estudios de postgrado
en computación

Análisis y Diseño de Algoritmos (AyDA)

Isabel Besembel Carrera

TÉCNICAS DE DISEÑO DE ALGORITMOS

Diseño de algoritmos

- Es una **actividad creativa** que no tiene una receta exitosa y precisa.
- Existen muchos problemas importantes para los que no se conoce un algoritmo eficiente
- Clasificando los algoritmos según patrones estructurales similares, es posible identificar ciertas estrategias que normalmente arrojan **algoritmos eficientes y correctos**

1. Proceso de diseño

Estrategias:

1. Estar familiarizado con un repertorio de problemas estándares
   - Producir una especificación clara del problema

     Ignorando detalles como formatos, se debe centrar la atención en lo que parece ser la parte esencial del problema
   - Expresar el problema en una forma abstracta

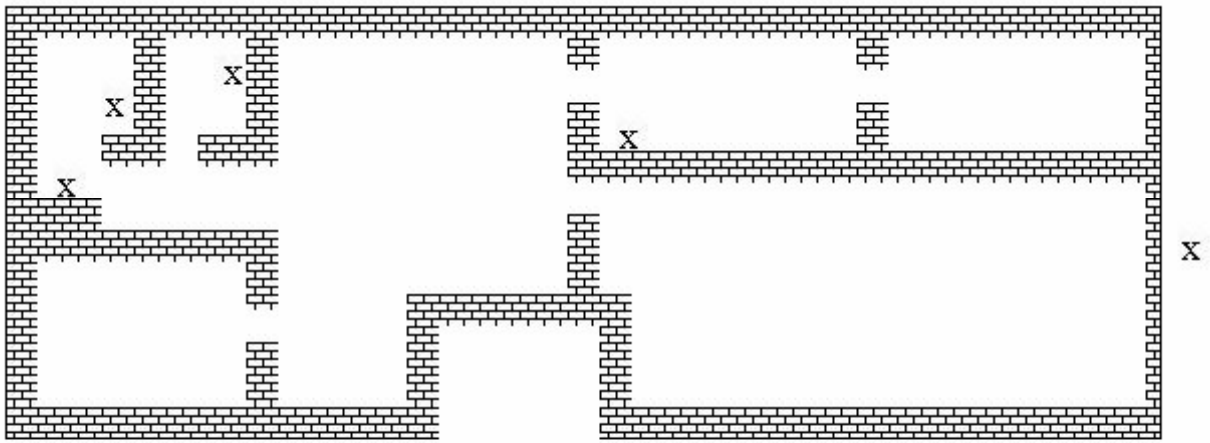
     Normalmente se obtiene uno estándar y no es necesaria innovación alguna

Estrategia 1

Ejemplo:

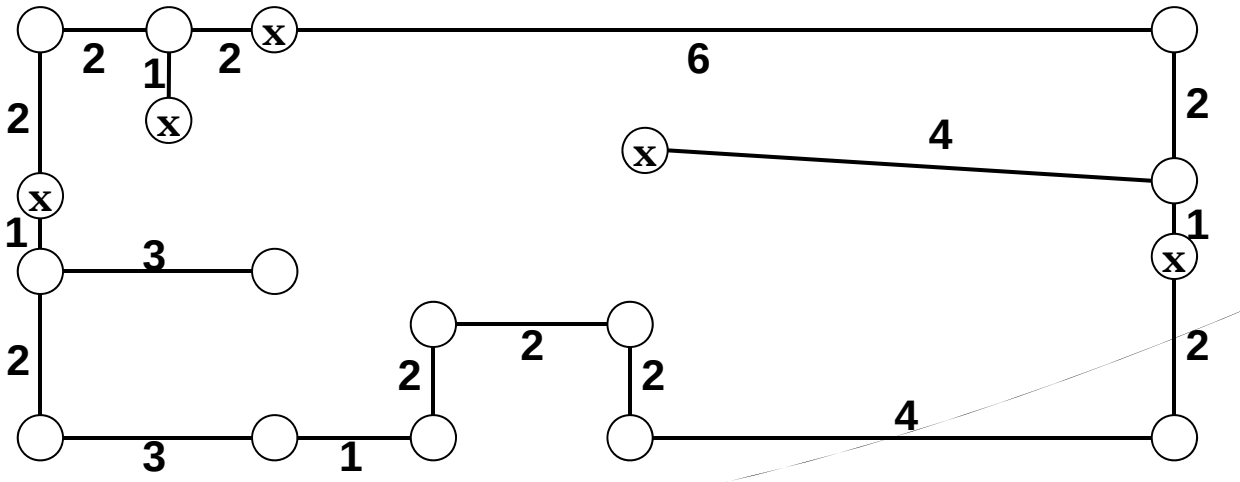
un arquitecto produce el plano mostrado y se desea conocer ¿cuál es la forma más económica de colocar las tuberías de agua?

