Algoritmos de busqueda y ordenamiento

# Que es un algoritmo?

Es un conjunto de instrucciones precisas y finitas que al ejecutarse permiten resolver un problema. No deben tener ambigüedad y debe ser eficaz. No necesariamente eficiente, es solo un valor ideal.

Es finito porque termina después de ejecutar todos los pasos.

Debe ser claro…

Efectividad, que cumpla con lo que tiene que hacer.

Eficiencia, hágalo de la mejor manera.

Posee un input o un output

Se pueden estudiar los algoritmos en términos de eficiencia o rendimiento

El compilador, el OS, el tamaño del input

# Complejidad

## Tiempo (CPU)

Es la evaluación más común

Se intenta determinar cuanto tiempo tarda el algoritmo en completarse. Se cuenta en términos de ciclo de reloj. Cuantas instrucciones elementales ejecuta este algoritmo

(cuantos ciclos se necesitan para completar, cuantas instrucciones elementales por ciclo)

Hay instrucciones (operaciones elementales) que en el cual el CPU tarta T. T depende de la arquitectura.

## Operaciones elementales

Suma,resta…

Operaciones lógicas (AND, OR …)

Operaciones de corrimiento

Asignaciones de variables

Accesos a estructuras indexadas

Comparaciones 0 > < <= …

Todas estas son realizadas en un tiempo T.

## Espacio (RAM, disco duro)

# Análisis de algoritmos

Formas estándar:

## Empírico (Bench mark)

Comparo dos algoritmos en un mismo ambiente.

El mas simple, pero el que aporta menos.

## Simulacional

Definir casos de prueba. Y a partir de estos casos se tabula y se hace un análisis para estos algoritmos. Ver cómo se comporta según la data que le entrego.

Es mas exacto q el empírico, pero no puedo definir casos de prueba para todo el flujo del sistema.

## Analítico

Un modelo matemático que engloba todos los posibles casos para el sistema.

# Algoritmos de ordenamiento

Mantener una estructura ordenada permite una mayor facilidad de búsqueda.

## Condición

Bajo que condición voy a comparar los dos elementos. Bajo que criterio.

## Tamaño del

## Selección Sort

Ejemplo de las hojas. Pero hecho desde una misma lista.

Buscar el menor y comenzar del siguiente índice.

(para comparar algoritmos de ordenamiento se puede hacer en términos de cuantas comparaciones hago, pero también puede ser por T)

Comparaciones: n(n-1)/2

# Big O

Notación asintótica, es el grado de complejidad de un algoritmo.

Toma el n más complejo.

## Bubble sort

Tienen la misma complejidad de la selección sort (n\*\*2).

Hace la misma cantidad de comparaciones, que el selection

Hace swap inmediatos

Si no hago swaps en una iteración puedo levantar un “flag” para detener el algoritmo (no es estándar)

## Insertion Sort

Es mas eficiente con estructuras preordenadas.

Su complejidad también es O n\*\*2

(estos últimos 3 tienen una parte ordenada)

### Ejercicio comparativo

18-11-5-27-49-34-8-32-3-29

Selection

S 8

C 45

Bubble

S 22

C 45

Insertion

S 22

C 28

## Shell sort

Donald L. Shell. Basado con el insertion sort

Bueno para arreglos de tamaño medio

h = n/2 y comparo el primero con la posición h (eso solo una forma de elegir h)

y muevo la línea y comparo en distacia h.

cuando termino uso una nueva distacia h = h/2

cuando h es 1 si hago swap comparo hacia atrás (guardando de donde hice swap)

no tiene una complejidad estándar, ya que el puede ser calculado de varias formas.

## Merge Sort

Divido la lista a la mitad y aplico el merge sort (es decir vuelvo a dividir) hasta llegar a descomponerlo en unidades básicas (un solo numero). Y comparo con la sigente pareja.

nLog2N

## Quick Sort

## Radix Sort