Estructuras de datos y Algoritmos II

UNIDAD 1: Algoritmos de ordenamiento

Profesor: Ing. Jonathan Roberto Torres Castillo

Ingenieria en Computación

Universidad Nacional Autónoma de México

Contenido

La unidad 1 de algoritmos de ordenamiento esta dividida en:

1. Algoritmos de ordenamiento básicos
2. Algoritmos de ordenamiento rápidos
3. MargeSort

Introducción

- Los problemas más comunes en la informática son los de búsqueda y de ordenación.
- Número de búsquedas diarias en Google 3500 millones (2017).
- La eficiencia de la búsqueda es importante.
- Ordenar un grupo de datos es arreglarlos en algún orden secuencial ya sea en forma ascendente o descendente, con el fin de acelerar la búsqueda.

Tipos de ordenamiento

INTERNO

- Se lleva a cabo completamente en la memoria de acceso aleatorio de alta velocidad de la computadora es decir todos los elementos que se ordenan caben en la memoria principal de la computadora.

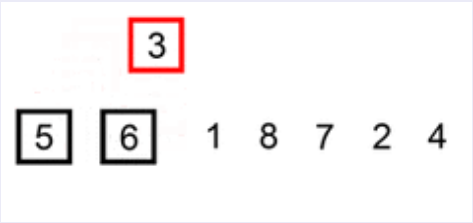
EXTERNO

- Cuando no cabe toda la información en memoria principal es necesario ocupar memoria secundaria y en este caso es necesario realizar un ordenamiento externo.

Algoritmos de ordenamiento de información

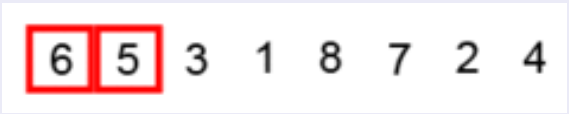
Algoritmos de Inserción

- Se considera un elemento a la vez y cada elemento es insertado en la posición apropiada con respecto a los demás elementos que ya han sido ordenados.



Algoritmos de Intercambio

- Se trabaja con parejas de elementos que se van comparando y se intercambian si no están en el orden adecuado. El proceso se realiza hasta que se han revisado todos los elementos del conjunto a ordenar.



Algoritmos de ordenamiento de información

Algoritmos de Selección

- En estos algoritmos se selecciona el elemento mínimo o el máximo de todo el conjunto a ordenar y se coloca en la posición apropiada. Esta selección se realiza con todos los elementos restantes del conjunto.

Algoritmos de Enumeración

- Se compara cada elemento con todos los demás y se determina cuántos son menores que él. La información del conteo para cada elemento indicará su posición de ordenamiento.

Algoritmo BubbleSort

El método de ordenación por burbuja (BubbleSort en inglés) es uno de los más básicos, simples y fáciles de comprender, pero también es poco eficiente.

¿En que consiste?

La técnica se denomina ordenación por burbuja, esta consiste en comparar todos los elementos de una lista contra todos, si se cumple que uno es mayor a otro entonces se intercambia de posición. Los valores más pequeños suben hacia la cima o parte superior de la lista, mientras que los valores mayores se hunden en la parte inferior.

Descripción Algoritmo

Para una lista a de n elementos numerados de 0 a $n-1$, el algoritmo requiere $n-1$ pasadas. Por cada pasada se comparan elementos adyacentes (parejas sucesivas) y se intercambian sus valores cuando el primer elemento es mayor que el segundo. Al final de cada pasada el elemento mayor se dice que ha burbujeado hasta la cima de la sub-lista actual.

Descripción Algoritmo

BubbleSort

- Sea la lista $a_0, a_1, a_2 ... a_{n-1}$, entonces:
- En la pasada 0 se comparan elementos adyacentes: (a_0, a_1), (a_1, a_2), (a_2, a_3), ..., (a_{n-2}, a_{n-1})
- Se realizan $n-1$ comparaciones por cada pareja (a_i, a_{i+1}) y se intercambian si $a_{i+1} < a_i$. Al final el mayor elemento de la lista estará situado en la posición $n-1$
- En la pasada 1 se realizan las mismas comparaciones e intercambios, terminando con el segundo elemento de mayor valor el cual estará situado en la posición $n-2$.
- Se realizan las siguientes pasadas de forma similar y el proceso termina con la pasada $n-1$ donde se tendrá elemento más pequeño en la posición 0.

Actividad

Aplicar el algoritmo a la lista $\{9, 21, 4, 40, 10, 35\}$.

