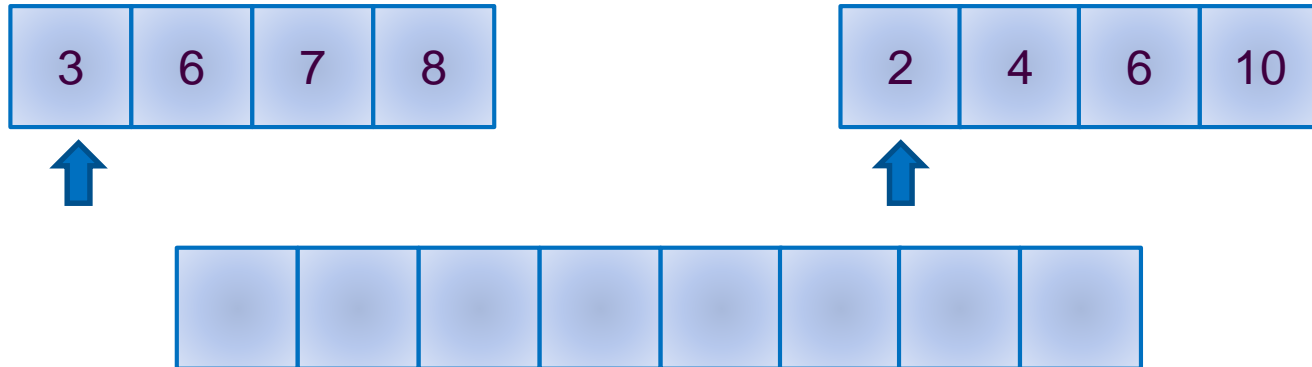
2	3	4	6	6	7	8	10
---	---	---	---	---	---	---	----

Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado

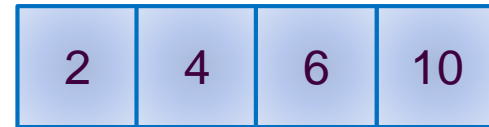
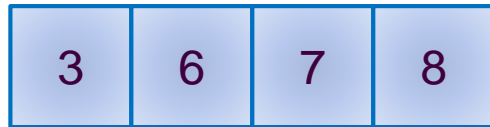
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



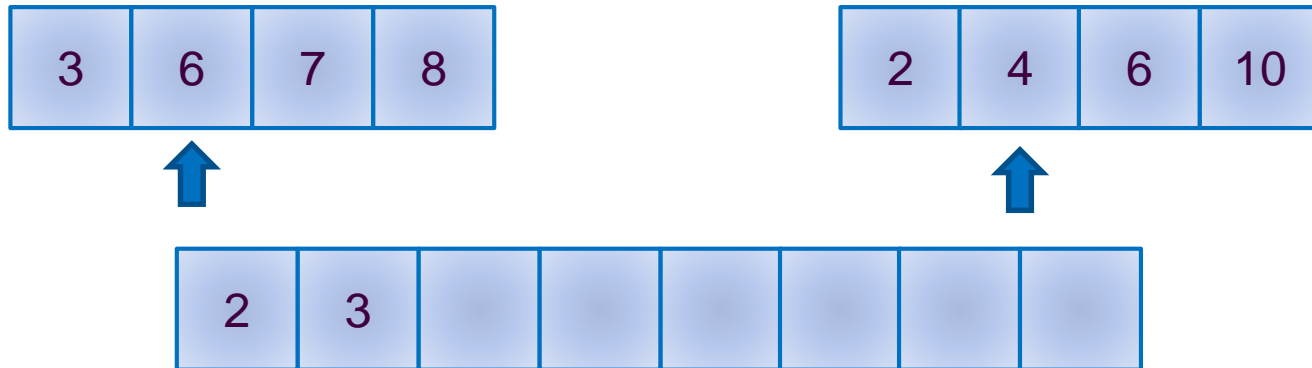
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



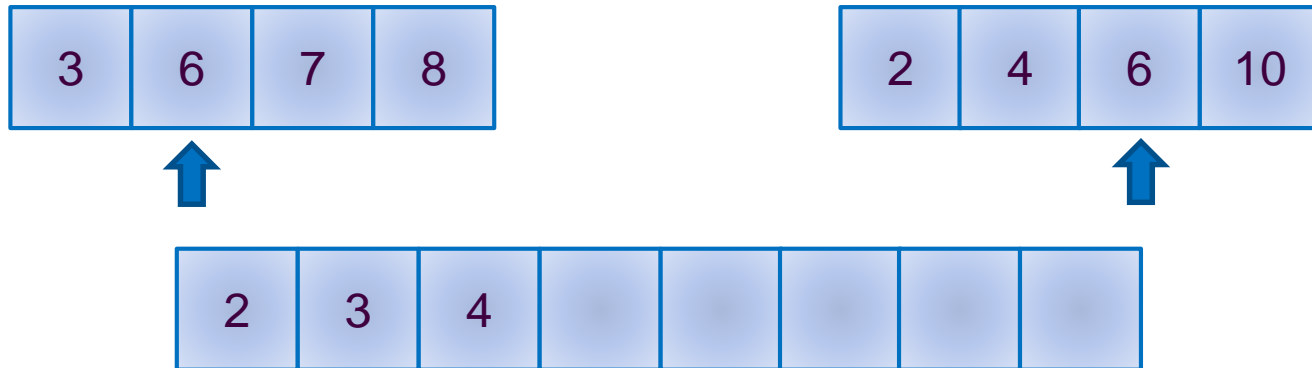
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



