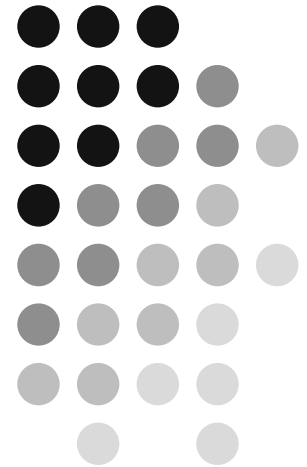


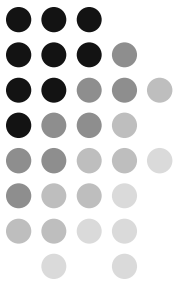
Divide-y-vencerás, backtracking y programación dinámica



UNIVERSIDAD
DE LOS ANDES

Diseño y Análisis de Algoritmos
Cátedra de Programación
Carrera de Ingeniería de Sistemas
Prof. Isabel Besembel Carrera

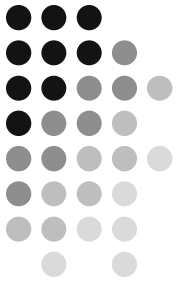




- ✓ Utilización de la técnica de divide-y-vencerás
 - ❖ Dividir la instancia del problema en dos o más instancias más pequeñas del **mismo** problema,
 - ❖ Resolver cada una de estas instancias recursivamente y
 - ❖ Ensamblar sus soluciones para obtener la solución de la instancia original

La recursión se detiene cuando la instancia alcanzada sea tan pequeña que no puede ser dividida, arrojando una solución directa.

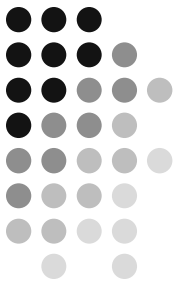
La base para probar la corrección del algoritmo: por medio de la inducción sobre el tamaño de la instancia



- ❖ División del problema
 - Utilizar la primera forma que se encuentre o
 - Estudiar cuidadosamente la mejor forma de dividir, para que el proceso de ensamblaje se simplifique

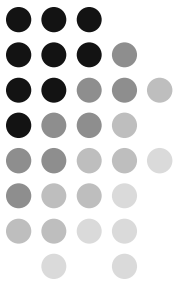
Junio,04		
divide-y-venceras(Tipo: instancia):tipoDeResultado		
{ pre: precondiciones }		{ pos: poscondiciones }
1	Si (instancia es suficientemente pequeña o sencilla) entonces algoritmoAdHoc(instancia) fsi	Documentación
2	división de la instancia en subinstancias más pequeñas i_1, i_2, \dots, i_n	
3	$[y(j)=divide-y-venceras(i_j)]$ resultado = ensamblaje de los y(j)] j = 1, n	
4	regrese resultado	

Ejemplo estrategia 3



- ✓ Multiplicar 2 enteros de n cifras: Algoritmo clásico tiene $O(n^2)$
 - ❖ Multiplicación a la rusa tiene el mismo rendimiento
 - ❖ Multiplicación con divide-y-vencerás: tiene 4 multiplicaciones de enteros de $n/2$ cifras, sin presentar mejora en el tiempo de ejecución
 - Mejora: reducir a 3 multiplicaciones de enteros de $n/2$ cifras
 - Igualar el número de cifras de ambos operandos: 0981 y 1234
 - Dividir ambas cifras por la mitad: $w=09$, $x=81$, $y=12$, $z=34$
 $981 = 10^2w + x$ $1234 = 10^2y + z$

Ejemplo estrategia 3



$$981 \times 1234 = (10^2w + x)(10^2y + z) = 10^4wy + 10^2(wz + xy) + xz$$

- Sólo se necesita calcular la suma de wz , xy en una sola multiplicación ya que

$$r = (w+x)(y+z) = wy + (wz + xy) + xz$$

Esto se traduce en tener sólo 3 multiplicaciones, a saber:

$$p = wy = 09 * 12 = 108$$

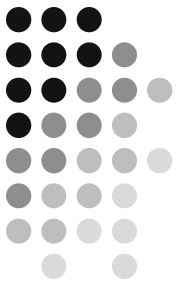
$$q = xz = 81 * 34 = 2754$$

$$r = (w+x)(y+z) = 90 * 46 = 4140$$

Para finalmente hacer

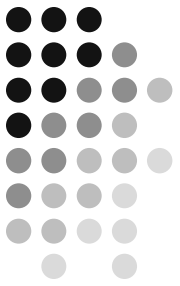
$$981 * 1234 = 10^4 p + 10^2(r - p - q) + q = 1080000 + 127800 + 2754 = 1210554$$

