estudios de postgrado
en computación

Análisis y Diseño de Algoritmos (AyDA)

Isabel Besembel Carrera

TÉCNICAS DE DISEÑO
DIVIDE-Y-VENCERÁS
VUELTA-ATRÁS

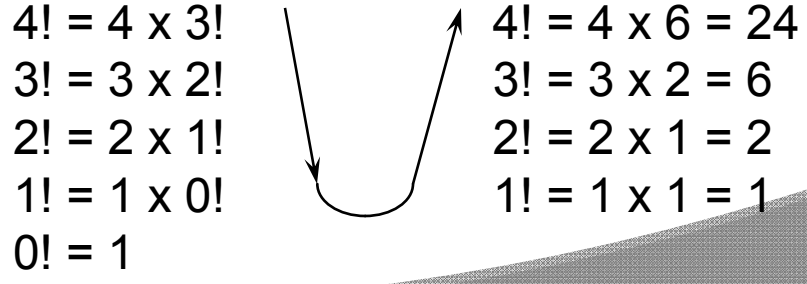
RECURSIÓN

• La recursión es una **técnica fundamental** en el diseño de algoritmos, que está basada en la solución de versiones más pequeñas del problema, para obtener la solución general. Una instancia se soluciona en base a la solución de una o más instancias **diferentes** y **más pequeñas** que ella.

• Todo programa **recursivo** debe tener una condición de finalización.

• Para probar la corrección de los algoritmos recursivos se utiliza la inducción sobre el tamaño de las instancias.

Ejemplo:



RECURSIÓN. Ejemplo.

E1: Calcular n!

pre: $\exists n / n \in \mathbb{N} \wedge n \geq 0$ pos: f = n!

Ene.97. factorial(Entero: n):Entero {pre: n ≥ 0} {pos: Calcula el factorial de n}		
1	regrese (Si (n = 0) entonces 1 sino n * factorial(n - 1) fsi)	**-factorial:** Entero: Contiene el factorial de n.
1 2	n = 3, factorial = 6 n = 0, factorial = 1	Caso exitoso. Caso exitoso.

Prueba por inducción sobre n

Etapa básica: n = 0. El algoritmo regresa 1. (correcto).

Etapa inductiva: La hipótesis inductiva es que factorial(j) regresa j! $\forall$ j en el rango $0 \leq j \leq n-1$. Se debe demostrar que factorial(n) regresa n!. Como $n > 0$, la prueba de n = 0 falla y el algoritmo regresa n * factorial(n-1). Por la hipótesis inductiva, factorial(n-1) regresa (n-1)!, por lo cual factorial(n) regresa n x (n-1)! lo cual es igual a n!.

Estrategia 3

- Utilización de la técnica de divide-y-vencerás
  - Dividir la instancia del problema S(n) en dos o más instancias más pequeñas del **mismo** problema $S(n_1)$, $S(n_2)$, …, $S(n_k)$, donde $n_i < n$, $i = 1, 2, \ldots, k$
  - Resolver cada una de estas instancias (i) recursivamente y
  - Ensamblar sus soluciones $S(n_1)$, $S(n_2)$, …, $S(n_k)$, para obtener la solución de la instancia original S(n)