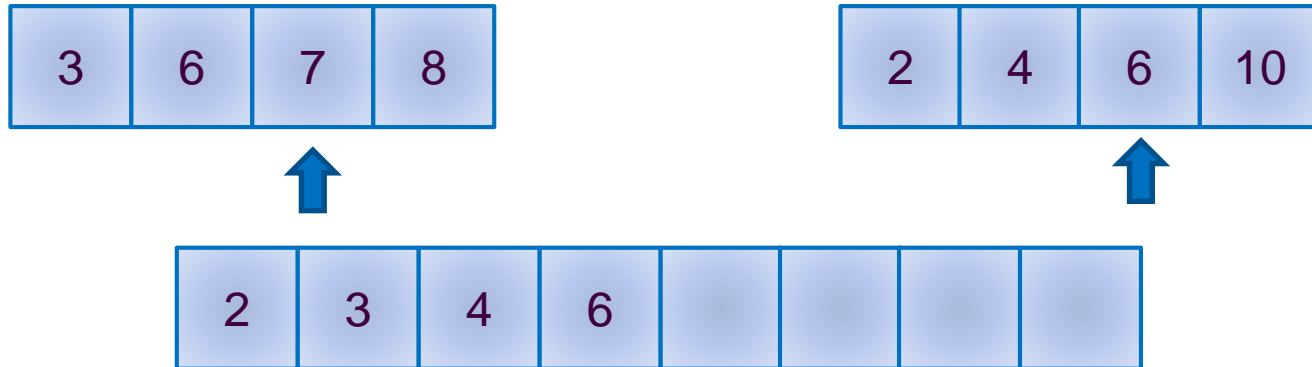
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



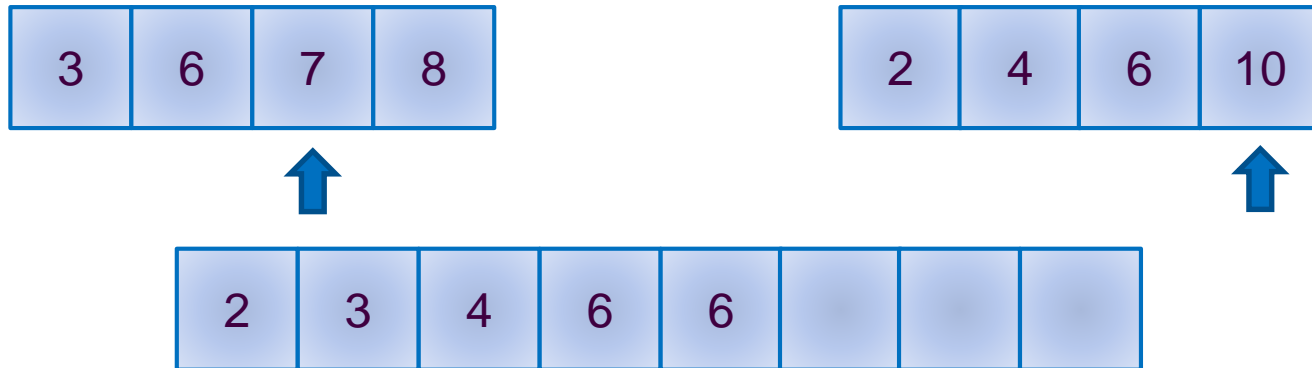
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



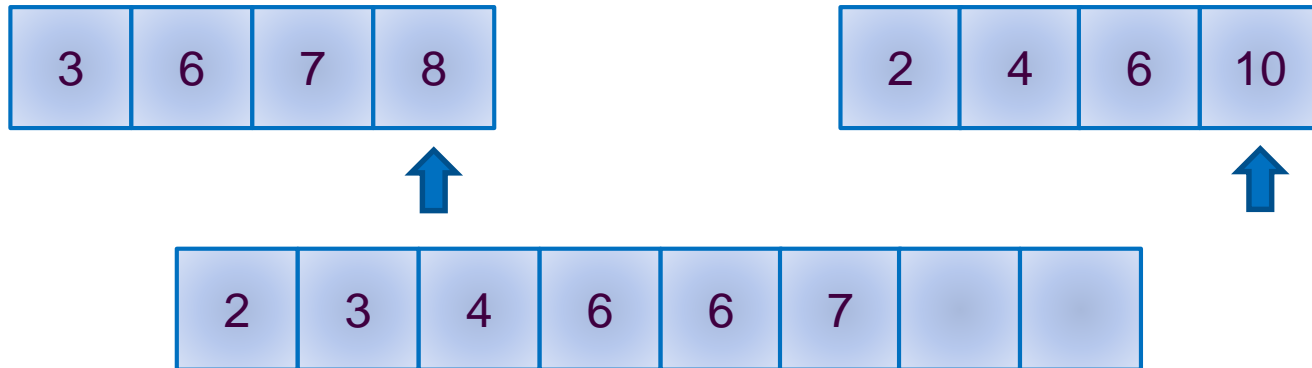
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