Los lugares donde se necesita agua y por donde llega a la casa están marcados con una **x**, y solo se pueden colocar tubos en las paredes



Estrategia 1

- Después de abstraer el problema se tiene:
  - Dado un grafo G con costos en sus arcos, encontrar la forma de conectar un subconjunto de sus nodos tal que minimice el costo total de los arcos utilizados
- Problema estándar: Encontrar el árbol de Steiner para G
- No se le conoce aún un algoritmo eficiente, es un problema NP



Estrategia 2

2. Utilizar algoritmos incrementales

- Una instancia de un problema es como un territorio desconocido
- La vía más natural es examinar alguna parte del problema, luego tomar la próxima parte y repetir esto hasta que no hayan partes que examinar.
- Esta estrategia incremental se expresa como:

Junio,04		
	incremental(Tipo: instancia):tipoDeResultado	
	{ pre: precondiciones } **{ pos: poscondiciones }**	
1	**resultado=valor inicial**	**Documentación**
2	**(instancia≠vacía)[X=un elemento de la instancia del problema** **eliminar X de la instancia del problema** **actualizar resultado para que refleje la solución X]**	
3	**regrese resultado**	

Estrategia 2

- Cuando el diseñador/programador decide utilizar esta estrategia debe a continuación considerar **cómo escoger X en cada paso**
  - **Algoritmos incrementales de tipo 1**

    Es irrelevante la escogencia de X
    - **Ventaja**: la selección de X es simple y eficiente
    - **Desventaja**: el algoritmo es ciego, pues no conoce nada de los valores del elemento X que ha seleccionado

    Se caracterizan por tener como **invariante del lazo**:
    - ***resultado*** es una solución completa del subproblema, representado por la parte de la instancia que ha sido eliminada

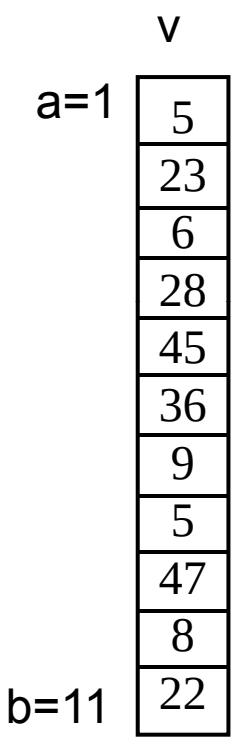
Estrategia 2, tipo 1

- Ejemplos:
  1. Encontrar la suma de los elementos de un arreglo

Junio, 2004 suma(Arreglo(N)DeNatural v, Natural a, Natural b):Natural {pre: a ≤ b+1 } {pos: s ≥ 0 }		
1 2 3	s = 0 (s = s + v(i)) i = a, b, 1 regrese s	➢**i**: Natural. **Contador**. ➢**s:** Natural. Acumulador que tendrá la suma de números en el arreglo v.
1 2	v = (4, 6, 8, 23), a = 1, b = 4 -> s = 41 v = ( ), a = 0, b = 0 -> s = 0	

Invariante del lazo: "s contiene el total de todos los elementos examinados"

Estrategia 2, tipo 1



inicio

i = ? s = 0
i = 1 s = 5
i = 2 s = 28
i = 3 s = 34
i = 4 s = 62
i = 5 s = 107
i = 6 s = 143
i = 7 s = 152
i = 8 s = 157
i = 9 s = 204
i = 10 s = 212
i = 11 s = 234

a=4

b=6

i = ? s = 0
i = 4 s = 28
i = 5 s = 73
i = 6 s = 109

Invariante del lazo: "s contiene el total de todos los elementos examinados"