  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

Estrategia 3

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

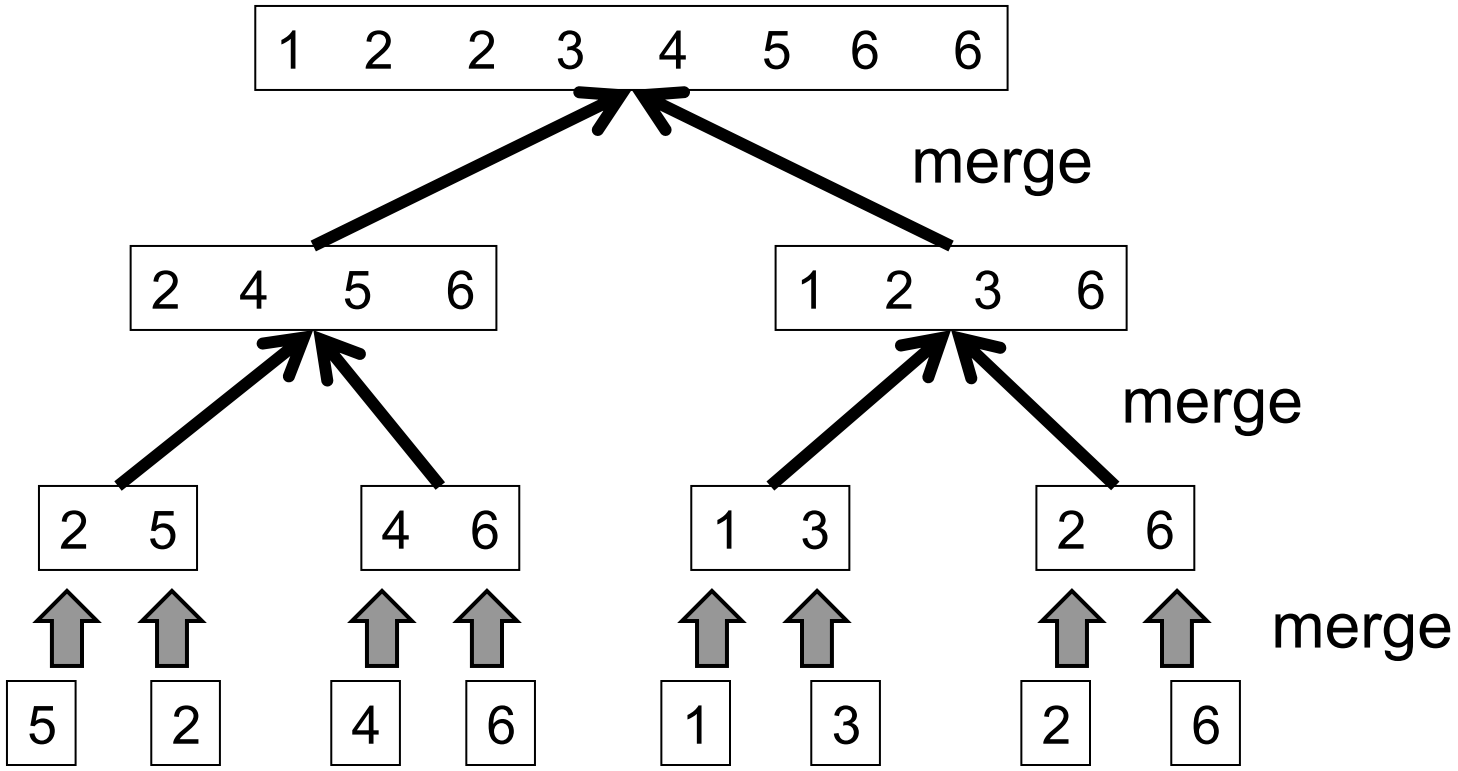
Junio,04	**divide-y-venceras(Tipo: instancia):tipoDeResultado** **{ pre: precondiciones }** **{ pos: poscondiciones }**	
1	**Si (instancia es suficientemente pequeña o sencilla) entonces** **algoritmoAdHoc(instancia)** **fsi**	**Documentación**
2	**división de la instancia en subinstancias más pequeñas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

Estrategia 3

Ordenamiento y mezcla (merge-sort)

- Análisis: b > 0 es una constante y n es potencia de 2
  - Si n < 2, la solución $\Theta(1)$
  - $T(n) = 2T(n/2) + b\,n$, si $n \geq 2$ $\quad\quad$ $T(n) = O(n \lg n)$

Sep,09	**mergeSort(ArregloDe TipoEle: A, Natural: p, Natural: r)**	
{ pre: |A| > 0}		**{ pos: Ai ≤ Aj, i < j }**
1	**Si (p < r) entonces** **q = ⌊(p+r)/2⌋** **mergeSort(A, p, q)** **mergeSort(A, q+1, r)** **merge(A, p, q, r)** **fsi**	**A:** Arreglo[n]De TipoEle. Vector de elementos a ser ordenado **p, r:** Natural. Límite inferior y superior del rango de elementos en el vector a ordenar. **q:** Natural. Subíndice auxiliar.



