UNIDAD III. ALGORITMIA AVANZADA (ALGORITMOS DE ORDENAMIENTO)

Francisco J. Hernández López

fcoj23@cimat.mx

ORDENAR

- Acomodar algo en alguna secuencia especifica
- Objetos ordenados:
  - Directorio telefónicos
  - Registro de pacientes
  - Libros en una biblioteca
  - Cosas del hogar
- Se puede definir de la siguiente manera:

Sea A una lista de N elementos $A_1, A_2, A_3, \ldots, A_N$, entonces, ordenar significa permutar estos elementos de tal forma que queden de acuerdo con una distribución preestablecida

- Ascendente: $A_1 \leq A_2 \leq A_3 \leq \cdots \leq A_N$
- Descendente: $A_1 \geq A_2 \geq A_3 \geq \cdots \geq A_N$

TIPOS DE ORDENACIÓN

- Ordenación de arreglos
  - Métodos directos
    - Ordenación por intercambio (método de la burbuja)
    - Ordenación por inserción
    - Ordenación por selección
  - Métodos logarítmicos
    - Quicksort
    - Heapsort
  - Métodos lineales
    - Counting sort
    - Radix sort
    - Bucket sort
- Ordenación de archivos

Métodos Directos

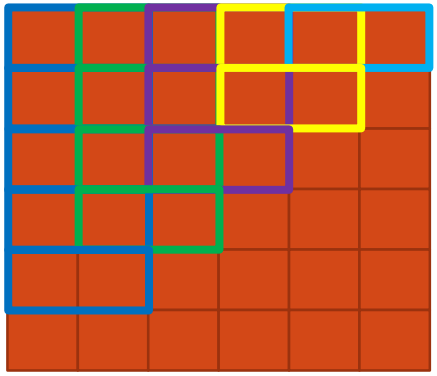
MÉTODO DE LA BURBUJA

- También conocido como: intercambio directo
- Se le llama burbuja por la forma en que se van acomodando los datos
- Fácil de programar y entender
- Poco eficiente (fuerza bruta)
- La idea básica del algoritmo consiste en comparar pares de elementos adyacentes e intercambiarlos entre sí hasta que se encuentren ordenados

ALGORITMO DE LA BURBUJA

- El siguiente algoritmo ordena los elementos de un arreglo A de menor a mayor:

1. Repetir con i desde 1 hasta $N - 1$
   1. Repetir con j desde 0 hasta $N - i$
      1. Si $(A[j] > A[j+1])$ entonces
         1. $aux \leftarrow A[j]$
         2. $A[j] \leftarrow A[j+1]$
         3. $A[j+1] \leftarrow aux$
      2. Fin si
   2. Fin ciclo
2. Fin ciclo



$$O\left(\frac{N(N-1)}{2}\right)$$

$$O(N^2)$$

ALGORITMO DE LA BURBUJA CON SEÑAL

1. Hacer $i \leftarrow 1$ y $ban \leftarrow 0$
2. Mientras $(i \leq N - 1)$ y $(ban == 0)$ repetir
    1. Hacer $ban \leftarrow 1$
    2. Repetir con j desde 0 hasta $N - i$
        1. Si $(A[j] > A[j + 1])$ entonces
            1. $aux \leftarrow A[j]$
            2. $A[j] \leftarrow A[j + 1]$
            3. $A[j + 1] \leftarrow aux$
            4. $ban \leftarrow 0$
        2. Fin si
    3. Fin ciclo
    4. $i \leftarrow i + 1$
3. Fin ciclo

$O(N^2)$

Método de la sacudida (Shaker sort)

- Es una optimización del método de la burbuja
- La idea es mezclar las dos formas en que se pueden realizar las comparaciones del método de la burbuja
- Hay dos etapas en cada iteración:
  - De Derecha a Izquierda (pasando los elementos pequeños hacia la izquierda)
  - De Izquierda a Derecha (pasando los elementos grandes hacia la derecha del arreglo)
- Una variable auxiliar almacena la posición del último elemento intercambiado
- Se finaliza cuando ya no hay más intercambios o cuando el contenido de la variable que guarda el extremo izquierdo es mayor que el de la variable derecha