Primera Pasada

$\{\mathbf{9,21}, 4, 40, 10, 35\} -- > \{\mathbf{9,21}, 4, 40, 10, 35\}$ No se realiza intercambio

$\{9, \mathbf{21,4}, 40, 10, 35\} -- > \{9, \mathbf{4,21}, 40, 10, 35\}$ Intercambio 21 y el 4

$\{9, 4, \mathbf{21,40}, 10, 35\} -- > \{9, 4, \mathbf{21,40}, 10, 35\}$ No se realiza intercambio

$\{9, 4, 21, \mathbf{40,10}, 35\} -- > \{9, 4, 21, \mathbf{10,40}, 35\}$ Intercambio 40 y el 10

$\{9, 4, 21, 10, \mathbf{40,35}\} -- > \{9, 4, 21, 10, \mathbf{35,40}\}$ Intercambio 40 y el 35

Actividad

Aplicar el algoritmo a la lista $\{9, 21, 4, 40, 10, 35\}$.

Segunda Pasada

$\{\mathbf{9,4}, 21, 10, 35, 40\} -- > \{\mathbf{4,9}, 21, 10, 35, 40\}$ Intercambio 9 y el 4
$\{4, \mathbf{9,21}, 10, 35, 40\} -- > \{4, \mathbf{9,21}, 10, 35, 40\}$ No se realiza intercambio
$\{4, 9, \mathbf{21,10}, 35, 40\} -- > \{4, 9, \mathbf{10,21}, 35, 40\}$ Intercambio 21 y el 10
$\{4, 9, 10, \mathbf{21,35}, 40\} -- > \{4, 9, 10, \mathbf{21,35}, 40\}$ No se realiza intercambio
$\{4, 9, 10, 21, \mathbf{35,40}\} -- > \{4, 9, 10, 21, \mathbf{35,40}\}$ No se realiza intercambio

Aunque la lista ya está ordenada se harían otras 3 pasadas.

Algoritmo en pseudocódigo

```
bubbleSort(A, n)
 Para i=1 hasta n -1
  Para j=0 hasta i < n -2
   Si (a[j]>a[j+1]) entonces
    tmp=a[j]
        a[j]=a[j+1]
        a[j+1]=tmp
    Fin Si
   Fin Para
  Fin Para
 Fin bubbleSort
```

En este pseudocódigo se considera una lista con elementos, las dos estructuras de repetición se realizan $n-1$ veces es decir se hacen $n-1$ pasadas y $n-1$ comparaciones en cada pasada.

Algoritmo en pseudocódigo

Una posible mejora consiste en añadir una bandera que indique si se ha producido algún intercambio durante el recorrido del arreglo y en caso de que no se haya producido ninguno, el arreglo se encuentra ordenado y se puede abandonar el método.

```
bubbleSort2 (a, n)
Inicio
bandera = 1;
pasada=0
Mientras pasada < n-1 y bandera es igual a 1
```

Algoritmo en pseudocódigo

```
bandera = 0
 Para j = 0 hasta j < n-pasada-1
  Si a[j] >a[j+1] entonces
   bandera = 1
   tmp= a[j];
   a[j] = a[j+1];
   a[j+1] = tmp;
  Fin Si
 Fin Para
 pasada=pasada+1
Fin Mientras
Fin
```

Contenido 2^{da} sesión

1. Aplicaciones de los algoritmos de ordenación.
2. Descripción del Algoritmo Merge Sort.
3. Principio de Dividir
4. Principio de ordenar
5. Principio de Mezclar o intercalar.

Aplicaciones: Algoritmos de Búsqueda

RECORDANDO

Aplicaciones comunes

1. Mostrar un listado ordenado del contenido de una base de datos (Listados ordenados alfabéticamente).
2. Ordenar los resultados de una busqueda en internet .
3. Mostrar los ficheros de un directorio.

