

Algoritmos de ordenación

Nombre: Alexandra Valeria Hernández Quintero

Matricula: 1663507

Unidad de aprendizaje: Matemáticas Computacionales

Fecha: 17/09/2017

Resumen: Este reporte trata sobre de los 4 tipos de algoritmos de ordenación que vimos en clase: Bubble sort, insercion sort, selection sort y quick sort.

Bubble sort:

El tipo de algoritmo “burbuja” (su traducción en español), es el más ineficiente y “lento” podríamos decir, como ya vimos anteriormente en clases, en si no es que tarde más que los otros algoritmos, más bien que hace más operaciones, su complejidad es de n^2 , el peor de los casos seria que el arreglo estaría ordenado al revés, es decir que si se quiere ordenar de manera ascendente, el arreglo este ordenado de forma descendente y viceversa. En si lo que hace este algoritmo es agarrar los primeros dos números compararlos,” preguntar” cuál de los dos es el menor (o mayor), para luego moverlos o dejarlos donde estaban, según sea el caso, asi hasta terminar de comparar todos los pares de números, y eso solo es en la primera pasada, porque tiene que dar varias “pasadas”, para ordenar los números de un arreglo.

Código:

```
def burbuja(A):
```

```
    cnt_burbuja=0
```

```
    for i in range (0, len(A)-1):
```

```
        for j in range(0, len(A)-1):
```

```
            cnt_burbuja+=1
```

```
            if(A[j+1]<A[j]):
```

```
                aux = A[j+1]
```

```
                A[j+1]=A[j]
```

```
                A[j]=aux
```

```
    return cnt_burbuja
```

Insertion sort:

El tipo de algoritmo “inserción” (su traducción en español), también es uno de los más ineficientes, pero este a diferencia de Bubble, no empieza a comparar con el primer número (elemento del arreglo), más bien compara a partir del segundo número, entonces si quisiéramos ordenar de forma ascendente, “pregunta” si el segundo es menor que el anterior, y si es lo es, lo desplaza a donde debe de ir , después toma el tercer elemento y los compara con el

primer y segundo elemento, en pocas palabras va “preguntando” si es menor (o mayor) según sea el caso, para luego ir desplazando números hasta llegar a él orden deseado. Pero mientras esta en el proceso, “elimina “ los valores comparados que no cumplían la condición(solo temporalmente), y luego vuelve a agregarlos en las posición correcta, de ahí el nombre de inserción, su complejidad al igual que Bubble es de n^2 .

Código:

```
def orden_por_inserccion(array):
```

```
    cnt_insertion=0
    for indice in range(1, len(array)):
        valor = array[indice]
        i=indice-1
        while i>=0:
            cnt_insertion+=1
            if valor<array[i]:
                array[i+1]=array[i]
                array[i]=valor
                i-=1
            else:
                break
    return cnt_insertion
```

Selection sort:

El tipo de algoritmo “selección” (su traducción en español), este en comparación con inserción y Bubble, realiza menos operaciones, ya que selection busca el valor más grande desde el primera pasada y después de completar el pase lo pone en la posición correcta, en la segunda pasada, toma el segundo valor más grande y lo ubica en las posición deseada, de esta forma continua hasta que tiene todos los elementos del arreglo ordenados, es más sencillo de visualizarlo que un Bubble o un inserción, se puede ver que el orden de selection hace el mismo número de comparaciones que el tipo de bubble , por lo tanto su complejidad también es n^2 . Pero debido a la reducción en el número de intercambio, selection podría decirse que “tarda” menos que un bubble.

Código:

```

def selection(arr):

    cnt_selection=0

    for i in range(0,len(arr)-1):

        val= i

        for j in range(i+1, len(arr)):

            cnt_selection+=1

            if arr[j]< arr[val]:

                val=j

        if val != i:

            aux=arr[i]

            arr[i]=arr[val]

            arr[val]=aux

    return cnt_selection

```

Quick sort:

El tipo de algoritmo “clasificación rápida” (su traducción en español), es la más eficaz que vimos en clase, y no se parece a ningunos de los otros 3 algoritmos de ordenación mencionados anteriormente, este toma un número(elemento del arreglo), este es cualesquiera de los elementos(pero lo ideal sería justo el de la mitad), y ese va ser nuestro “pivote”, a partir de ese “pivote”, separa el arreglo en dos partes, una vez que ya está partido el arreglo, pone de un lado los menores(o mayores) según sea el caso y del otro lado los mayores(o menores) al pivote, de esta forma ira ordenando el arreglo hasta llegar a la forma deseada, su complejidad es de $n\log_2(n)$ la cual es menor que n^2 , y es por eso que esta es mucho más eficiente que las 3 anteriores .

