Algoritmos de Ordenamiento

Algoritmos

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort

Algoritmo de la Burbuja (bubblesort)

- El algoritmo más popular y básico para ordenar un conjunto de “n” elementos en un arreglo

```
BUBBLESORT(A)
1 for i ← 1 to length[A]
2     do for j ← length[A] downto i + 1
3            do if A[j] < A[j - 1]
4                   then exchange A[j] ↔ A[j - 1]
```

Algoritmo de la Burbuja

```
void bubbleSort(int numbers[], int array_size)
{ int i, j, temp;
  for (i = (array_size - 1); i >= 0; i--)
   { for (j = 1; j <= i; j++)
      { if (numbers[j-1] > numbers[j])
        { temp = numbers[j-1];
          numbers[j-1] = numbers[j];
          numbers[j] = temp;
        }
      }
   }
}
```

Complejidad Algoritmo Burbuja

```
void bubbleSort(int numbers[], int
    array_size)
{ int i, j, temp;
  for (i = (array_size - 1); i >= 0; i--)
   { for (j = 1; j <= i; j++)
      { if (numbers[j-1] > numbers[j])
        { temp = numbers[j-1];
          numbers[j-1] = numbers[j];
          numbers[j] = temp;
        }
      }
   }
}
```

i	j
n-1	n-1
n-2	n-2
n-3	n-3
…	…
0	0

$$total = \sum_{k=1}^{n-1} k = \sum_{k=1}^{n} k - n = \frac{n(n+1)}{2} - n$$

Algoritmo de la Burbuja

- Claramente, el algoritmo esta en $O(n^2)$
- Este algoritmo se puede optimizar para el mejor caso, en cuyo caso puede comportarse como un algoritmo lineal ¿Cómo se modificaría el código?

Algoritmo de Selección

- Algoritmo en el cual, por cada ciclo de ejecución se busca colocar en la posición “i” el menor valor entre los valores que se encuentran desde la posición “i” hasta la posición “n”
  - Por cada pasada, se identifica el menor valor en la parte del arreglo que no se ha procesado
  - En cada pasada, la dimensión del arreglo a procesar disminuye en 1

Ejemplo Algoritmo de Selección

8	6	1	4	9	2	5	3	0	7

0	6	1	4	9	2	5	3	8	7
0	1	6	4	9	2	5	3	8	7
0	1	2	4	9	6	5	3	8	7
0	1	2	3	9	6	5	4	8	7
0	1	2	3	4	6	5	9	8	7
0	1	2	3	4	5	6	9	8	7
0	1	2	3	4	5	6	9	8	7
0	1	2	3	4	5	6	7	8	9

Algoritmo Selección

- ¿Cuál es el peor caso para el algoritmo de selección?
- ¿Cuál es su mejor caso?
- Al igual que el método de la burbuja, este algoritmo se encuentra en $O(n^2)$

Complejidad Algoritmo Selección

```
void Seleccion(int A[],int N)
{
 int i,menor,k,j;
 for(i=0;i<=N-2;i++)
 { menor=A[i];
   k=i;
   for(j=i+1;j<=N-1;j++)
     { if(A[j]<menor)
        { menor=A[j];
          k=j;
        }
     }
  A[k]=A[i];
  A[i]=menor;
 }
}
```

- Se considera el bucle más interno (if (A[j]<menor))
  - Tiempo de ejecución: c
- Por cada pasada de "i", el bucle interno se ejecuta: n-(i+1)+1 = n – i
- El tiempo del bucle interno es: t(i) <= (n-i)c
- Por cada pasada de "i", el bucle externo se tarda: b + t(i), donde b es el número de operaciones realizadas en el bucle
- Por lo anterior, el tiempo total es menor o igual a:

$$\sum_{i=1}^{n-1} b + (n-i)c = \sum_{i=1}^{n-1} (b+cn) - c\sum_{i=1}^{n-1} i =$$

$$(n-1)(b+cn) - cn(n-1)/2 = \frac{1}{2}cn^2 + \left(b - \frac{1}{2}c\right)n - b$$

Complejidad Algoritmo Selección

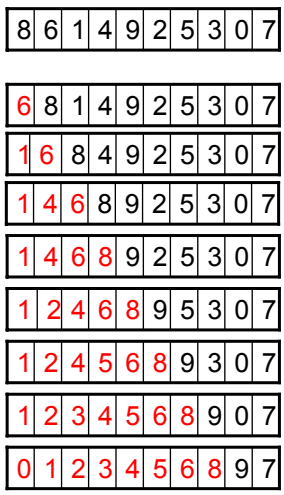
- Si se utiliza un barómetro (instrucción más interna), la expresión se simplifica a:

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}1=\sum_{i=1}^{n-1}(n-i)=\sum_{k=1}^{n-1}k=\frac{n(n-1)}{2}$$

Algoritmo Inserción

- En este algoritmo, el arreglo se va ordenando de izquierda a derecha
- Se ubica el orden correcto de un valor, recorriendo los valores más grandes una posición a la derecha y colocando el valor ordenado en su posición correcta

Algoritmo de Inserción



Análisis Algoritmo de Inserción

- A diferencia del algoritmo de Selección, el tiempo de ejecución del Algoritmo de Inserción depende del orden original de los elementos
- El orden máximo de este algoritmo, al igual que los dos algoritmos anteriores, esta en $O(n^2)$

Análisis Algoritmo de Inserción

```
void Insertar(int A[], int n)
{ int i, j, x;
 for (i=1; i<n; i++)
  { x = A[i];
    j = i-1;
    while((j>=0) && (x<A[j]))
      { A[j+1] = A[j];
        j = j-1;
      }
    A[j+1] = x;
  }
}
```

- Consideremos como barómetro la instrucción j>=0 y x<A[j]
- Para el valor "i", en el peor caso x = A[i] se tiene que comparar con los valores A[i-1], …, A[0] (i veces)
- Por tanto, el número total de operaciones a realizar es:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} - 1$$

Algoritmo QuickSort

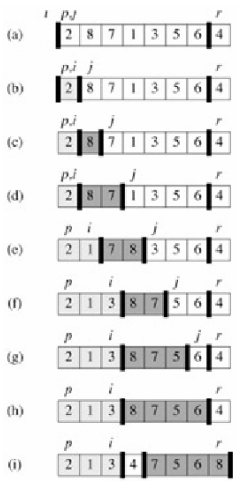
- Uno de los mejores algoritmos para ordenar un arreglo
- Quicksort es un algoritmo en $O(n^2)$, sin embargo, en promedio se comporta como un algoritmo O(n logn)

Algoritmo QuickSort

```
QUICKSORT(A, p, r)
1  if p < r
2     then q ← PARTITION(A, p, r)
3          QUICKSORT(A, p, q − 1)
4          QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
1  x ← A[r]
2  i ← p − 1
3  for j ← p to r − 1
4       do if A[j] ≤ x
5            then i ← i + 1
6                 exchange A[i] ↔ A[j]
7  exchange A[i + 1] ↔ A[r]
8  return i + 1
```

Algoritmo QuickSort



Actividad

- Implementar el algoritmo QuickSort en C
- Realizar una investigación del algoritmo ordenar por fusión (mergesort), identificando:
  - Idea principal de cómo trabaja el algoritmo
  - Complejidad del algoritmo
  - Ejemplo de una implementación en C