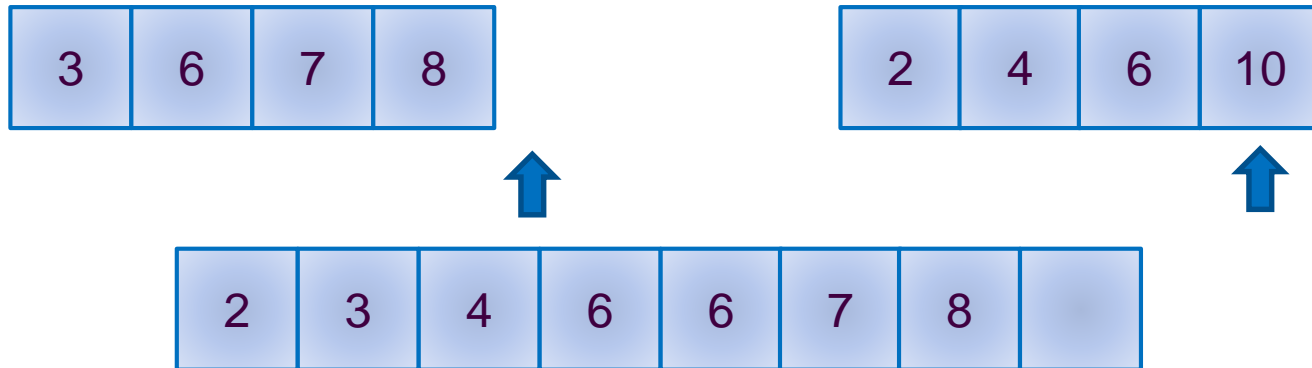
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



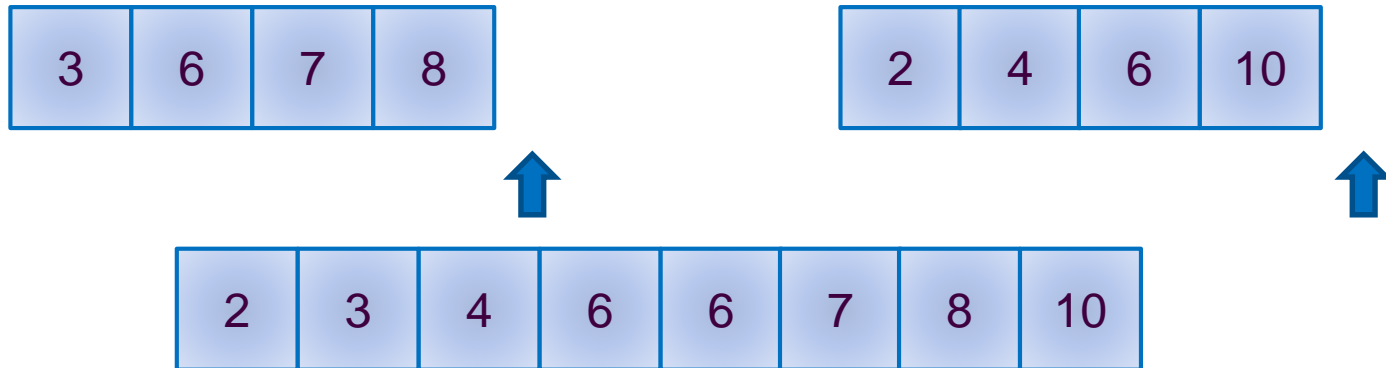
Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado



Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado
- Mientras queden elementos sin estudiar en los dos vectores
 - seleccionar el menor
 - copiarlo en la tabla auxiliar
 - avanzar su índice

Mergesort

- ¿Cómo combinamos los dos vectores?
 - Nos aprovechamos de que cada uno de ellos está ordenado
- Mientras queden elementos sin estudiar en los dos vectores
 - seleccionar el menor
 - copiarlo en la tabla auxiliar
 - avanzar su índice
- Cuando solo queden elementos en vector
 - copiarlos en orden al final de la tabla auxiliar

Mergesort

- Implementación
 - Cuidado con la tabla auxiliar!!
 - ¿Por qué empieza el índice en inf?

Mergesort

- Implementación
 - Cuidado con la tabla auxiliar!!
 - ¿Por qué empieza el índice en inf?
- Reescribir el pseudocódigo para que la tabla auxiliar no ocupe más espacio del necesario

Mergesort

- Complejidad

- La función mezclar tiene una complejidad $O(n)$, por lo que la ecuación del tiempo total del algoritmo es

- $$T(n) = \begin{cases} a & \text{si } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{si } n > 1 \end{cases}$$

- $O(n \log_2 n)$
- La memoria utilizada es mucho mayor que la que estamos acostumbrados, debido a la utilización recursiva de vectores auxiliares

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
- Ordenación
 - Mergesort
 - Quicksort
- Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Quicksort

- En lugar de dividir el problema (vector) en subproblemas más pequeños directamente, preparamos estos subproblemas con antelación, para evitarnos la fase de combinación posterior
- Divide y vencerás
 - Paso 1: preparar los datos
 - Paso 2: dividir en subproblemas
 - Paso 3: ejecutar los subproblemas individualmente

Quicksort

- Preparar los datos
 - Reordenar algunos datos, de tal forma que todos los elementos del primer subvector sean menores que todos los elementos del segundo subvector
 - A priori es muy complicado hacer que esto se cumpla para dos subvectores de igual tamaño

Quicksort

- Preparar los datos
 - Reordenar algunos datos, de tal forma que todos los elementos del primer subvector sean menores que todos los elementos del segundo subvector
 - A priori es muy complicado hacer que esto se cumpla para dos subvectores de igual tamaño
 - Utilizamos la elección de un elemento pivote, que va a determinar el corte del vector