E ipo 1

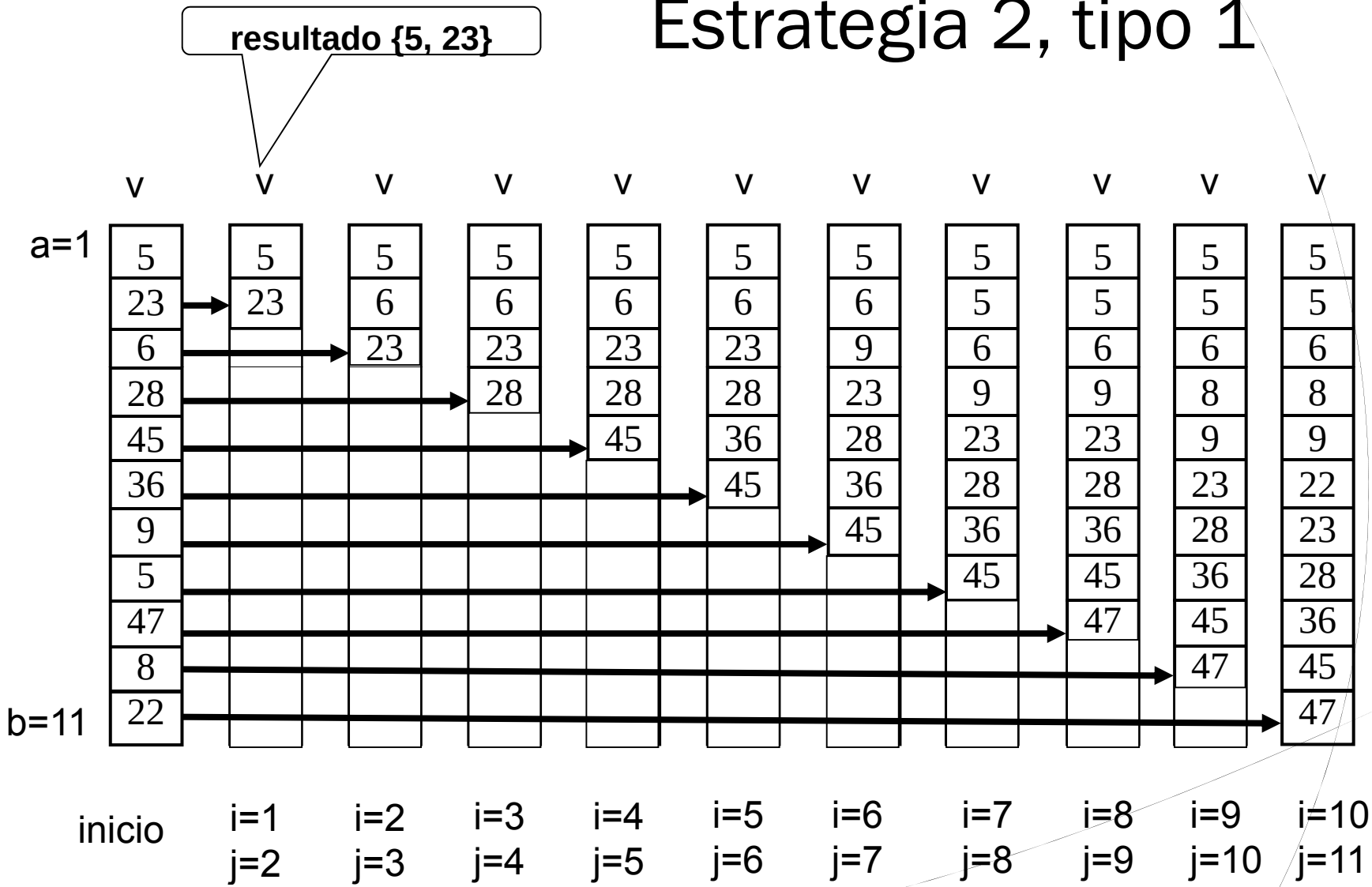
instancia={9,2,6}
1era iteración resultado={9}
2da iteración resultado={2,9}
resultado={2, 6, 9}

2. Ordenar en forma ascendente un conjunto de números con el método de inserción simple

Junio, 2004 insercionSimple(Arreglo(N)DeNatural v, Natural a, Natural b) {pre: a ≤ b+1 } {pos: s ≥ 0 }		
1	((si (v(j) < v(j-1)) entonces k, v(j), v(j-1) = v(j), v(j-1), k fsi) j = i+1, a, -1) i = a, b, 1	➢**i, j**: Natural. **Contadores**. ➢**k:** Natural. Variable auxiliar para realizar el intercambio.
1 2	v = (4, 6, 8, 23), a = 1, b = 4 -> v = (4,6,8,23) v = ( ), a = 0, b = 0 -> v = ( )	Caso exitoso Arreglo vacío

Estrategia de ordenamiento: como ordenar las cartas en una mano cuando las reparten, cada nueva carta se coloca en el sitio que le corresponde

Estrategia 2, tipo 1



Estrategia 2, tipo 2

- **Algoritmos incrementales de tipo 2**

  La escogencia de X es primordial para evitar la reactualización de lo hecho anteriormente.