Secuencia inicial

Ejemplo 1. Estrategia 3

✓Multiplicación de 2 enteros de n cifras:

- Enteros representados en una secuencia grande de bits que no puede ser directamente manipulada en la UAL de un CPU
- Usado en criptografía, RSA
- Dados 2 enteros I y J representados con n bits
  - I+J y I-J se realizan en O(n)
  - I * J en $O(n^2)$
- Asuma que n es potencia de 2 (sino se rellena con 0s)
  - I se pica en I_l y I_h, J en J_l y J_h, $I = I_h 2^{n/2} + I_l$, $J = J_h 2^{n/2} + J_l$

Ejemplo 1. 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
- $981 = 10^2 w + x$ $1234 = 10^2 y + z$

Ejemplo 1. 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 1. Estrategia 3

- Algoritmo clásico: $T(n) = 4T(n/2)+cn$
  $c > 0$, por el teorema maestro $\Theta(n^2)$
- Con 3 multiplicaciones:
  Para $n \geq 2$, $T(n) = 3\,T(n/2) + cn$, para $c > 0$
  $= \Theta(n^{\lg 3}) = \Theta(n^{1.585})$
  - Mejora de aproximadamente 25%, cuando n es suficientemente grande
- Una mejora adicional se logra con el algoritmo denominado transformada rápida de Fourier en $O(n \lg n)$

Ejemplo 2. 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 3. 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

$$\mathbf{T(n) = 8T(n/2) + bn^2 \ , b > 0}$$

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

Ejemplo 3. Estrategia 3

1) Divida la matriz de entrada (n en potencias de 2) A y B en cuatro matrices de n/2 x n/2

2) Usando tiempo de $\theta(n^2)$ calcule 14 sumas y restas

3) Calcule recursivamente 7 multiplicaciones

$P_i = A_i.B_i$ para i=1,2,...,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: $T(n) = 7T(n/2) + bn^2$ con b > 0