Ejemplo estrategia 3



- Algoritmo clásico: $h(n) = cn^2 + bn + a \approx cn^2$
- Con 3 multiplicaciones: $3 h(n/2) + g(n) = 3c(n/2)^2 + g(n) = \frac{3}{4} cn^2 + g(n) = \frac{3}{4} h(n) + g(n)$
- ❖ $h(n) \in \Theta(n^2)$ y $g(n) \in \Theta(n)$
- ❖ Mejora de aproximadamente 25%, cuando n es suficientemente grande

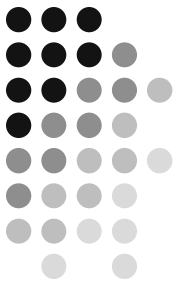
Ejemplo estrategia 3



- ✓ Ordenar ascendentemente un conjunto de números con el método quicksort [Hoare,1962]
- ✓ **Divide:** selección del pivote (un elemento de la secuencia) y acomodo de los elementos, los menores que el pivote a su izquierda y los mayores que el pivote a su derecha
- ✓ **Vencerás:** se aplica el quicksort a la partición izquierda y luego a la partición derecha
- ✓ **Combinar:** No es necesario ya que las particiones están ordenadas



Ejemplo estrategia 3



- ✓ Multiplicar dos matrices con el algoritmo de Strassen

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

Donde: $C_{11} = A_{11} B_{11} + A_{12} B_{21}$

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

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

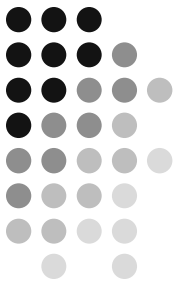
$$C_{22} = A_{21} B_{12} + A_{22} B_{22}$$

Asume a N en potencias de 2 y divide A y B en cuatro submatrices de N/2 x N/2

$$T(n) = 8T(n/2) + \theta(n^2)$$

$$\text{Tiempo total } \theta(n^{\log 8}) = \theta(n^3)$$

Ejemplo estrategia 3



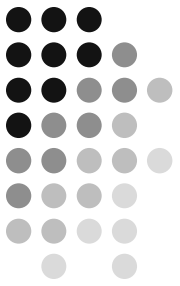
- 1) Divida la matriz de entrada (n en potencias de 2) A y B en cuatro matrices de $n/2 \times n/2$
- 2) Usando tiempo de $\theta(n^2)$ calcule 14 sumas y restas
- 3) Calcule recursivamente 7 multiplicaciones
 $P_i = A_i \cdot B_i$ para $i=1,2,\dots,7$
- 4) Calcule las submatrices c_{ij} para obtener C , por combinaciones de sumas y restas y las P_i multiplicaciones que satisfagan y en tiempo $\theta(n^2)$

Mejora el tiempo en: $T(n) = 7T(n/2) + \theta(n^2)$

$$\text{Tiempo total } \theta(n^{\log 7}) = \theta(n^{2.87})$$

preferible para matrices grandes y esparcidas

Ejemplo estrategia 3



$$m_1 = (A_{12} - A_{12}) (B_{21} + B_{22})$$

$$m_2 = (A_{11} - A_{22}) (B_{11} + B_{22})$$

$$m_3 = (A_{11} - A_{21}) (B_{11} + B_{12})$$

$$m_4 = (A_{11} - A_{21}) B_{22}$$

$$m_5 = (B_{12} - B_{22}) A_{11}$$

$$m_6 = (B_{21} - B_{11}) A_{22}$$

$$m_7 = (A_{21} - A_{22}) B_{11}$$

$$C_{11} = m_1 + m_2 - m_4 + m_6$$

$$C_{12} = m_4 + m_5$$

$$C_{21} = m_6 + m_7$$

$$C_{22} = m_2 - m_3 + m_5 - m_7$$

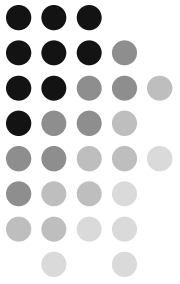
Multiplica 2 matrices
usando 7 multiplicaciones
y 14 sumas o restas