Quicksort

- Preparar los datos

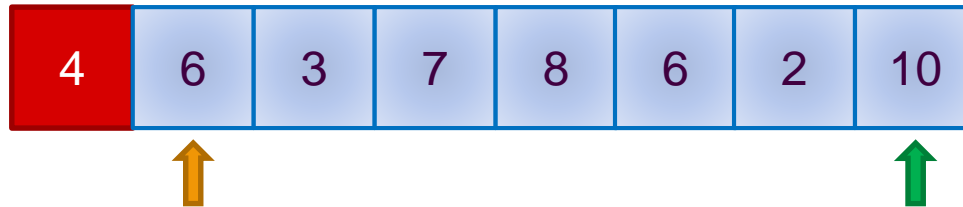
4	6	3	7	8	6	2	10
---	---	---	---	---	---	---	----

3	2	4	7	8	6	6	10
---	---	---	---	---	---	---	----

- Utilizamos una función pivote que, partiendo del valor del pivote, permuta los elementos del vector de tal forma que al finalizar la función, todos los elementos menores o iguales que el pivote están a su izquierda y los elementos mayores que él a su derecha. Además, devuelve la posición en la que ha quedado situado el pivote

Quicksort

- ¿Cómo implementamos la función pivote?
 - Tomamos como valor del pivote el primer elemento

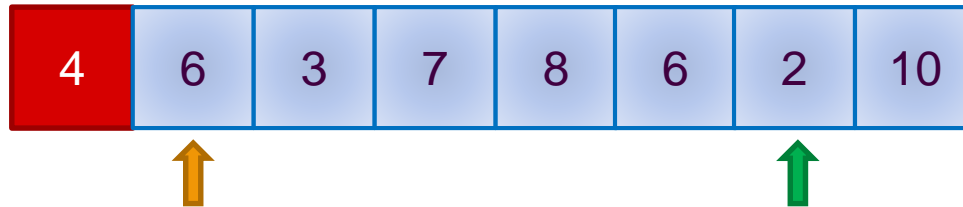


↑ Busca elementos mayores que el pivote

↑ Busca elementos menores que el pivote

Quicksort

- ¿Cómo implementamos la función pivote?
 - Tomamos como valor del pivote el primer elemento

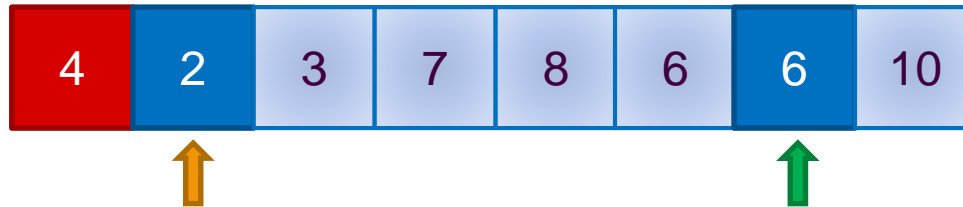


↑ Busca elementos mayores que el pivote

↑ Busca elementos menores que el pivote

Quicksort

- ¿Cómo implementamos la función pivote?
 - Tomamos como valor del pivote el primer elemento

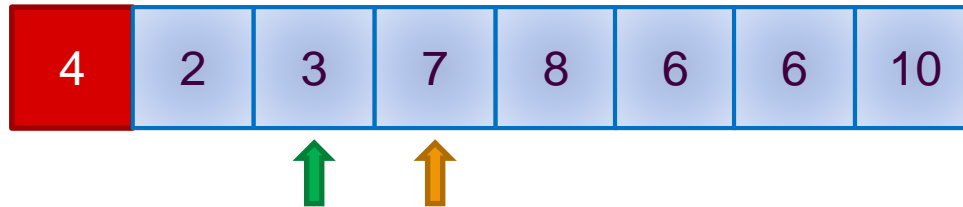


↑ Busca elementos mayores que el pivote

↑ Busca elementos menores que el pivote

Quicksort

- ¿Cómo implementamos la función pivote?
 - Tomamos como valor del pivote el primer elemento



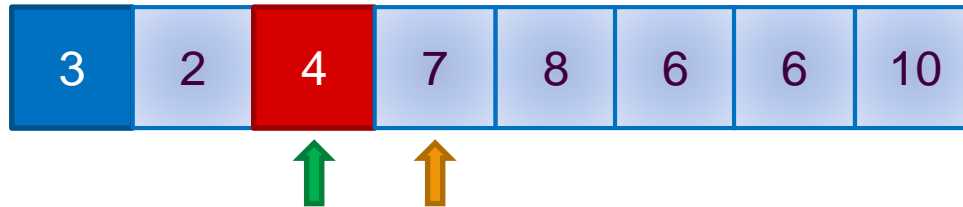
↑ Busca elementos mayores que el pivote

↑ Busca elementos menores que el pivote

Cuando las flechas se cruzan, ya están todos los elementos (excepto el pivote) ordenados en dos grupos

Quicksort

- ¿Cómo implementamos la función pivote?
 - Tomamos como valor del pivote el primer elemento



↑ Busca elementos mayores que el pivote

↑ Busca elementos menores que el pivote

Cuando las flechas se cruzan, ya están todos los elementos (excepto el pivote) ordenados en dos grupos