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

preferible para matrices grandes y esparcidas

Ejemplo 3. Estrategia 3

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

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

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

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

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

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

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

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

$C_{12} = m_4 + m_5$

$C_{21} = m_6 + m_7$

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

Multiplica 2 matrices usando 7 multiplicaciones y 14 sumas o restas

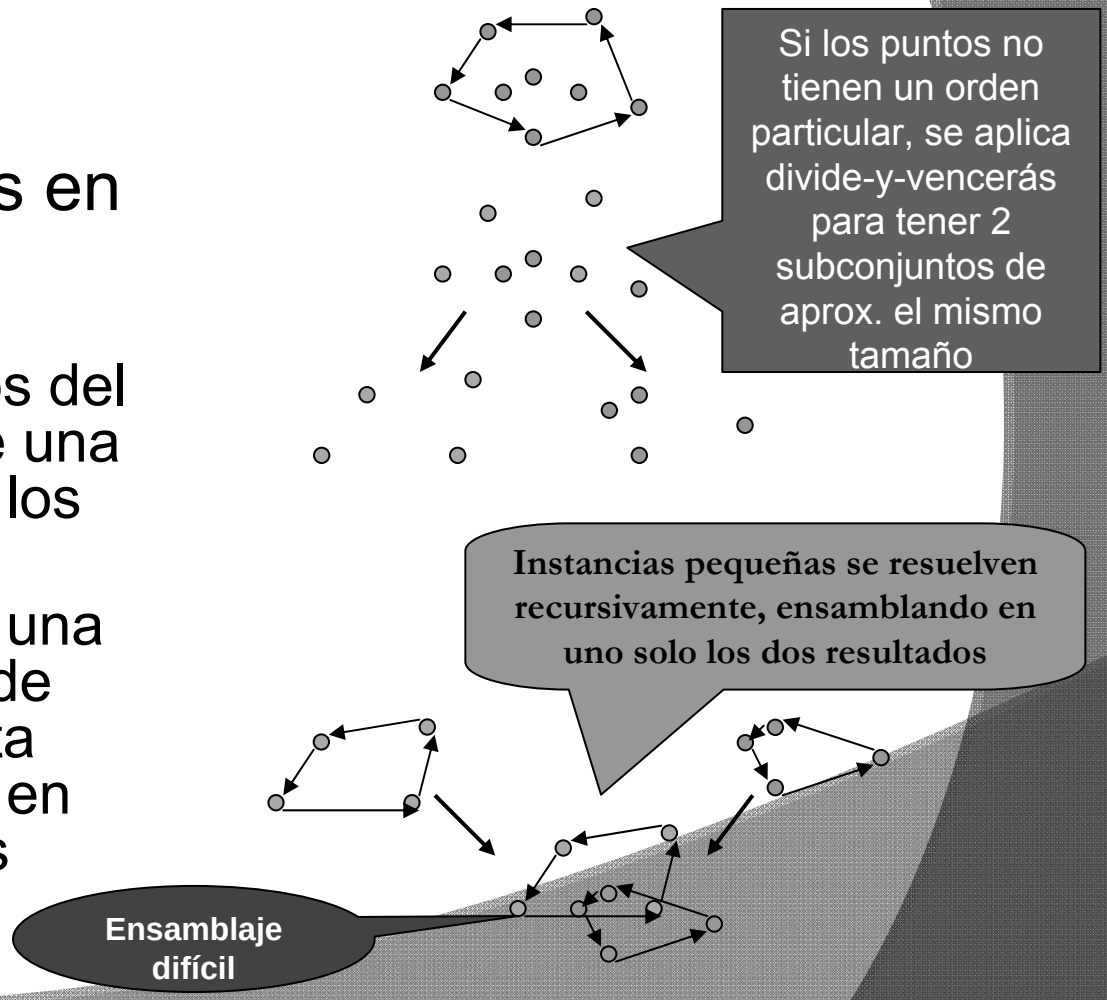
$T(n) \leq mT(n/2) + (a\ n^2)/4$

El más rápido hoy

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

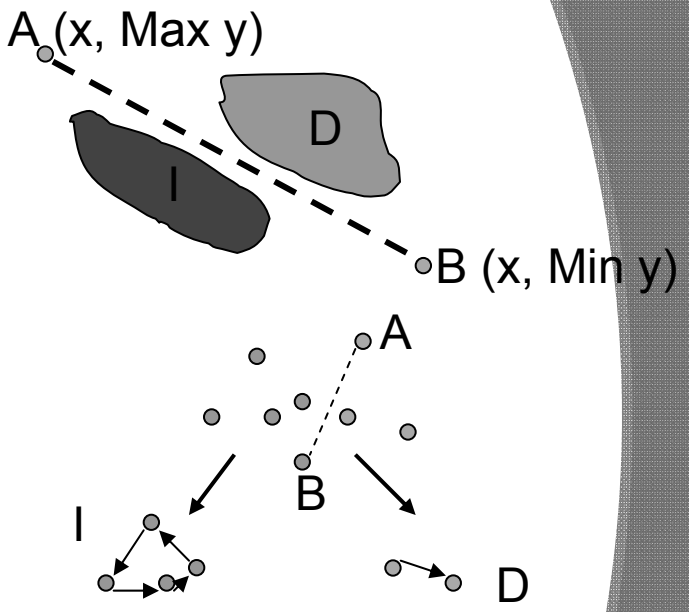
Ejemplo 4. Estrategia 3

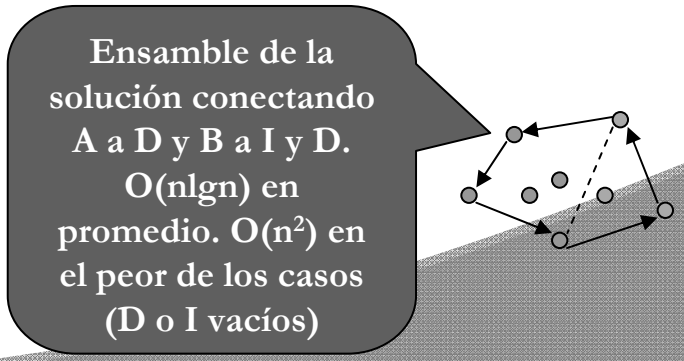
- ✓ Encontrar el cerco convexo de un conjunto de puntos en el plano XY
  - Cerco convexo: secuencia de puntos del conjunto que define una figura convexa que los encierra.
  - Figura convexa: es una figura cerrada, donde cada línea intersecta una figura convexa en no más de 2 puntos