$$T(n) \leq mT(n/2) + (a n^2)/4$$

El más rápido hoy

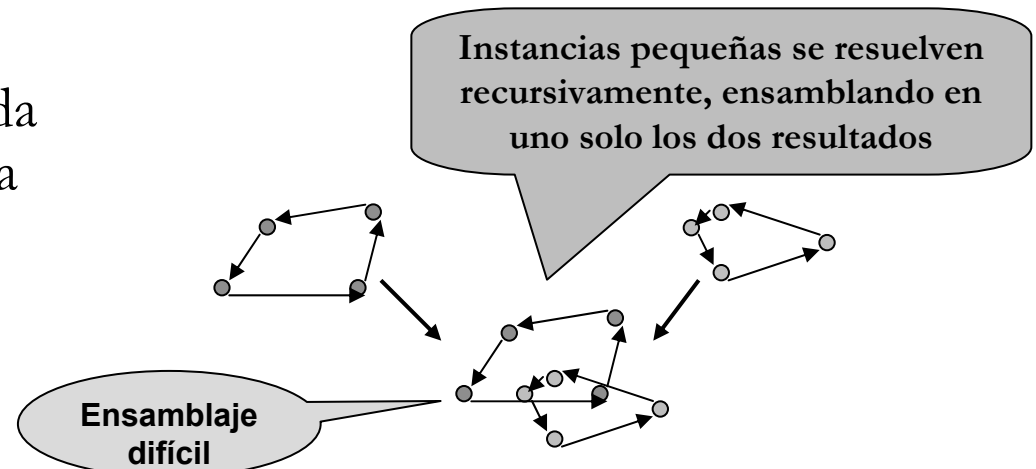
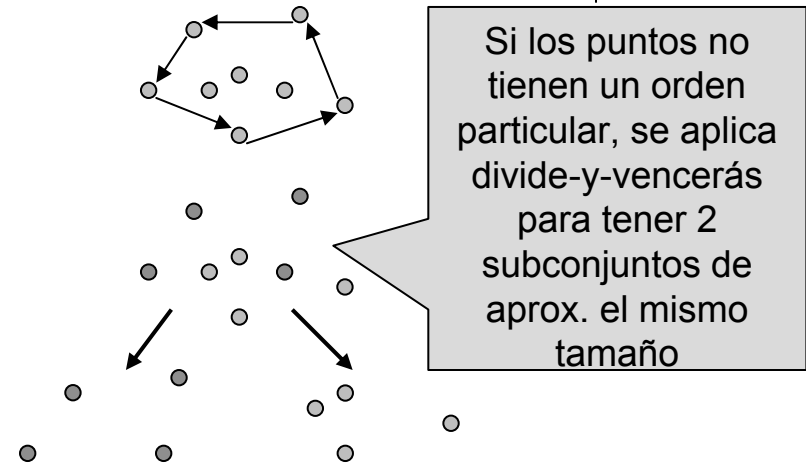
Coppersmith y Winograd,
1968: posibilidad de
realizar la multiplicación
en $O(n^{2,376})$

Ejemplo estrategia 3

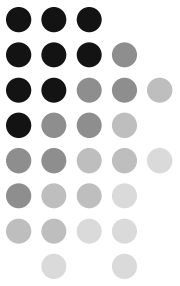


✓ Encontrar el cerco convexo de un conjunto de puntos en el plano XY

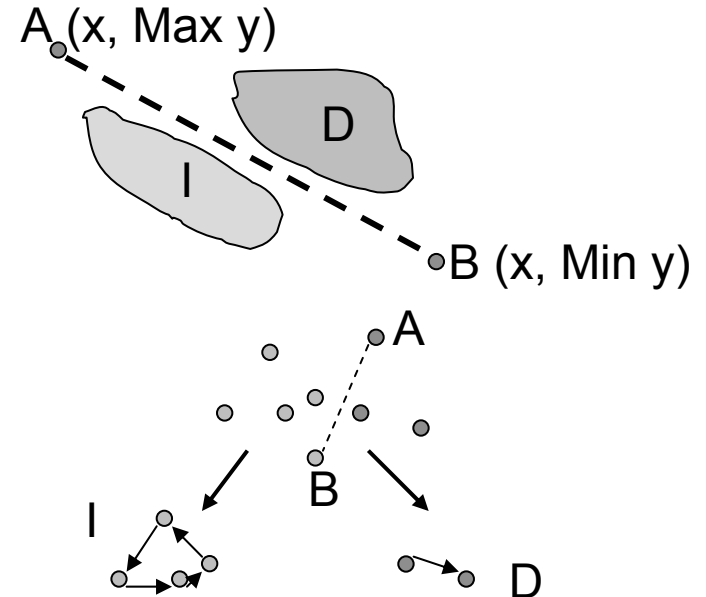
- ❖ Cerco convexo: secuencia de puntos del conjunto que define una figura convexa que los encierra.
- ❖ Figura convexa: es una figura cerrada, donde cada línea intersecta una figura convexa en no más de 2 puntos