Falta cambiar el último de los elementos menores (apuntado por ↑) por el pivote

Quicksort

- Complejidad
 - En este algoritmo los casos mejor, peor y medio dependen de la elección del pivote
 - Caso mejor
 - El pivote divide el vector en dos subvectores del mismo tamaño
 - $O(n \log n)$
 - Caso peor
 - El pivote queda colocado en el primer o último elemento
 - $O(n^2)$
- Pivote ideal: mediana de los elementos (complejidad extra para calcularlo)
- Posibilidad: seleccionar tres elementos y coger el intermedio

Ejemplos

- Ordena el siguiente vector de enteros, utilizando los algoritmos mergesort y quicksort

6	3	8	5	1	11	8	9	2	4	7
---	---	---	---	---	----	---	---	---	---	---

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

Calendario deportivo

- Se quiere elaborar un calendario deportivo para una serie de competiciones donde cada participante debe competir exactamente una vez con cada uno de los demás
- Suponemos que el número de participantes, n , es potencia de 2. A cada participante le asignamos un número entre 1 y n
- Representamos la solución en una tabla de $n(n - 1)$ elementos, donde el elemento (i, j) contiene el número del participante con el que tiene que competir i el día j

1							
2							
3							
4			1				
5							
6							
7							
8							
	1	2	3	4	5	6	7
	días						

El día 3, el participante 4 juega
Contra el participante 1

Calendario deportivo

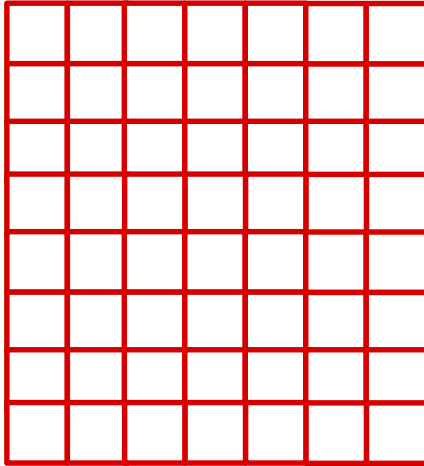
- Posible solución
 - Tiene que cumplir las siguientes restricciones
 - En cada fila no se puede repetir ningún número (cada participante juega una sola vez contra cada contrincante) y deben aparecer todos los números menos el de la fila (juega contra todos los demás)
 - En cada columna no se puede repetir ningún número (en una mismo día, cada participante sólo puede jugar contra un contrincante)

Calendario deportivo

- Posible solución
 - Tiene que cumplir las siguientes restricciones
 - En cada fila no se puede repetir ningún número (cada participante juega una sola vez contra cada contrincante) y deben aparecer todos los números menos el de la fila (juega contra todos los demás)
 - En cada columna no se puede repetir ningún número (en un mismo día, cada participante sólo puede jugar contra un contrincante)bar todas
 - Probar todas las posibles combinaciones de números en la tabla, y seleccionar aquellas que cumplan las restricciones
 - Complejidad $O(n!)$

Calendario deportivo

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cómo podemos dividir el problema?
 - ¿Pueden ser todos los subproblemas del mismo tamaño?
 - ¿Y de la misma naturaleza?

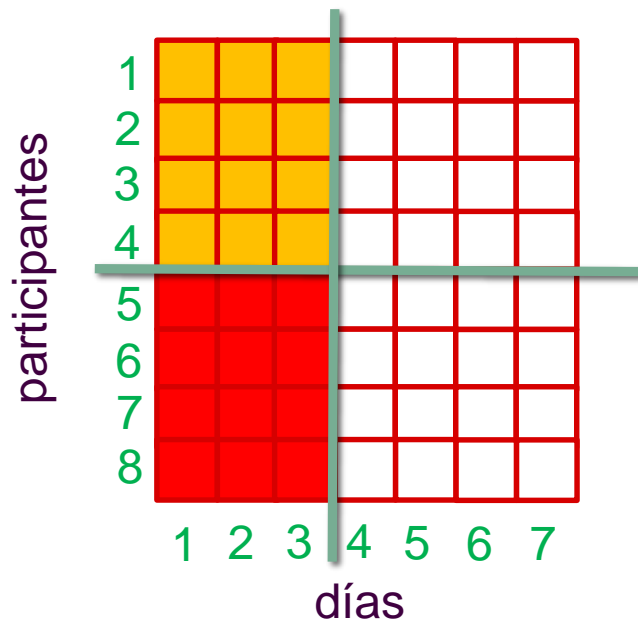


A diagram showing a grid representing a sports calendar. The vertical axis is labeled 'participantes' (participants) and ranges from 1 to 8. The horizontal axis is labeled 'días' (days) and ranges from 1 to 7. The grid consists of 8 rows and 7 columns, totaling 56 cells, representing the possible combinations of participants and days.


	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							
8							

Calendario deportivo

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cómo podemos dividir el problema?
 - ¿Pueden ser todos los subproblemas del mismo tamaño?
 - ¿Y de la misma naturaleza?



El subproblema  consiste en crear un calendario deportivo entre los equipos 1, 2, 3 y 4

El subproblema  consiste en crear un calendario deportivo entre los equipos 5, 6, 7 y 8

Calendario deportivo

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cuál es el caso base?