ALGORITMO DE LA SACUDIDA

1. Hacer $IZQ \leftarrow 1, DER \leftarrow N-1$ y $k \leftarrow N-1$
2. Mientras $(DER \geq IZQ)$ repetir
    1. Repetir con i desde DER hasta IZQ
        1. Si $(A[i-1] > A[i])$ entonces
            1. $swap(A[i-1], A[i])$
            2. k$\leftarrow i$
        2. Fin si
    2. Fin ciclo
    3. Hacer $IZQ \leftarrow k+1$
    4. Repetir con i desde IZQ hasta DER
        1. Si $(A[i-1] > A[i])$ entonces
            1. swap$(A[i-1], A[i])$
            2. k$\leftarrow i$
        2. Fin si
    5. Fin ciclo
    6. Hacer $DER \leftarrow k-1$
3. Fin ciclo

$swap(a, b)$
Inicio
1. $aux \leftarrow a$
2. $a \leftarrow b$
3. $b \leftarrow aux$

Fin

$O(N^2)$

MÉTODO DE INSERCIÓN DIRECTA

- También conocido como el método de la baraja
- Algoritmo
  1. Repetir con i desde 1 hasta $N - 1$
     1. Hacer $AUX \leftarrow A[i]$ y $k \leftarrow i - 1$
     2. Mientras $(k \geq 0)$ y $(AUX < A[k])$ repetir
        1. Hacer $A[k + 1] \leftarrow A[k]$ y $k \leftarrow k - 1$
     3. Fin ciclo
     4. Hacer $A[k + 1] \leftarrow AUX$
  2. Fin ciclo

$O(N^2)$

Método de Selección Directa

- Más eficiente que los presentados anteriormente

- Idea:
  - Buscar el menor y colocarlo en la primer posición
  - Luego se busca el segundo menor y se coloca en la segunda posición
  - Y así sucesivamente…

ALGORITMO DE SELECCIÓN DIRECTA

1. Repetir con i desde 0 hasta $N - 2$
   1. Hacer $MENOR \leftarrow A[i]$ y $k = i$
   2. Repetir con j desde $i + 1$ hasta $N - 1$
      1. Si $(A[j] < MENOR)$ entonces
         1. Hacer $MENOR \leftarrow A[j]$ y $k \leftarrow j$
      2. Fin si
   3. Fin ciclo
   4. Hacer $A[k] \leftarrow A[i]$ y $A[i] \leftarrow MENOR$
2. Fin ciclo

Número de movimientos en el arreglo: $N - 1$

Tiempo de ejecución:
$O(N^2)$

Métodos Logarítmicos

MÉTODO QUICKSORT

- También conocido como:
  - método rápido
  - ordenación por partición

- Idea:
  - Se toma un elemento X de una posición cualquiera del arreglo
  - Se trata de ubicar a X en la posición correcta del arreglo, de tal forma que:
    - Los elementos a la izquierda sean menores o iguales que X
    - Los elementos a la derecha sean mayores o iguales que X
  - Se repiten los pasos anteriores pero ahora para los elementos que están a la izquierda y a la derecha de X (podemos pensar en recursividad)
  - El proceso termina cuando todos los elementos se encuentran en su posición correcta

ALGORITMO QUICKSORT

- $Quicksort(INI, FIN)$
  1. Hacer $IZQ \leftarrow INI$, $DER \leftarrow FIN$, $POS \leftarrow INI$ y $ban \leftarrow 1$
  2. Mientras $(ban == 1)$ repetir
     1. Hacer $ban = 0$
     2. Mientras $(A[POS] \leq A[DER])$ y $(POS \neq DER)$ repetir
        1. Hacer $DER \leftarrow DER - 1$
     3. Fin ciclo
     4. Si $(POS \neq DER)$ entonces
        1. $swap(A[POS],\ A[DER])$
        2. $POS \leftarrow DER$
        3. Mientras $(A[POS] \geq A[IZQ])$ y $(POS \neq IZQ)$ repetir
           1. Hacer $IZQ \leftarrow IZQ + 1$
        4. Fin ciclo
        5. Si $(POS \neq IZQ)$ entonces
           1. $ban = 1$
           2. $swap(A[POS],\ A[IZQ])$
           3. $POS \leftarrow IZQ$
        6. Fin si
     5. Fin si
  3. Fin ciclo
  4. Si $(POS - 1 > INI)$ entonces
     1. $Quicksort(INI, POS - 1)$
  5. Fin si
  6. Si $(FIN > POS + 1)$ entonces
     1. $Quicksort(POS + 1, FIN)$
  7. Fin si