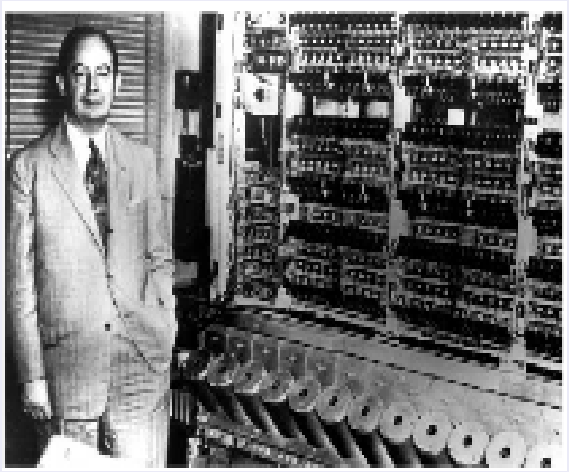
Aplicaciones: Algoritmos de Búsqueda

Otras aplicaciones no tan comunes

1. Encontrar la mediana.
2. Encontrar el par mas cercano.
3. Identificar valores repetidos.
4. Detectar anomalías.

MergeSort

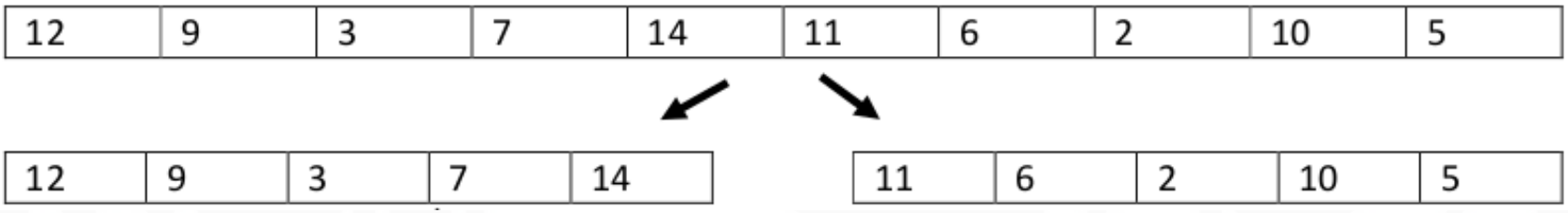
Algoritmo de ordenación 'Divide y vencerás': Descripción y características



1. Es un algoritmo recursivo con un número de comparaciones mínimo. El tiempo de ejecución promedio es $O(Nlog(N))$.
2. **Desventaja:** Trabaja sobre un array auxiliar lo cual tiene dos consecuencias: *uso de memoria extra y trabajo extra consumido en las copias entre arreglos (aunque es un trabajo de tiempo lineal)*.

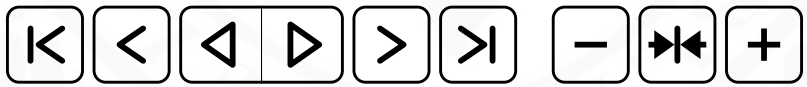
MergeSort: Dividir y ordenar

1. **'Divide y vencerás'** plantea que un problema puede ser dividido en varios **subproblemas** y una vez resueltos estos se puede proceder a **unir las soluciones** para formar la solución del problema general.



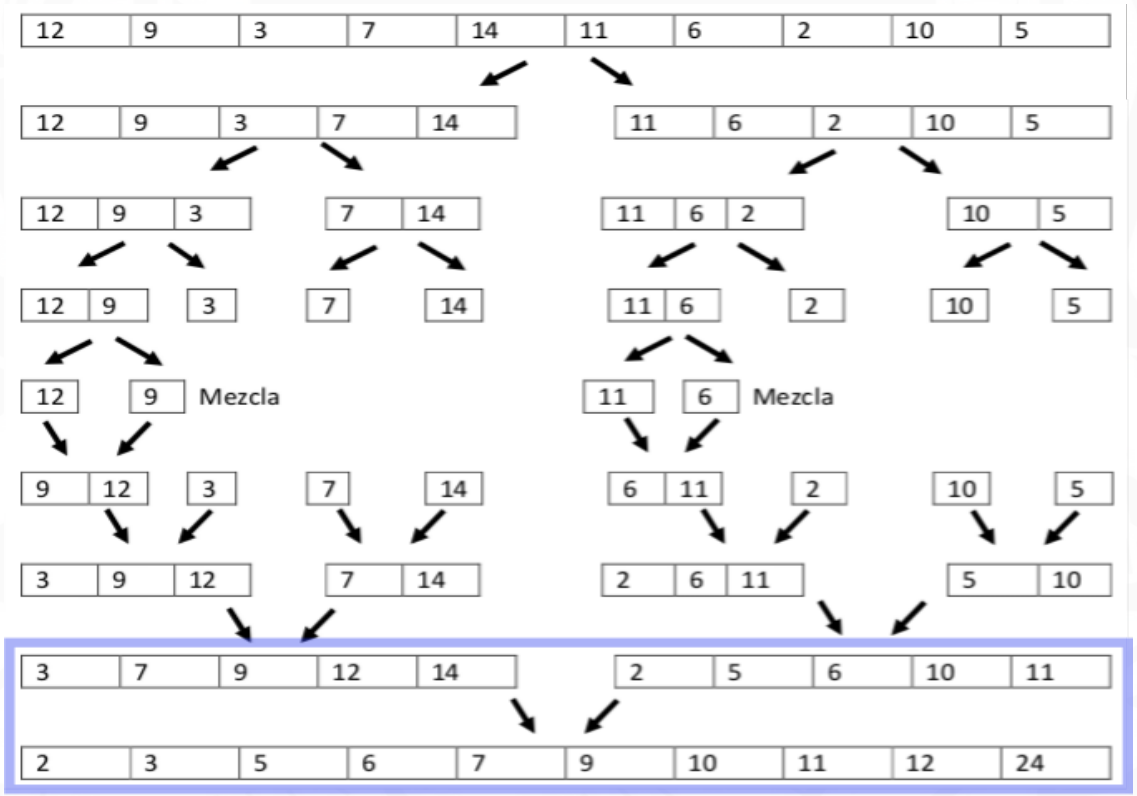
La solución de los **subproblemas** más pequeños se realiza de la misma manera: es decir, se van resolviendo problemas cada vez más pequeños (hasta llegar a un caso base: **problema no divisible** con solución **tri-bial**).