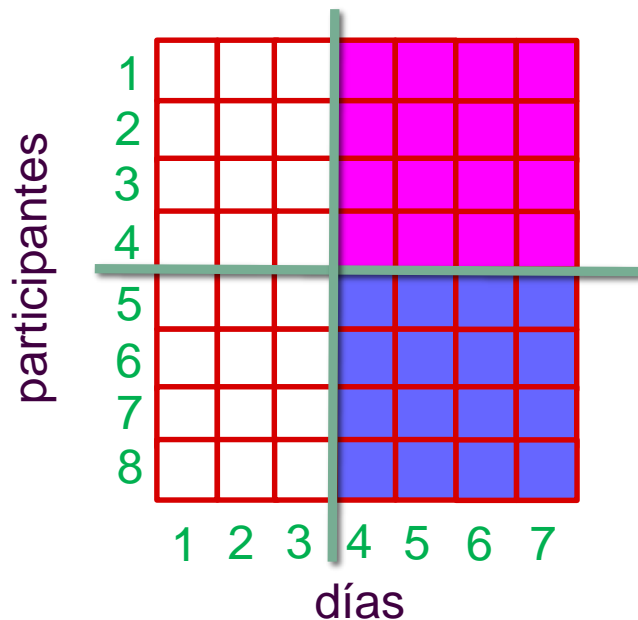
Calendario deportivo



- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cuál es el caso base?
 - Cuando sólo hay dos equipos compitiendo, juegan un único día, el uno contra el otro

participantes	1	2
	2	1
	1	
	días	

Calendario deportivo

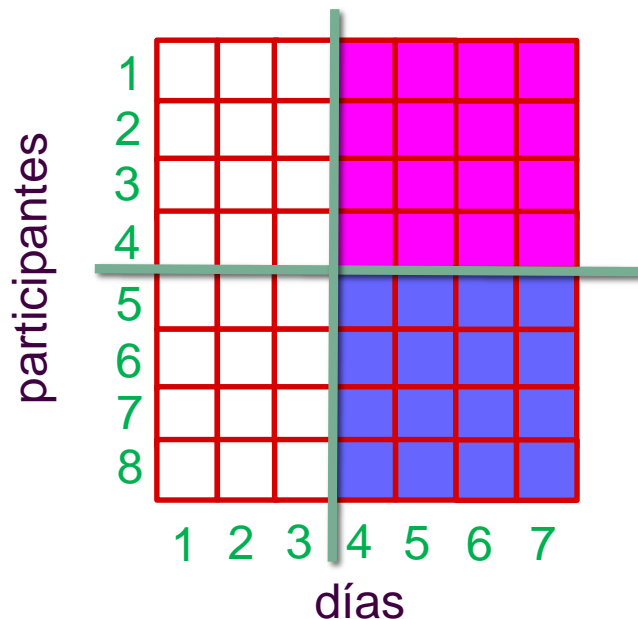
- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cómo podemos dividir el problema?
 - ¿Pueden ser todos los subproblemas del mismo tamaño?
 - ¿Y de la misma naturaleza?





¿Los subproblemas  y  se pueden resolver mediante el algoritmo que cree un calendario deportivo?

Calendario deportivo

- Paso 1 - Divide y vencerás: dividir el problema en subproblemas más pequeños de la misma naturaleza
 - ¿Cómo podemos dividir el problema?
 - ¿Pueden ser todos los subproblemas del mismo tamaño?
 - ¿Y de la misma naturaleza?

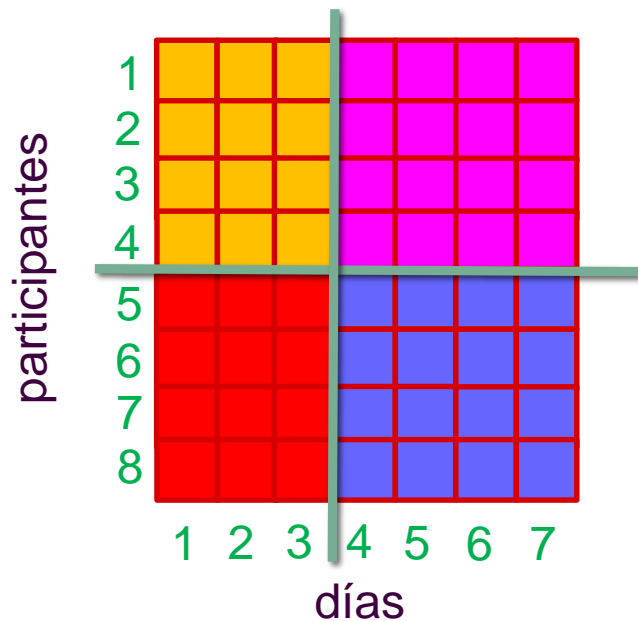


¿Los subproblemas  y  se pueden resolver mediante el algoritmo que cree un calendario deportivo?

Necesitamos otra función, para crear los enfrentamientos entre una mitad de la tabla (participantes 1, 2, 3 y 4) y la otra mitad (participantes 5, 6, 7 y 8)

Calendario deportivo

- Paso 2 – Divide y vencerás: resolver cada uno de los subproblemas



- Los problemas ■ y ■ se resuelven llamando a la función recursiva calendario
- Para los problemas ■ y ■ necesitamos crear otra función diferente (mezclados)

Calendario deportivo

- Paso 3 – Divide y vencerás: combinar las soluciones de los subproblemas
 - Como en cada subproblema ya hemos ido llenando una parte de la tabla solución, no necesitamos hacer nada en este paso