  Invariante del lazo que los caracteriza:

  - ***resultado*** es una parte de la solución global de la instancia del problema, que podrá ser añadida pero no modificada

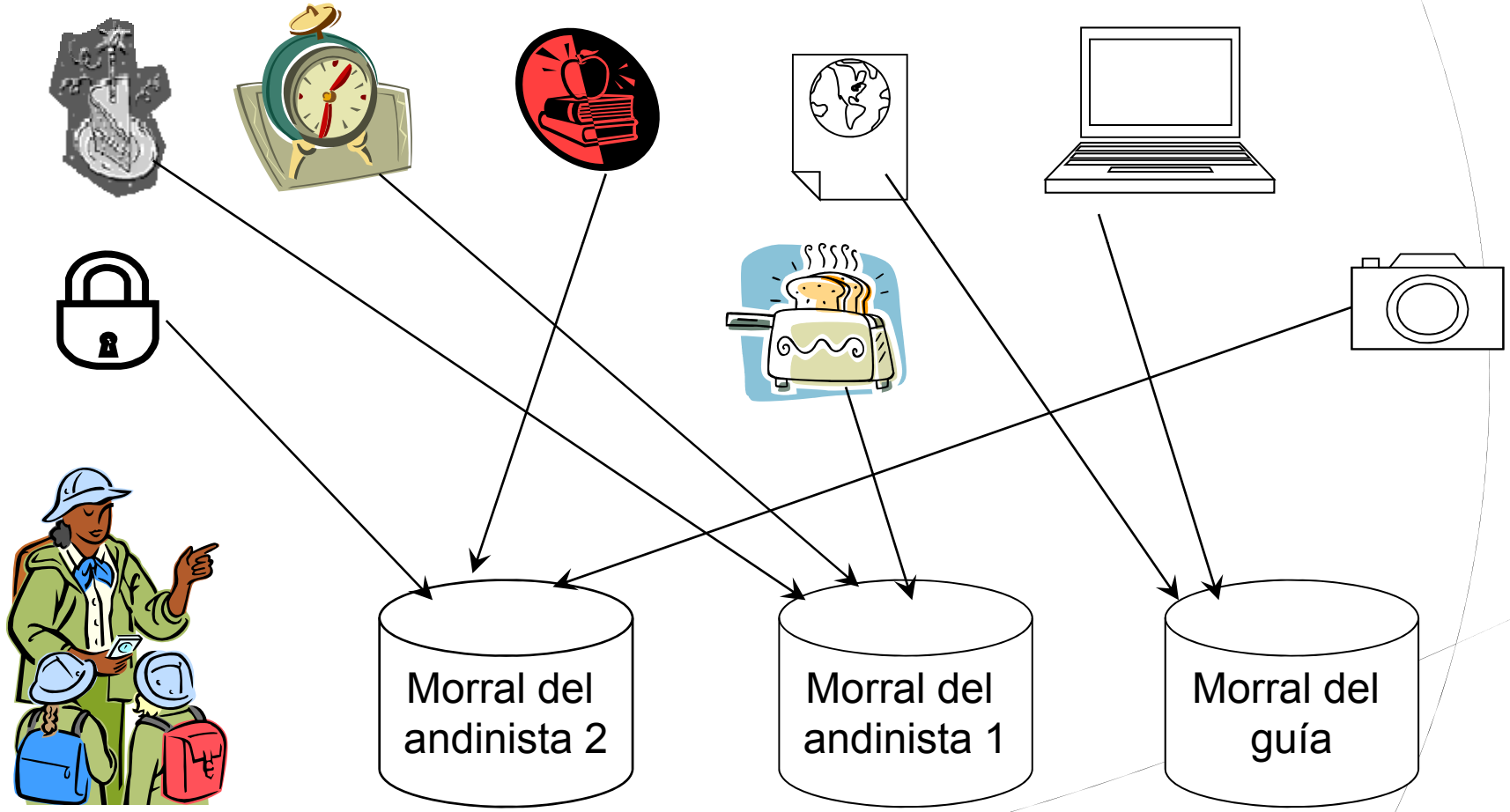
- Ejemplos:

