Andrea Estefanía Galván Irigoyen

Matricula 1735583

Matemáticas Computacionales

Reporte sobre métodos de ordenamiento

(Bubble, Insertion, Selection y Quicksort)

Bubble

El procedimiento que hace este método primero acomoda un numero moviendo el de mayor valor hasta la última posición comenzando desde el numero en la posición cero, o sea el primer número, de la lista que queremos acomodar, primero compara el primer elemento con el segundo y si el primero es mayor que el segundo entonces se intercambian y se repite esta acción con cada número hasta el último, cuando ya se haya acomodado el más grande sigue para encontrar  y acomodar el otro más grande y va comparando los números desde el inicio de la lista y así sigue hasta acomodar todo los elementos el arreglo, este método es más tardado que otros ya que aun con los numero ya acomodados al momento de acomodar otros vuelve a compararlos con los ya ordenados.

Código en Python

lista = [4, 12, 8, 2, 1]

for recorrido in range(1,len(lista)):

for posicion in range(len(lista) - recorrido):

if lista [posicion] > lista [posicion + 1]:

temp = lista [posicion]

lista[posicion] = lista[posicion + 1]

lista[posicion + 1] = temp

print lista

Lo malo de este método es que es más lento que otros métodos y aparte realiza muchas comparaciones e intercambios.

La complejidad es n\*n = O(n^2)

Insertion

Primero se toma al segundo elemento de la lista y lo acomoda en la posicion en la que pertenece con respecto al primero, después de esto toma al tercer elemento y lo acomoda en la posicion que pertenece con respecto al primer y segundo elemento, y así sucesivamente se va con cada elemento de la lista y comparando valores con los otros elementos que se encuentren a su izquierda hasta llegar a su posicion correspondiente.

La función principal recorre la lista desde el segundo elemento hasta el último, y cuando uno de estos elementos no está ordenado con respecto al anterior, llama a la función auxiliarreubicar, que se encarga de colocar el elemento en la posición que le corresponde.

En la función reubicar se busca la posición correcta donde debe colocarse el elemento, a la vez que se van corriendo todos los elementos un lugar a la derecha, de modo que cuando se encuentra la posición, el valor a insertar reemplaza al valor que se encontraba allí anteriormente.

Ordenar por inserción una lista de tamaño N puede insumir (en el peor caso) tiempo del orden de N^2. En cuanto al espacio utilizado, nuevamente sólo se tiene en memoria la lista que se desea ordenar y algunas variables de tamaño 1.

def insercion (lista) :

for i in range(1,len(lista)) :

aux=lista[i]

j=i

while j>0 and aux<lista[j-1] :

lista[j]=lista[j-1]

j-=1

lista[j]=aux

return lista

Selección

primero se busca el elemento de menor valor de la lista, ya cuando se haya localizado se intercambia con el elemento que se encuentra en la primera posicion, después se busca el segundo elemento con menor valor y se intercambia por el segundo elemento y así hasta haber acomodado todos los elementos de la lista, para encontrar el de menor valor se van comparando numero por número de la lista hasta que se haya encontrado el menor, pero no se compara con los que ya se hayan intercambiado en las primeras posiciones ya que esos ya están en su lugar correspondientes.

Python def seleccion(lista)

n = len(lista)

for i in range(0,n-1):

k = i

t = lista[i]

for j in range(i,n):

if lista[j] < t:

k = j

t = lista[j]

lista[k] = lista[i]

lista[i] = t

return lista

Quicksort

En este método lo primero que hace es agarrar un elemento cualquiera y tomarlo como pivote después lo que hace es checar todos los elementos uno por uno y los de menor valor colocarlos a la izquierda del pivote y los de mayor valor colocarlos a la derecha del pivote, así el elemento seleccionado como pivote ya está en su posicion correspondiente, después tanto en la parte izquierda como en la derecha con referencia al primer pivote se vuelve a tomar algún elemento como pivote e igual se checan los numero y los de menor valor van hacia la izquierda y los de mayor valor hacia la derecha y así hasta que ya estén acomodados todos los elementos de la lista.

En este método las operaciones que realiza son de 0(n log n), el peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas.

Código

def particion(lista, izq, der):

global comparaciones

pivote = lista[der]

indice = izq

for i in xrange(izq, der):

comparaciones += 1

if lista[i] <= pivote:

lista[indice], lista[i] = lista[i], lista[indice]

indice += 1

lista[indice], lista[der] = lista[der], lista[indice]

return indice

def quicksort(lista, izq, der):

if izq < der:

pivote\_indice = particion(lista, izq, der)

quicksort(lista, izq, pivote\_indice-1)

quicksort(lista, pivote\_indice+1, der)

lista = [36, 71, 16, 21, 73, 9, 0, 40, 66, 5]

comparaciones = 0

t0 = time()

quicksort(lista, 0, len(lista)-1)

t1 = time()

print "Lista ordenada:"

print lista, "\n"

print "Tiempo: {0:f} segundos".format(t1 - t0)

print "Comparaciones:", comparaciones