6 5 3 1 8 7 2 4



MergeSort

Cada recursión toma un arreglo de elementos desordenados. Se divide en dos mitades, se aplica la recursión en cada una de estas y ya que al finalizar estas recursiones tenemos las dos mitades ordenadas se **intercalan ambas** para obtener el arreglo ordenado.

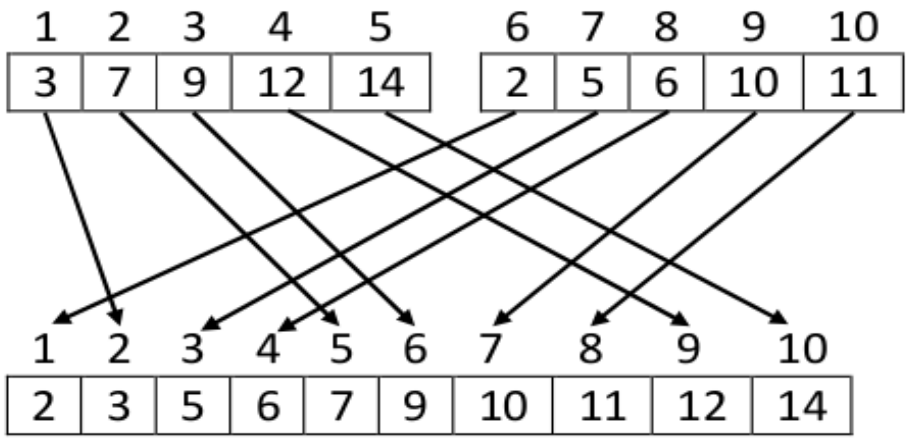


INTERCALAR

MergeSort: Mezclar-intercalar

Intercalar (Merge)

Es la operación que le da el nombre a este algoritmo. La intercalación toma dos secuencias (arreglos) de elementos y a partir de estas construye una tercera secuencia que contiene todos los elementos de estas en orden.



MergeSort: Algoritmo

El algoritmo de ordenamiento por mezcla o intercalación de un arreglo `A` consta de dos partes como se mostró anteriormente, el de división y ordenación (`MergeSort(A,p,r)`) y el de intercalación o mezcla (`Merge(A,p,q,r)`):

`MergeSort(A,p,r)`

```
MergeSort(A,p,r)
Inicio
 Si p<r entonces
   q=(p+r)/2
   MergeSort(A, p , q)
   MergeSort(A, q+1 , r)
   Merge(A,p,q,r)
Fin
```

MergeSort: Algoritmo

```
Merge(A,p,q,r)
Inicio
Formar subarray Izq[0...q-p] y subarray Der[0...r-q]
Copiar de A[p...q] a Izq[0...q-p] y A[q+1...r]
... a Der[0...r-q-1]
i=0, j=0
Para k=p hasta r
    Si (j>=r-q) o (i<q-p+1 y Izq[i]<Der[j]) entonces
        A[k]=Izq[k]
        i=i+1
    En otro caso
        A[k]=Der[j]
        j=j+1
    Fin Si
Fin Para
Fin
```