1. Equilibrar el peso de los morrales de un grupo de Andinistas

Estrategia 2, tipo 2

- Se tiene un número de items de pesos variados y se desea que todos los morrales tengan, en lo posible, el mismo peso
- Suponga que ya hay varios items empacados y los morrales tienen aproximadamente el mismo peso, no es simple escoger el próximo item!!! Si éste es muy pesado, reempacar es inevitable!!!
- **Estrategia**: seleccionar los items más pesados primero y tratar de equilibrar con los livianos

Estrategia 2, tipo 2



Problema del morral (Fractional knapsack problem)

- S={i}, con su beneficio o ganancia $b_i \geq 0$ y su peso $w_i \geq 0$, maximizar la ganancia en el subconjunto que no pase de un peso máximo W dado.
- Los items i se pueden dividir o fraccionar
- Cada fracción x_i, $0 \leq x_i \leq w_i$, $\forall i \in S$
  $\sum_{i \in S} x_i \leq W$
- Función objetivo para el beneficio total
  $\sum_{i \in S} b_i(x_i / w_i)$

Estrategia 2, tipo 2

Sep, 2009 morralFrac(Conjunto S, Natural W, Conjunto x):Natural {pre: W>0 ∧ S≠{}} {pos: W>0, w>0 }		
1 2 3 4	(x(i), v(i) = 0, b(i)/w(i)) i ∈ S w=0 (w<W)[remover de S item i con el mayor v(i) a, x(i), w =min{w(i), W-w}, a, w+a] Regrese w	➢**w**: Natural. Peso total que maximiza el beneficio ➢**a:** Natural. Variable auxiliar para la nueva cantidad del item. ➢**x:** Conjunto. Cantidad de los items selccionados para maximizar el beneficio
1 2	S={(2,4), (5,1), (3,2)}, W=6 ⇒ x={0, 1, 0.5} S ={}, W = 3 ⇒ x={}	Caso exitoso Conjunto vacío

Implementación O(n lg n) donde n es el número de items en S,
Con una cola de prioridad basada en un montículo binario por
El mayor valor, con O(lg n)

Morral fraccionado

- Teorema: Dada una colección S de n items, tal que cada item tiene un beneficio bi y un peso wi, se puede construir el subconjunto de S con el beneficio máximo, donde se permiten fracciones de los items, con un peso total de W en O(n lg n)

instancia={9,2,6}
1era iteración resultado={2,9,6}
2da iteración resultado={2,6,9}