Calendario deportivo

- **función** calendario (ENT inf, sup: 1..n; ENT - SAL t: tabla [1..n][1..n-1])
- **variables**
 - medio : 1..n
- **principio**
 - si $\text{inf} = \text{sup} - 1$ **entonces**
 - $t[\text{inf}, 1] = \text{sup}$
 - $t[\text{sup}, 1] = \text{inf}$
 - **si no**
 - $\text{medio} = (\text{inf} + \text{sup}) \text{ div } 2$
 - calendario (inf, medio, t)
 - calendario (medio + 1, sup, t)
 - mezclados (inf, medio, medio – inf + 1, sup - inf, medio + 1, t)
 - mezclados (medio + 1, sup, medio – inf + 1, sup - inf, inf, t)
 - **fin si**
- **fin**

Calendario deportivo

- Función mezclados
 - El primer participante de la primera mitad de la tabla va a competir en orden contra el primer participante de la segunda mitad, luego contra el segundo, contra el tercero, ...

participantes	1	5	6	7	8
	2				
	3				
	4				
		4	5	6	7
		días			

participantes	5	1			
	6		1		
	7			1	
	8				1
		4	5	6	7
		días			

Calendario deportivo

- Función mezclados
 - El segundo participante de la primera mitad de la tabla va a competir contra los mismos participantes, en el mismo orden, a partir del segundo día

participantes	1	5	6	7	8
	2	8	5	6	7
	3				
	4				
		4	5	6	7
		días			

participantes	5	1	2		
	6		1	2	
	7			1	
	8	2			1
		4	5	6	7
		días			

Calendario deportivo

- Función mezclados
 - El tercer participante de la primera mitad de la tabla va a competir contra los mismos participantes, en el mismo orden, a partir del tercer día

participantes	1	5	6	7	8
	2	8	5	6	7
	3	7	8	5	6
	4				
		4	5	6	7
		días			

participantes	5	1	2	3	
	6		1	2	3
	7	3		1	2
	8	2	3		1
		4	5	6	7
		días			

Calendario deportivo

- Función mezclados
 - El cuarto participante de la primera mitad de la tabla va a competir contra los mismos participantes, en el mismo orden, a partir del cuarto día

participantes	1	5	6	7	8
	2	8	5	6	7
	3	7	8	5	6
	4	6	7	8	5
		4	5	6	7
		días			

participantes	5	1	2	3	4
	6	4	1	2	3
	7	3	4	1	2
	8	2	3	4	1
		4	5	6	7
		días			

Calendario deportivo

- **función** mezclados(ENT equipointf, equiposup, diainf, diasup, equipoini: 1..n; ENT-SAL t: tabla [1..n][1..n-1])
- **variables**
 - $l, j : 1..n$
- **principio**
 - **para** $j = \text{diainf}$ **a** diasup **hacer**
 - $t[\text{equipointf}, j] = \text{equipoini} + j - \text{diainf}$
 - **fin para**
 - **para** $i = \text{equipointf} + 1$ **a** equiposup **hacer**
 - $t[i, \text{diainf}] = t[i-1, \text{diasup}]$
 - **para** $j = \text{diainf} + 1$ **a** diasup **hacer**
 - $t[i, j] = t[i-1, j-1]$
 - **fin para**
 - **fin para**
- **fin**

Calendario deportivo

- Debido a la función mezclados:
 - $O(n^2)$

Índice

- Técnica Divide y vencerás
- Problemas
 - Multiplicación de grandes enteros
 - Búsqueda binaria
 - Mediana de dos vectores ordenados
 - Buscar el elemento en su posición
 - Matriz cuadrada transpuesta
 - Subsecuencia de suma máxima
 - Ordenación
 - Mergesort
 - Quicksort
 - Otros problemas
 - Calendario deportivo
 - Multiplicación de matrices

