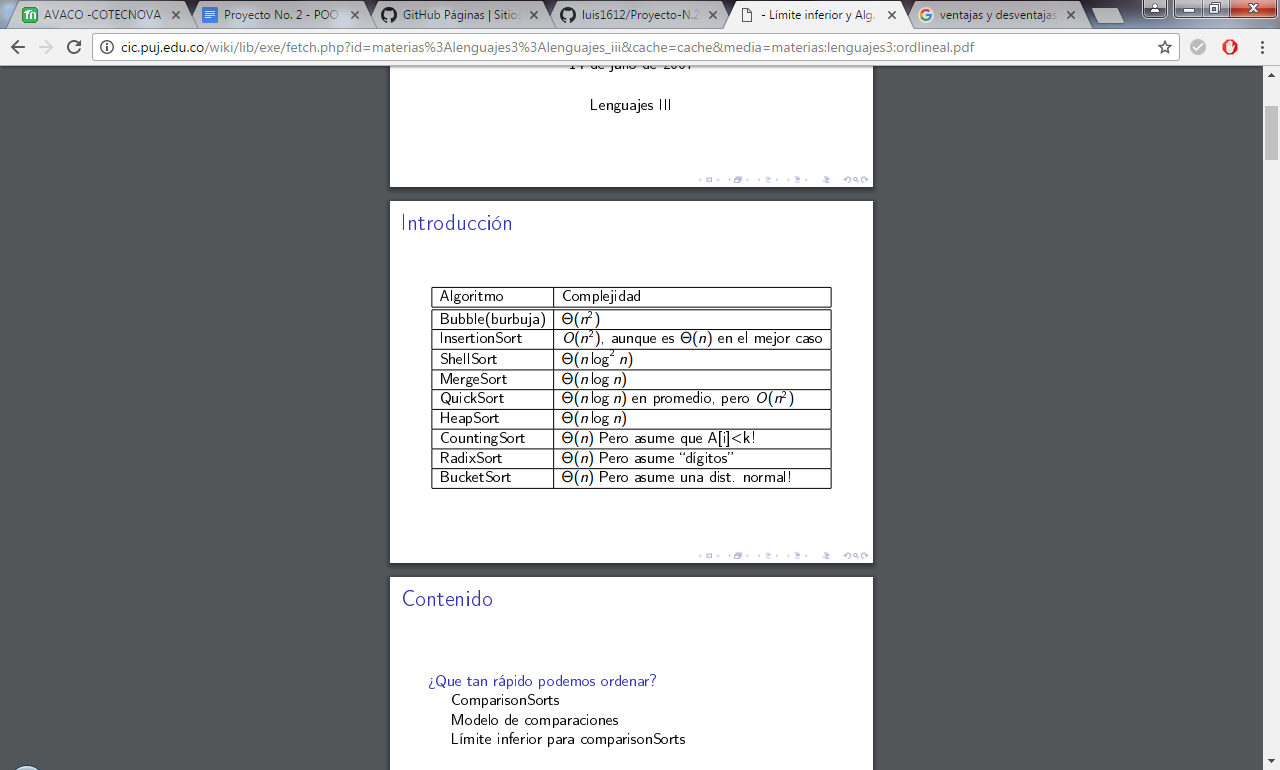
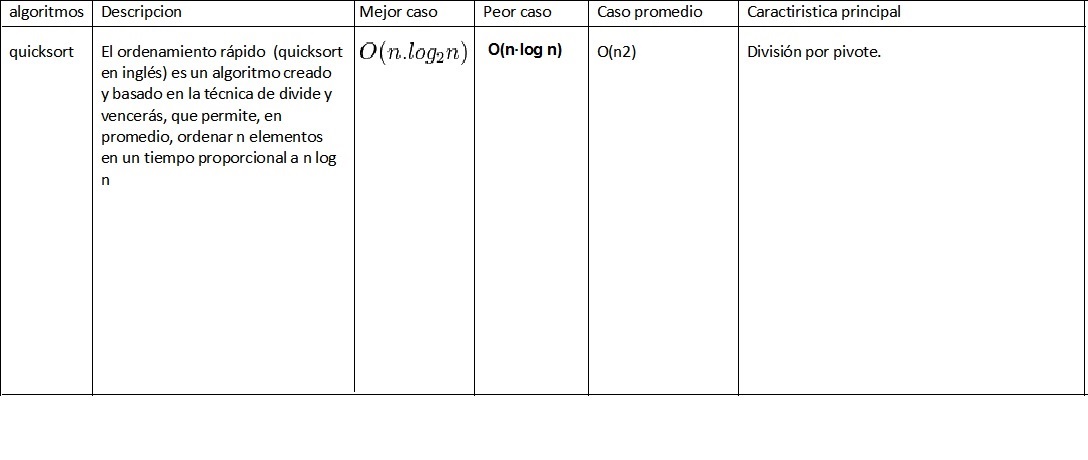
**Ventajas y Desventajas**

