**Quicksort**

***Definición:***

El método Quicksort basa su estrategia en la idea intuitiva de que es más fácil ordenar una gran estructura de datos subdividiéndolas en otras más pequeñas introduciendo un orden relativo entre ellas.

El ordenamiento por partición (Quick Sort) se puede definir en una forma más conveniente como un procedimiento recursivo. Tiene aparentemente la propiedad de trabajar mejor para elementos de entrada desordenados completamente, que para elementos semiordenados.

***Descripción del algoritmo:***

Se debe elegir un elemento de la lista de elementos a ordenar que será el pivote.

La idea central del algoritmo consiste en: se toma cualquier posición de elemento x.

Se trata de ubicar a x en la posición correcta del arreglo, de tal forma que todos los elementos que se encuentran a su izquierda sean menores o iguales a x y todos los elementos que se encuentren a su derecha sean mayores o iguales a x.

Se repiten los pasos anteriores, pero ahora para los conjuntos de datos que se encuentran a la izquierda y a la derecha de la posición correcta de x en el arreglo.

Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

***Análisis del algoritmo:***

**Ventajas:** Es rápido, no requiere memoria adicional

**Desventajas:**La implementación es un poco mas complicada, utiliza muchos métodos.

**Complejidad Computacional:**

En el mejor de los casos tiene un costo de O (n\*log (n)). Que es cuando el pivote siempre queda al medio del arreglo.

En el peor de los casos tiene un costo de O(n^2). Cuando el pivote siempre se inclina hacia a un lado, es decir, genera un arreglo de sólo un elemento y una segunda con el resto de los elementos.

En el caso promedio también tiene un costo de O (n\*log (n)). Se produce cuando el pivote se inclina más hacia un lado y los 2 subarreglos tienen distinto tamaño de elementos.

**Ejemplo:**

class Program

{

int h;

int[] vector;

private void mostrar()

{

Console.WriteLine("\nVector ordenados en forma ascendente");

for (int i = 0; i < h; i++)

{

Console.Write("\nValor= "+ vector[i]);

}

Console.ReadLine();

}

public void llenar(int n)

{

h = n;

vector = new int[h];

Random r = new Random();

for (int i = 0; i < h; i++)

{

vector[i] = r.Next(0,3000);

Console.Write("\nValor random= "+vector[i]);

}

quicksort(vector, 0, h - 1);

mostrar();

}

static void Main(string[] args)

{

int n;

Console.WriteLine("Quick Sort");

Console.Write("Cuantos longitud del vector: ");

n = Int32.Parse(Console.ReadLine());

Program h = new Program();

h.llenar(n);

h.mostrar();

Console.ReadKey();

}

public void quicksort(int[] vec,int izq, int der)

{

int i, j, pivote, aux;

i = izq;

j = der;

pivote = vec[(izq + der) / 2];

do

{

while (vec[i] < pivote && i < der)

{

i++;

}

while (vec[j] > pivote && j > izq)

{

j--;

}

if (i <= j)

{

aux = vec[i];

vec[i] = vec[j];

vec[j] = aux;

i++;

j--;

}

} while (i <= j);

if (izq < j)

quicksort(vec,izq, j);

if (i < der)

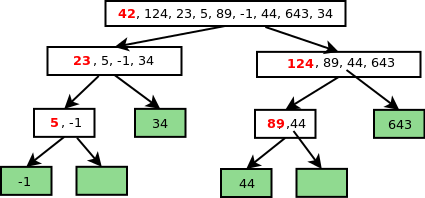
quicksort(vec,i, der);

}

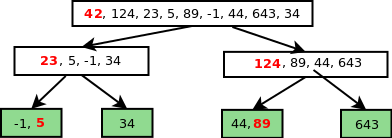
}

*Suponiendo que los valores random que eligió la computadora son {42, 124, 23, 5, 89, -1, 44, 643, 34}y que la magnitud del vector que elegio era de 8.*

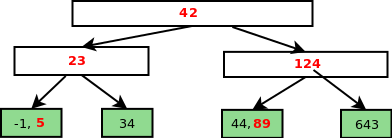
Se forma un árbol de llamados recursivos. En rojo se va a mostrar el pivote. A izquierda las sublistas menores, y a la derecha las sublistas mayores.



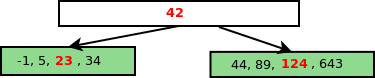
En verde están los casos bases, es la lista de un solo elemento, ya que son únicos se resolverán fácil y se concatenara con los pivotes de arriba, así empiezan las llamadas recursivas.



En la segunda fila se deja solos pivotes, que son los que están en rojo.



La tercera fila ya esta ordenada, por lo tanto, se va a retornar a la segunda fila para que se pueda concatenar con el pivote de la segunda, y quedara en medio.



Se tienen las dos listas ordenadas, una de menores y otra de mayores, y el pivote que es el que se concatenará en medio de los 2, y se podrá concluir el orden.



**Mergesort**