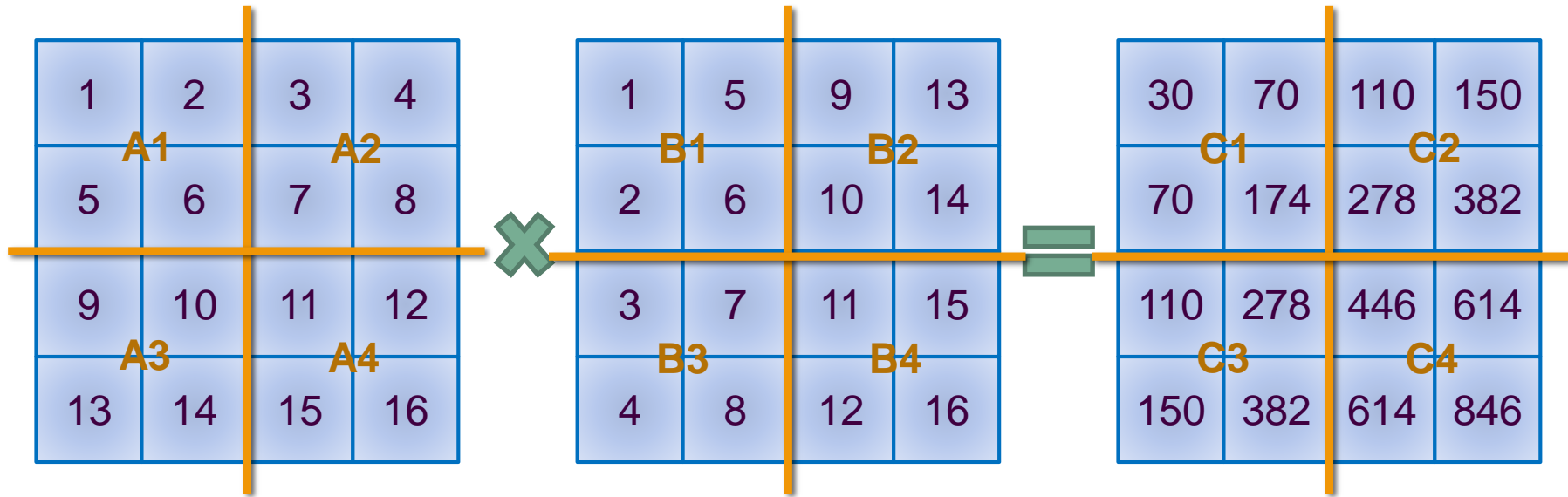
Multiplicación de matrices

- Dadas A y B dos matrices de tamaño $n \times n$, calcular su producto C

1	2	3	4		1	5	9	13		30	70	110	150
5	6	7	8		2	6	10	14		70	174	278	382
9	10	11	12		3	7	11	15		110	278	446	614
13	14	15	16		4	8	12	16		150	382	614	846

- Método habitual
 - $C_{ij} = \sum_{k=1}^n A_{ik}B_{kj}, \quad i, j = 1 \dots n$
 - $\Theta(n^3)$

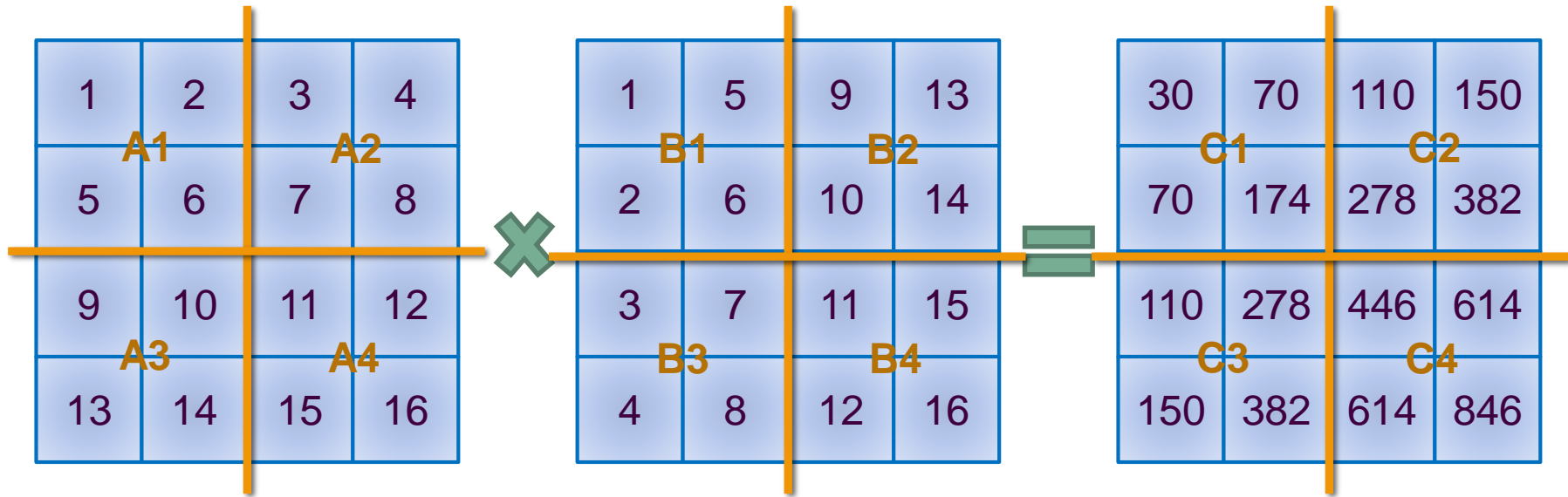
Multiplicación de matrices



- $C1 = A1 \times B1 + A2 \times B3$
- $C2 = A1 \times B2 + A2 \times B4$
- $C3 = A3 \times B1 + A4 \times B3$
- $C4 = A3 \times B2 + A4 \times B4$

→ Multiplicación de matrices recursiva

Multiplicación de matrices



- $C1 = A1 \times B1 + A2 \times B3$
- $C2 = A1 \times B2 + A2 \times B4$
- $C3 = A3 \times B1 + A4 \times B3$
- $C4 = A3 \times B2 + A4 \times B4$

- $T(n) = 8T\left(\frac{n}{2}\right) + 4n^2 + c \rightarrow \Theta(n^3)$

Multiplicación de matrices

- Reescribimos los cálculos de C1, C2, C3, C4 utilizando sólo 7 multiplicaciones de matrices
 - $AUX1 = (A3+A4-A1) \times (B4-B2+B1)$
 - $AUX2 = A1 \times B1$
 - $AUX3 = A2 \times B3$
 - $AUX4 = (A1-A3) \times (B4-B2)$
 - $AUX5 = (A3+A4) \times (B2-B1)$
 - $AUX6 = (A2-A3+A1-A4) \times B4$
 - $AUX7 = A4 \times (B1+B4-B2-B3)$
- $C1 = AUX2 + AUX3$
- $C2 = AUX1 + AUX2 + AUX5 + AUX6$
- $C3 = AUX1 + AUX2 + AUX4 - AUX7$
- $C4 = AUX1 + AUX2 + AUX4 + AUX5$
- $T(n) = 7T\left(\frac{n}{2}\right) + 24n^2 + c \rightarrow 7T\left(\frac{n}{2}\right) + 15n^2 + c \rightarrow \Theta(n^{2,81})$