Código:

```

def quicksort(arr):

    global cnt_quick

    if len(arr)<=1:

        return arr

    p=arr.pop(0)

    menores,mayores=[],[]

    for e in arr:

```

```

cnt_quick+=1
if e<=p:
    menores.append(e)
else:
    mayores.append(e)

```

```

return cnt_quick

```

Después de definir las cuatro funciones creamos un arreglo y lo hacemos cuatro copias de dicho arreglo, y realizamos pruebas, esto es:

```

import random
def rndar(longitud):
    arr=[]
    for r in range(longitud):
        arr.append(random.randint(0,longitud))
    return arr

```

```

import copy
L=4
print("L", "B", "S", "I", "")
while L<100:
    for replica in range(30):

```

```

        arr = rndar(L)
        a, b, c, d = copy.deepcopy(arr), copy.deepcopy(arr),
copy.deepcopy(arr), copy.deepcopy(arr)
        bc = burbuja(a)
        ic = orden_por_inserccion(b)
        sc = selection(c)

```

```

cnt_quick=0

qc = quicksort(d)

```

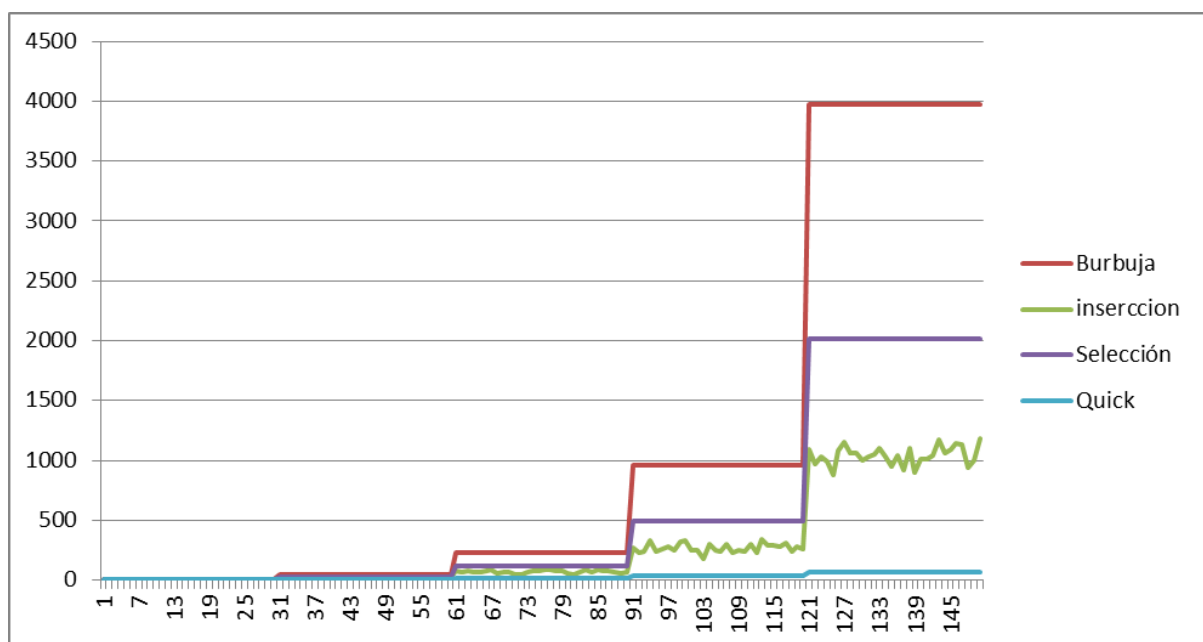
```

print(L, bc, ic, sc, qc)

```

$L^*=2$

Al finalizar, pasamos los resultados de las pruebas a un gráfica, como se muestra a continuación:



Algoritmo	Fortalezas	Debilidades
Bubble	No varia la cantidad de operaciones (es más fácil de predecir)	Realiza muchas operaciones incluso en arreglos pequeños
Insertion	En cuanto a las fortalezas dependerá mucho el arreglo. Asi que no es sencillo decir una fortaleza de insertion.	Varia demasiado la cantidad de operaciones(no es sencillo calcular o aproximar el número de operaciones que realizará)
Selection	No varia la cantidad de operaciones (es más fácil de predecir)	Trabaja muy parecido a bubble.
Quick	No varía tanto el número	Es el que realiza menos

	de operaciones que realiza como insertion pero no es predecible, porque depende mucho que pivote tomes para ordenarlo.	operaciones.
--	--	--------------

Conslusiones:

Después de hacer de hacer las pruebas y observar los resultados, pues claro está que quicksort es el más “rápido”, es decir que hace menos operaciones, en la gráfica se muestra muy por debajo de los otros 3 algoritmos de ordenamiento, realmente está muy “pegado” a la línea horizontal, es por eso que es el mas eficaz, otra cosa muy destacable es que insertionsort a diferencia de los otros 3 algoritmos es muy alternante en cuanto a las operaciones que realiza, no es para nada predecible como bubblesort y bueno como ya había mencionado antes en la gráfica podemos notar que desafortunadamente realiza más operaciones que los otros y pues selectionsort es muy parecido a bubblesort en cuanto a las operaciones que realiza. Aunque realiza menos operaciones que bubblesort, basada en los resultados obtenidos podría decirse que selectionsort realiza la mitad de operaciones que bubblesort.