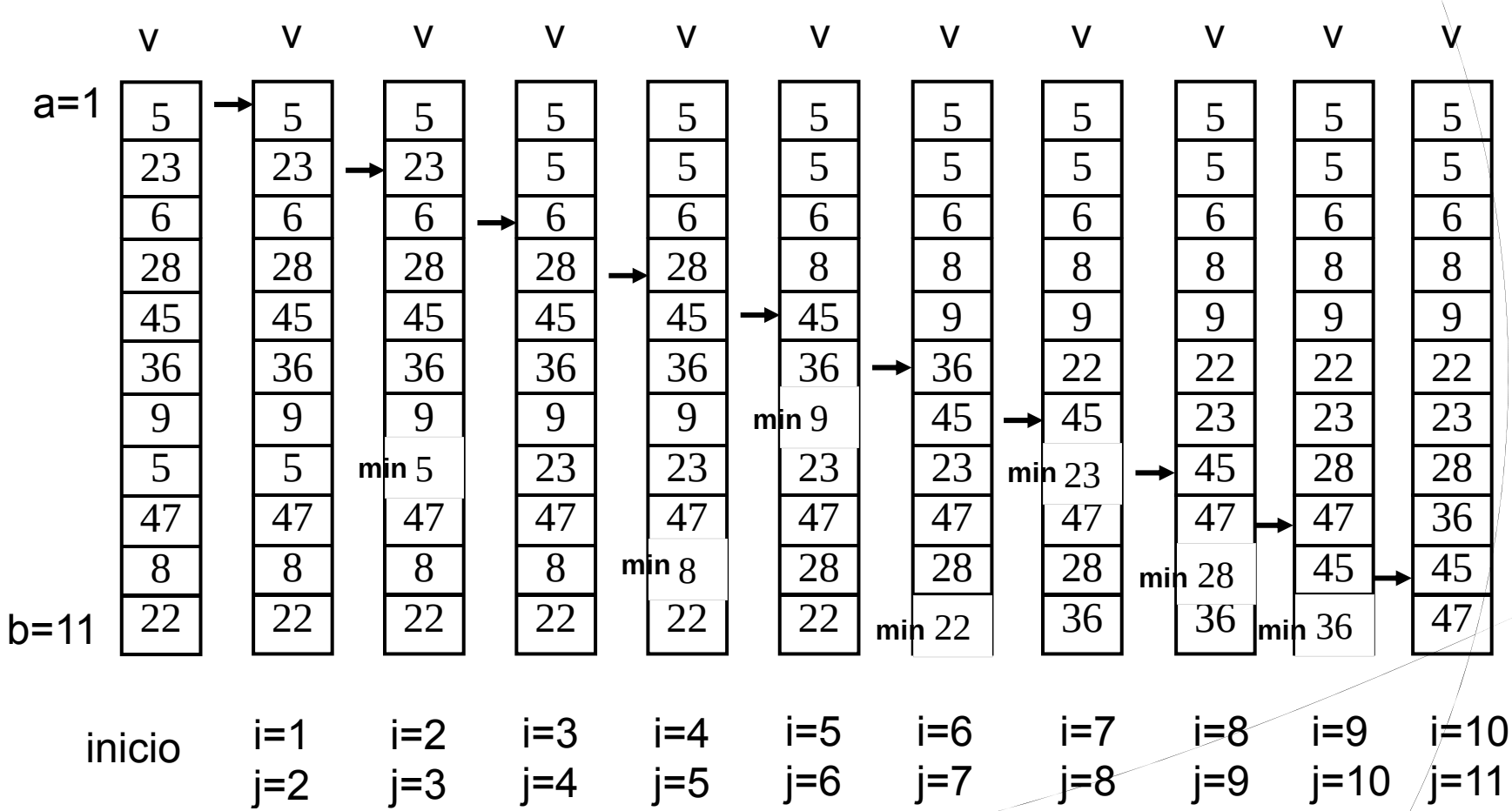
Estrategia 2, tipo 2

2. Ordenar en forma ascendente un conjunto de números con el método de selección simple

Junio, 2004 seleccionSimple(Arreglo(N)DeNatural v, Natural a, Natural b) {pre: a ≤ b+1 } {pos: s ≥ 0 }		
1	((si (v(j) < v(i)) entonces min, pmin = v(j), j fsi) j = i+1, b-1, 1 v(pmin), v(i) = v(i), min) i = a, b, 1	➢**i, j**: Natural. **Contadores**. ➢**min:** Natural. Variable auxiliar para realizar el intercambio. ➢**pmin:** Natural. Posición del valor mínimo
1 2	v=(4, 6, 8, 23), a=1, b=4 -> v=(4,6,8,23) v = ( ), a = 0, b = 0 -> v = ( )	Caso exitoso Arreglo vacío

Estrategia de ordenamiento: se selecciona el más pequeño y se coloca de primero, luego el siguiente más pequeño y se coloca de segundo, ….