**De los métodos de ordenamiento (JAVA).**



En esta sección daremos a conocer como otros tipos de algoritmos los cual pueden ser de gran ayuda al momento de ordenar un algoritmo en las siguientes tablas se mostraran estos su mejor, peor o promedio caso y sus características:

**Quicksort:**



Ventajas:

1) muy rápido

2) no requiere memoria adicional.

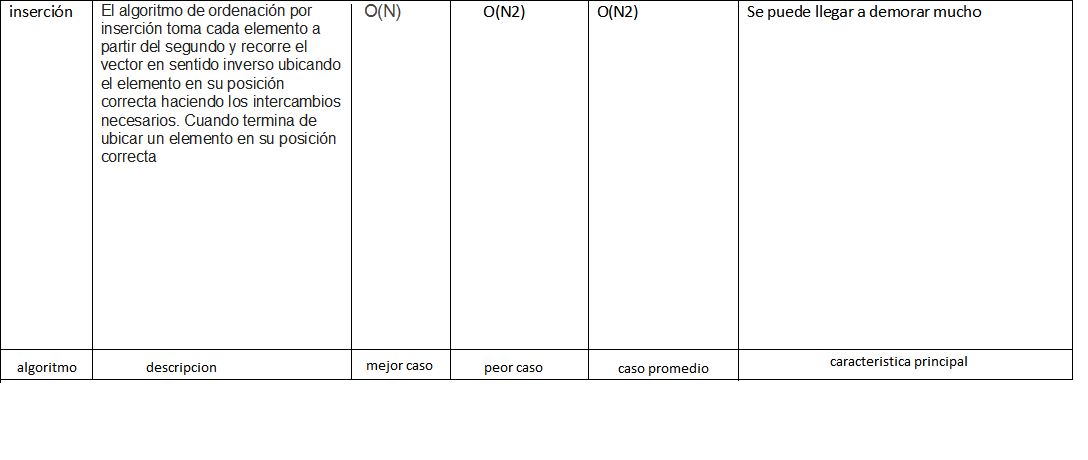
Desventajas:

1) la implementación es un poco más complicada

2) en la recursividad (este utiliza muchos recursos)

3) existe mucha diferencia entre el peor y el mejor caso.

**Inserción:**



Ventajas:

1) fácil de implementar

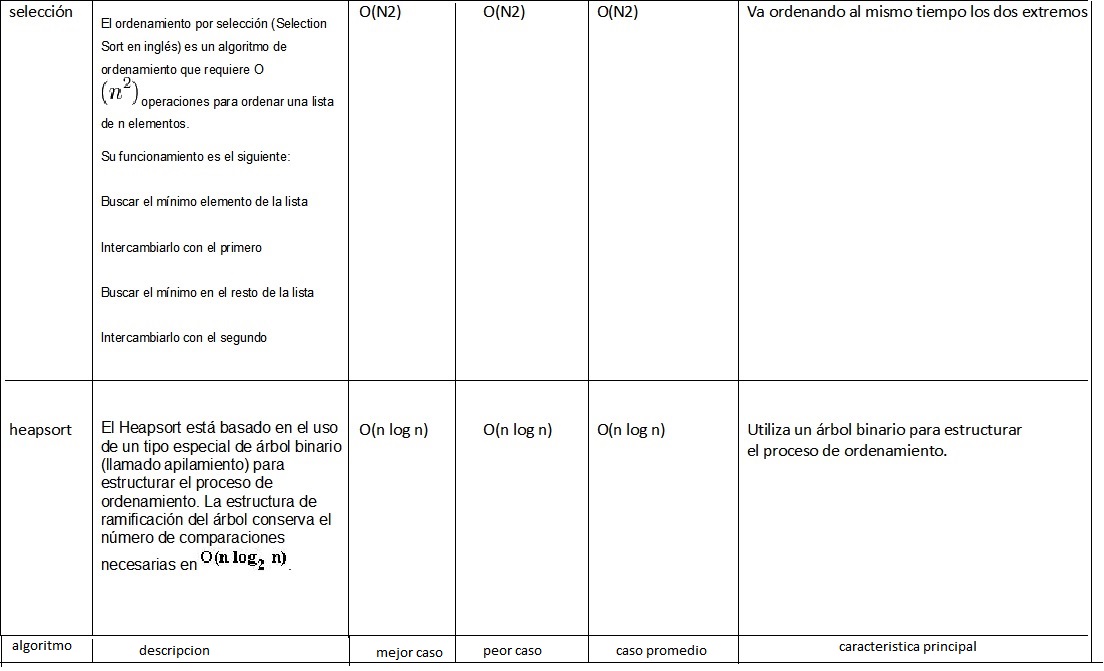
2) requerimiento mínimo de memoria

Desventaja:

1) lento

2) realiza numerosas comparaciones

**Heapsort:**



Ventajas y desventajas del algoritmo de heapsort:

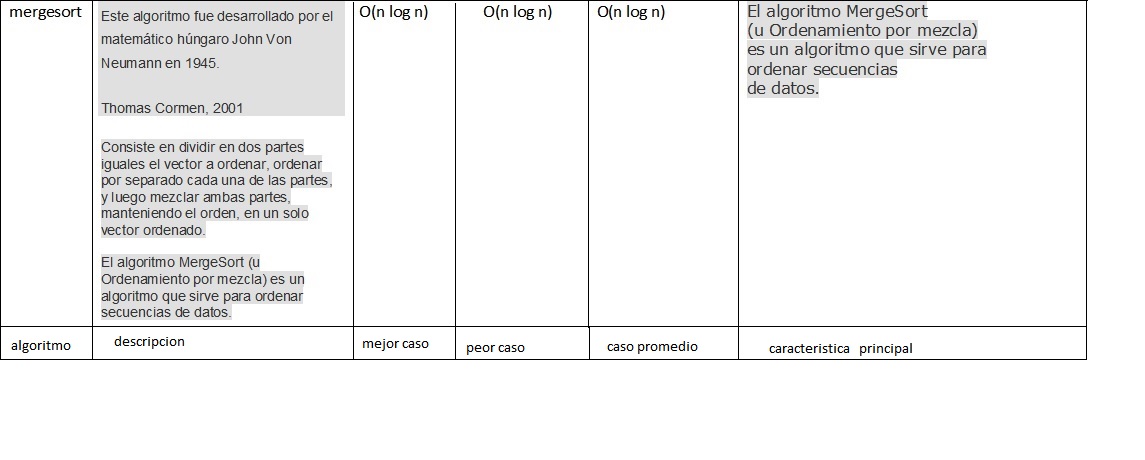
Ventajas

Su desempeño es en promedio tan bueno como el Quicksort y se comporta mejor que este último en los peores casos.

Desventajas

Aunque el Heapsort tiene un mejor desempeño general que cualquier otro método presentado de clasificación interna, es bastante complejo de programar.

**mergesort:**



Ventajas:

Método de ordenamiento estable mientras la función de mezcla sea implementada correctamente.

Muy estable cuando la cantidad de registros a acomodar es de índice bajo, en caso contrario gasta el doble del espacio que ocupan inicialmente los datos.

Efectivo para conjunto de datos a los que se puede acceder secuencialmente (arreglos, vectores, etc.)

Desventajas:

Principal desventaja: está definido recursivamente. Si se deseara implementarla no recursivamente se tendría que emplear una pila y se requeriría un espacio adicional de memoria para almacenarla.

Pd: en mergesort, heapsort y Shell sort donde los casos sale depende estos depende del tamaño del vector, etc.

**Radix Sort:**

Ventajas y Desventajas

Ventajas

Radix Sort es estable, preservando la orden de elementos iguales.

Radix Sort funciona en un tiempo lineal, en comparación de varios otros métodos de ordenamiento.

El tiempo de ordenar cada elemento es constante, ya que no se hacen comparaciones entre elementos.

Radix Sort es particularmente eficiente cuando se tratan con grandes grupos de números cortos.

Desventajas

Radix Sort no funciona tan bien cuando los números son muy largos, ya que el total de tiempo es proporcional a la longitud del número más grande y al número de elementos a ordenar.