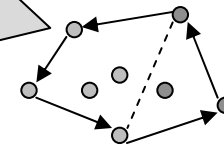
Ejemplo estrategia 3

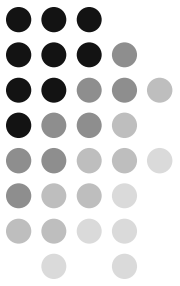


- No siempre es posible ensamblar las pequeñas soluciones
- Escogencia del método de división: Tanto A como B deben estar en el cerco convexo
- Encontrar el máximo y mínimo Y, para decidir que puntos están sobre la línea AB y cuales por debajo en $O(n)$ y luego resolver los subproblemas recursivamente.
- Análisis parecido al quicksort



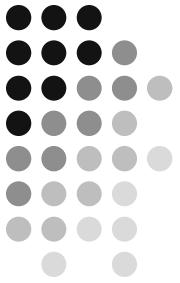
Ensamble de la solución conectando A a D y B a I y D.
 $O(n \lg n)$ en promedio. $O(n^2)$ en el peor de los casos (D o I vacíos)



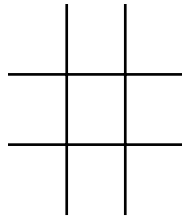


- Backtracking (vuelta atrás): técnica de búsqueda exhaustiva
- ❖ Árboles de juego:
 - Ejemplo: juego “la vieja” o tic-tac-toe
 - Colocar reglas para definir los movimientos y la finalización del juego
 - Seleccionar los nodos hojas con la decisión final del juego (gana A, gana B, empate)
 - Construir el árbol y visitar los nodos en posorden, se visita el nodo rama N, luego todos sus hijos, pero se evalúa N tomando el min o max de los valores de sus nodos hijos, según corresponda.

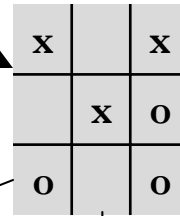
Estrategia 4



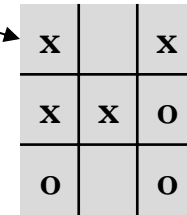
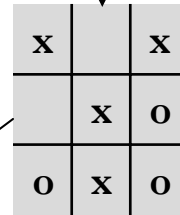
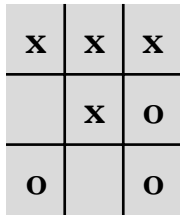
raíz



Juega A



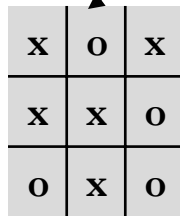
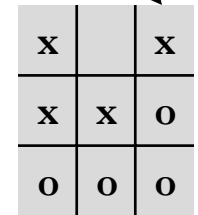
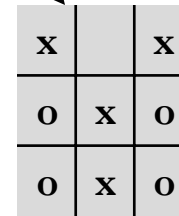
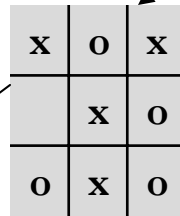
Juega B



Gana B

1

Juega A

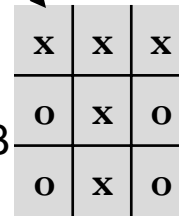


empate

0

Gana B

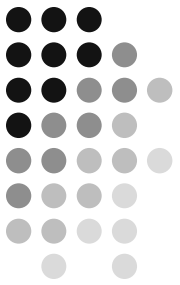
1



Gana A

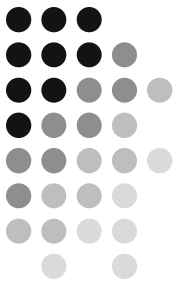
-1

Estrategia 4



- Los algoritmos de vuelta atrás se pueden utilizar aun cuando las soluciones buscadas no tengan la misma longitud