Estrategia 2, tipo 2

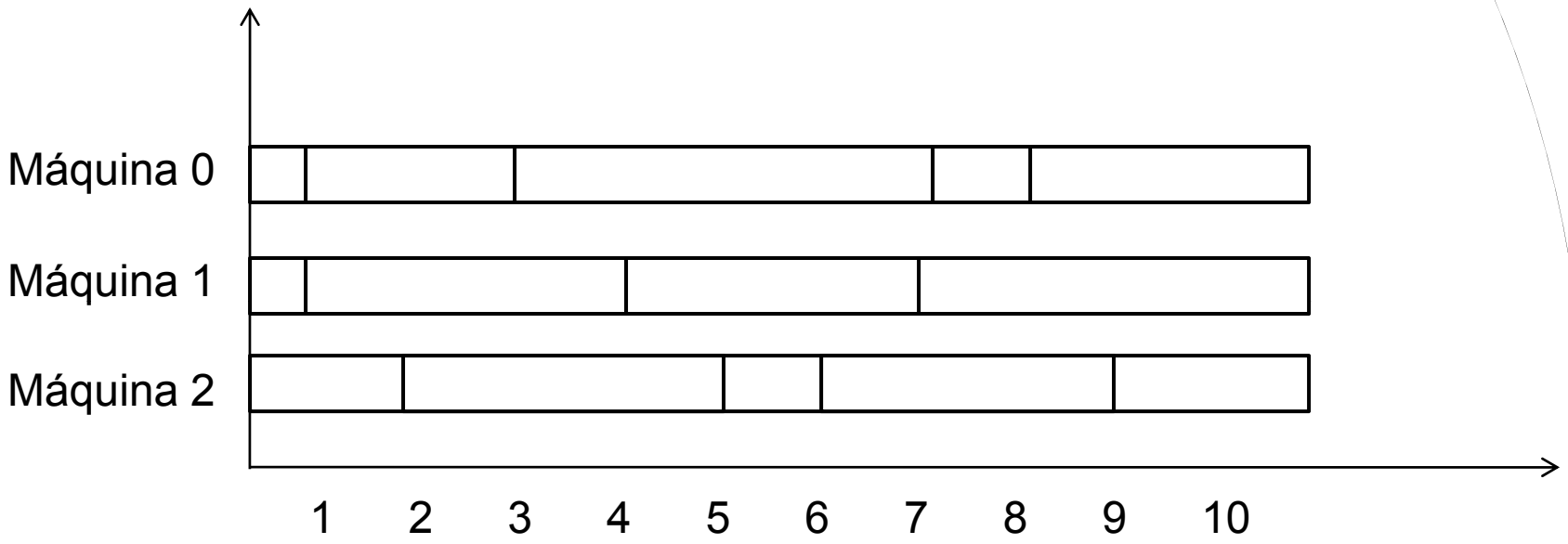


Estrategia 2, tipo 2

Planificación de las actividades o tareas

- Cada actividad tiene definido:
  - Tiempo de inicio: si
  - Tiempo de finalización: fi
  - Donde [si,fi)
- Actividades i, j compatibles si ($si \geq fj$)
- Cada actividad se puede desarrollar en una máquina
- Cada máquina puede efectuar una actividad en cada periodo
- Dos actividades se pueden efectuar en una misma máquina sólo si son compatibles
- Problema: planificar todas las actividades A en la menor cantidad de máquinas posible sin conflicto

Planificación de las actividades



Actividades: (1, 3), (1, 4), (2, 5), (3, 7), (4, 7), (6, 9), (7, 8)

$$A=\{(1,3),(1,4),(2,5),(3,7),(4,7),(6,9),(7,8)\}$$
$$\Rightarrow A=\{(1,3,0),(1,4,1),(2,5,2),(3,7,0),(4,7,1),(6,9,2),(7,8,0)\},\ m=2$$

Estrategia 2, tipo 2

3. Planificación de las actividades o tareas

Sep, 2009	planAct(Conjunto A,): Natural {pre: |A| ≥ 0} {pos: |A| ≥ 0 }	

1
2
 	
3
	m = 0
	(A ≠ {})[remover de A actividad i con menor s(i)
	Si (hay una máquina j cuya actividad
	no esté en conflicto con i) entonces
	planifique la actividad i en la máquina j
	sino
	m=m+1 // anexe una nueva máquina
	planifique la actividad i en la máquina m
	]
	Regrese m
		➢**m**: Natural. Número óptimo de máquinas
		➢**i:** Natural. Selector de la actividad actual.
		➢**j:** Natural. Selector de la máquina.

1 2	A={(1,3),(1,4),(3,4)} ⇒ A={(1,3,0),(1,4,1),(3,4,0)}, m=1 A={} ⇒ A={}, m=0	Caso exitoso no hay actividades

Estrategia 2, tipo 2

Planificación de las actividades o tareas

- ***Teorema:*** dado un conjunto de actividades o tareas definidas con su tiempo de inicio y fin, el algoritmo ***planAct*** produce un plan de actividades con el mínimo número de máquinas en O(n lg n)
- ***Prueba:***
  - suponga que se tiene el plan de actividades en A con k máquinas, pero hay otro plan con k-1 máquinas.
  - Sea k la última máquina utilizada e ***i*** la primera actividad planificada en ella.
  - Por ***planAct***, i está en conflicto con las otras actividades planificadas antes que ella, por ***si***, con ***fj>si*** $\Rightarrow$ que todas están en conflicto$\Rightarrow$ imposible el plan con k-1 máquinas