Tiempo de ejecución, en promedio: $O(NlogN)$

Tiempo de ejecución, en el peor caso: $O(N^2)$

MÉTODO HEAPSORT

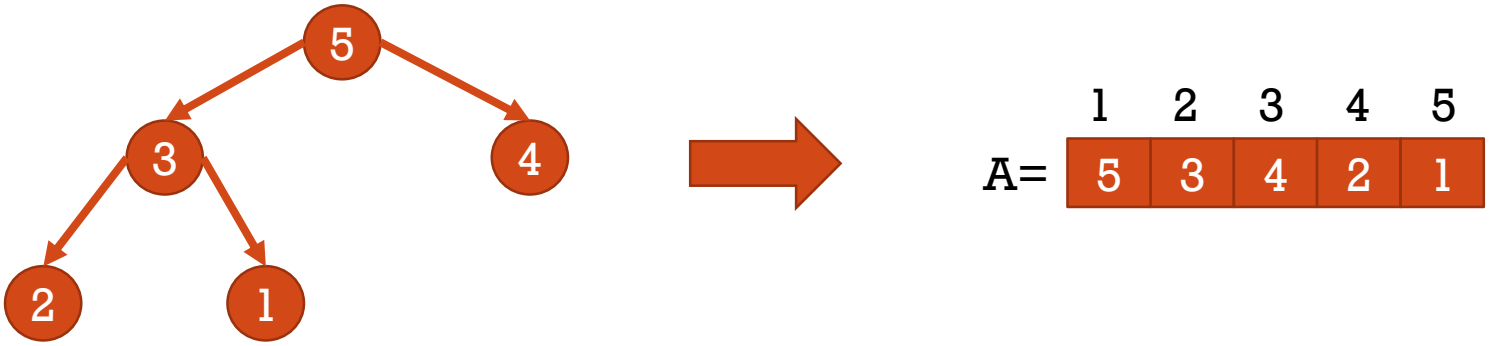
- También conocido como método del montículo

- Trabaja con árboles y considera dos operaciones:
  - Construir un montículo
  - Eliminar la raíz del montículo en forma repetida

- Un montículo debe cumplir la siguiente regla:
  - Para todo nodo del árbol se debe cumplir que su valor sea mayor o igual que el valor de cualquiera de sus hijos

MÉTODO HEAPSORT

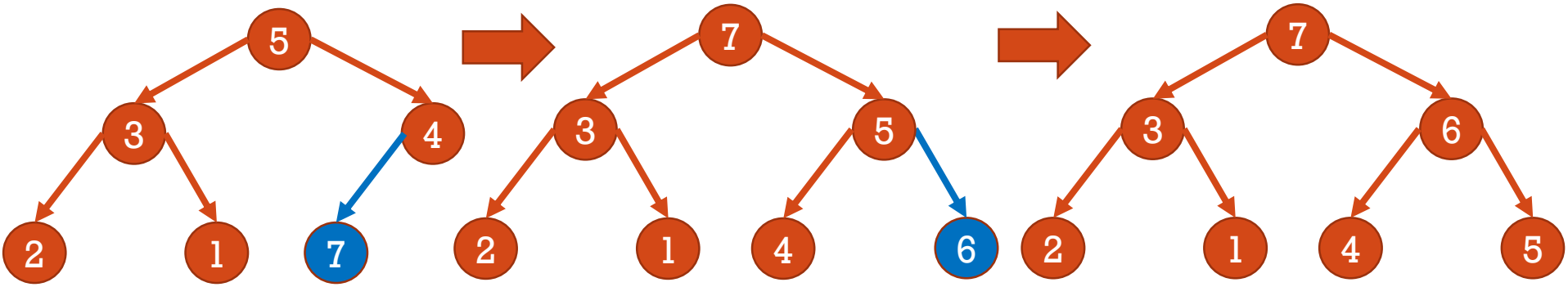
- Para representar un montículo en un arreglo lineal se debe tener en cuenta para todo nodo k lo siguiente:
  - El nodo k se almacena en la posición k correspondiente del arreglo
  - El hijo izquierdo del nodo k se almacena en la posición $2 * k$
  - El hijo derecho del nodo k se almacena en la posición $2 * k + 1$



Inserción de un elemento en un montículo

- Se inserta el elemento en la primera posición disponible
- Se verifica si su valor es mayor que el de su padre, si se cumple esta condición entonces se efectúa el intercambio, si no se deja en la posición donde quedó. (esto se hace de forma recursiva desde las hojas hacia la raíz)