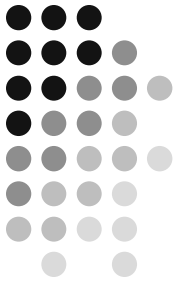
enero,05		vueltaAtras(Arreglo: v, Entero k)	
{ pre: precondiciones }		{ pos: poscondiciones }	
1	si (v es una solución) entonces Escribir v sino $\forall w$ que sea una solución parcial / $w_i = v_i$ con $i = 1, k$ vueltaAtras(w, k+1) fsi	Documentación	



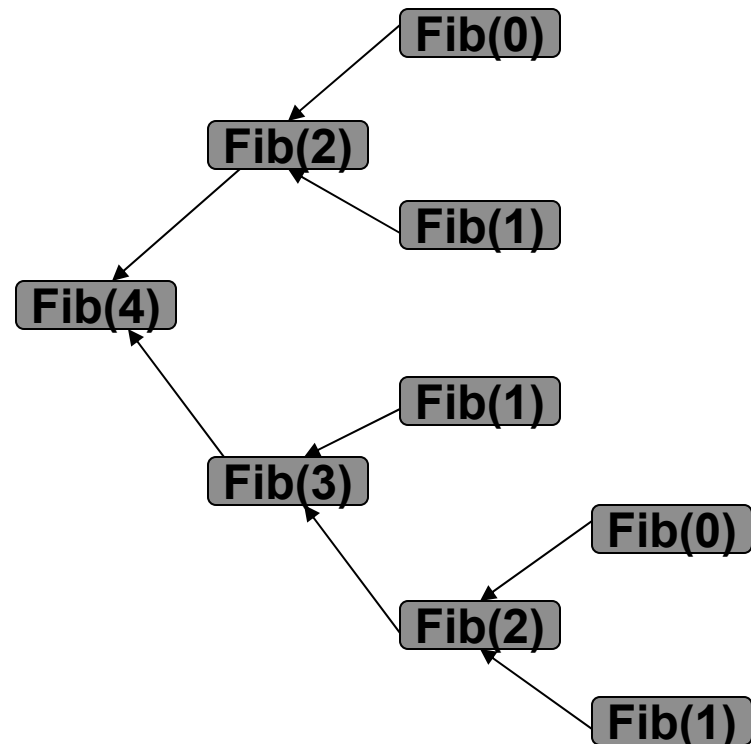
- Programación dinámica
 - ❖ Técnica utilizada para eliminar el trabajo redundante almacenando en un área de memoria especial las soluciones encontradas anteriormente
- Ejemplo:
 - ❖ Encontrar el n-ésimo número de la secuencia de Fibonacci

$$F(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ F(n-1) + F(n-2) & \text{si } n > 1 \end{cases}$$

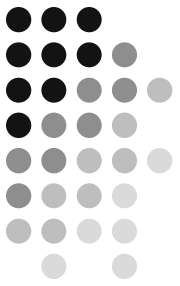
Estrategia 5



- La solución obvia es utilizar la estrategia divide-y-vencerás, pero ella tiene una complejidad en tiempo del tipo exponencial
- El árbol de las llamadas para el caso $n = 4$



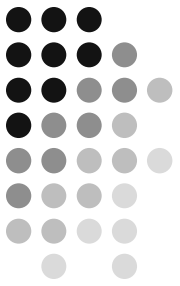
Estrategia 5



- La mejora del algoritmo se realiza con la programación dinámica, obteniéndose una complejidad $O(n)$
- La programación dinámica es una técnica de optimización de código útil, aunque no reduzca la complejidad asintótica

Junio,04		
fibonacci(Entero: n):Entero		
{ pre: $n \geq 0$ }		{ pos: número de Fibonacci ≥ 0 }
1	$v(0), v(1) = 0, 1$	-v: Arreglo(0..n)de Entero. Vector auxiliar que almacena los valores intermedios anteriores -j: Entero. Subíndice de v
2	$[v(j) = v(j - 1) + v(j - 2)] j = 2, n$	
3	regrese $v(n)$	
1	$n=4, v=(0,1,1,2,3), v(4)=3$	Caso exitoso
2	$n=0, v=(0,1), v(0)=0$	Caso exitoso

Ejercicio



- Dadas k papeleras, cada una con capacidad c , y un conjunto de items $t = \{t_1, t_2, \dots, t_n\}$ cada uno con un tamaño asociado. El problema es colocar los n items dentro de las k papeleras sin exceder la capacidad de cualquiera de ellas o en caso contrario reportar falla por ser imposible la solución.