Ejemplo 4. 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

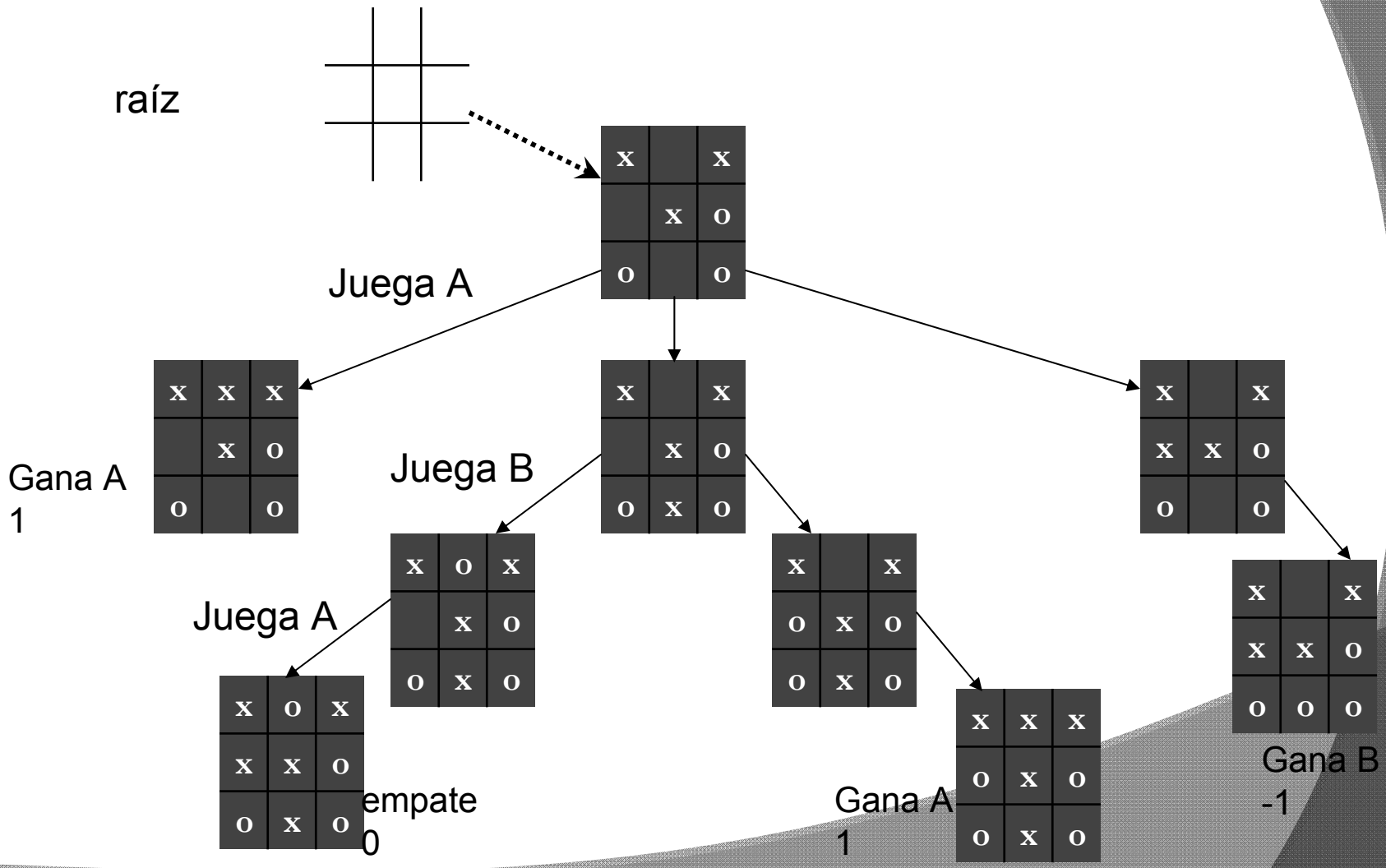




Estrategia 4

- 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



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** **∀ w que sea una solución parcial / $w_i = v_i$ con i = 1, k** **vueltaAtras(w, k+1)** **fsi**	**Documentación**