Ejemplo: Insertar el 7 y el 6



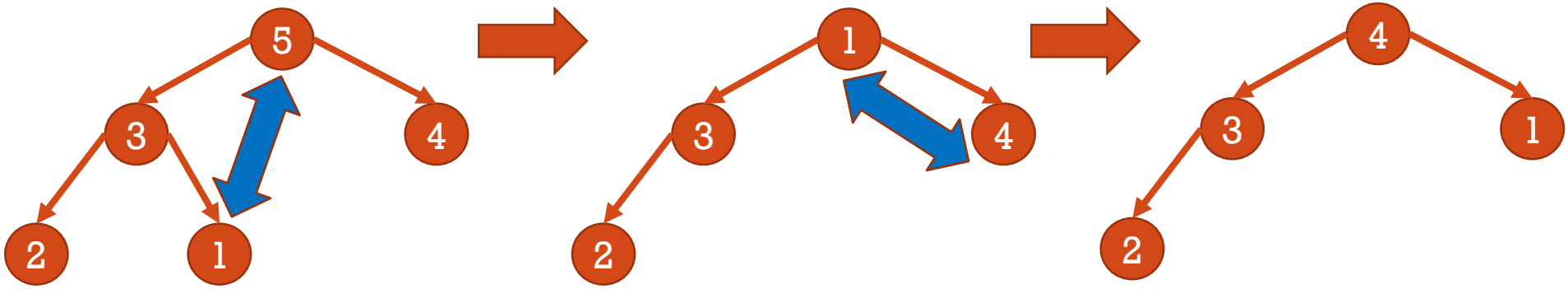
ALGORITMO PARA INSERTAR ELEMENTOS EN UN MONTÍCULO

- $inserta\_monticulo(A, N)$
  1. Repetir con i desde 2 hasta N
     1. Hacer $k \leftarrow i$ y $ban \leftarrow 1$
     2. Mientras $(k > 1)$ y $(ban == 1)$ repetir
        1. Hacer $ban \leftarrow 0$
        2. Si $A[k] > A[\lfloor k/2 \rfloor]$ entonces
           1. $swap(A[k],\ A[\lfloor k/2 \rfloor])$
           2. $k = \lfloor k/2 \rfloor$
           3. $ban \leftarrow 1$
        3. Fin si
     3. Fin ciclo
  2. Fin ciclo

Eliminación de un montículo

- Se reemplaza la raíz con el elemento que ocupa la última posición del montículo
- Se verifica si el valor de la raíz es menor que el valor más grande de sus hijos, si se cumple la condición, entonces se efectúa el intercambio, si no, se deja en la posición donde quedó. (esto se hace de forma recursiva desde la raíz hacia las hojas)

Ejemplo: Eliminar el 5



ALGORITMO PARA ELIMINAR UN MONTÍCULO

- $elimina\_monticulo(A, N)$
  1. Repetir con i desde N hasta 2
     1. Hacer $AUX \leftarrow A[i], A[i] \leftarrow A[1], IZQ \leftarrow 2, DER \leftarrow 3, k \leftarrow 1$ y $ban \leftarrow 1$
     2. Mientras $(IZQ < i)$ y $(ban == 1)$ repetir
        1. Hacer $MAYOR \leftarrow A[IZQ]$ y $AP \leftarrow IZQ$
        2. Si $(MAYOR < A[DER])$ y $(DER \neq i)$ entonces
           1. Hacer $MAYOR \leftarrow A[DER]$ y $AP \leftarrow DER$
        3. Fin si
        4. Si $AUX < MAYOR$ entonces
           1. Hacer $A[k] \leftarrow A[AP]$ y $k \leftarrow AP$
        5. Si no
           1. Hacer $ban \leftarrow 0$
        6. Fin si
        7. Hacer $IZQ \leftarrow k * 2$ y $DER \leftarrow IZQ + 1$
     3. Fin ciclo
     4. Hacer $A[k] \leftarrow AUX$
  2. Fin ciclo

ALGORITMO HEAPSORT

- $heapsort(A, N)$
  1. $inserta\_monticulo(A, N)$
  2. $elimina\_monticulo(A, N)$

Para analizar el tiempo de ejecución es importante tomar en cuenta tanto la parte de creación del montículo así como su eliminación

En ambas partes el tiempo de ejecución del algoritmo es: $O(NlogN)$

$heapsort$ garantiza que en el peor caso, su tiempo de ejecución es $O(NlogN)$ a diferencia de $quicksort$

Métodos Lineales

Método Counting Sort

- Asume que cada uno de los N elementos a ordenar es un entero en el rango $0\ to\ k$, para algún entero k
- Cuando $k = O(N)$, la ejecución del método es del orden $O(N)$
- Para cada elemento x se determina el número de elementos menores a x. Usando esta información, es posible posicionar al elemento x directamente en el arreglo

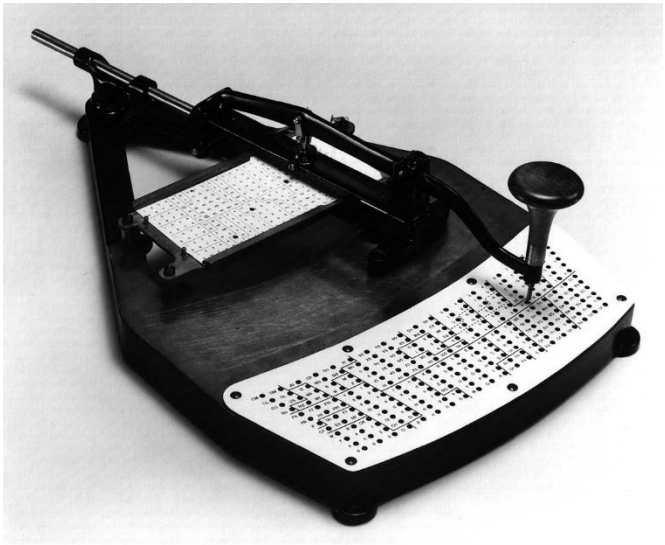
ALGORITMO COUNTING SORT

- $counting\_sort(A, B, k, N)$
  1. Sea $C[0, .., k]$ un nuevo arreglo
  2. Para $i = 0$ hasta k hacer
     1. $C[i] = 0$ → $O(k)$
  3. Fin ciclo
  4. Para $j = 1$ hasta N hacer
     1. $C[A[j]] = C[A[j]] + 1$ → $O(N)$
  5. Fin ciclo
  6. Para $i = 1$ hasta k hacer
     1. $C[i] = C[i] + C[i - 1]$ → $O(k)$
  7. Fin ciclo
  8. Para $j = N$ hasta 1 hacer
     1. $B\left[C[A[j]]\right] = A[j]$ → $O(N)$
     2. $C[A[j]] = C[A[j]] - 1$
  9. Fin ciclo

Si $k = O(N)$ entonces el orden del tiempo de ejecución del algoritmo será: $O(N)$

MÉTODO RADIX SORT

- Se utilizó en las máquinas tabuladoras que se encargaban de ordenar las tarjetas perforadas



Herman Hollerith (1887) → Censo de los EU

MÉTODO RADIX SORT

- Idea:
  - Ordenar los números en 10 casillas (0 al 9), comenzando por el digito menos significativo (de derecha a izquierda)
  - Luego, se unen todas las pilas de números considerando que los números de la casilla 0 deben quedar antes que los de la casilla 1 y así sucesivamente hasta la casilla 9
  - Entonces se vuelve a repetir el procedimiento, pero ahora considerando el segundo dígito menos significativo

Algoritmo:

$radix\_sort(A, d)$

1. Para $i = 1$ hasta d
   1. Ordenar A considerando el digito i
2. Fin ciclo

Puede ser: $O(N)$

Método Bucket Sort

- Suponer que se quiere ordenar N datos y además se sabe que siguen una distribución uniforme en el intervalo $[0,..k]$

- La idea es dividir el intervalo en N pedazos (buckets) y hacer corresponder cada número a estos pedazos, por ejemplo a un número j le corresponde el pedazo $\left\lfloor \frac{Nj}{k} \right\rfloor$

- Se puede implementar usando un *arreglo*, suponiendo que cada pedazo es una posición del arreglo y que cuando hay algún valor que va a dar a la misma posición, se puede utilizar una *lista enlazada*

ALGORITMO BUCKET SORT

- $bucket\_sort(A, N, k)$
  1. Sea $B[0, .., N]$ un nuevo arreglo
  2. Para $i = 0$ hasta N
     1. Hacer $B[i]$ una lista vacía
  3. Fin ciclo
  4. Para $i = 1$ hasta N
     1. Insertar $A[i]$ en la lista $B\left[\left\lfloor \frac{NA[i]}{k} \right\rfloor\right]$
  5. Fin ciclo
  6. Para $i = 0$ hasta N
     1. Ordenar la lista $B[i]$ con Inserción directa
  7. Fin ciclo
  8. Concatenar las listas $B[0], B[1], \ldots, B[N]$ en orden

Ya que los datos siguen una distribución uniforme, entonces las listas son pequeñas y se puede esperar que el método sea: $O(N)$