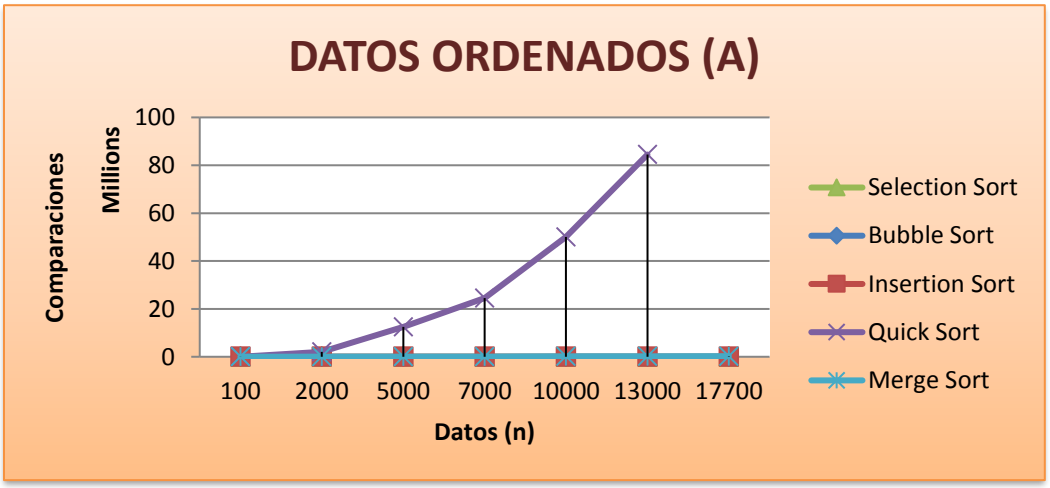
TAREA 1

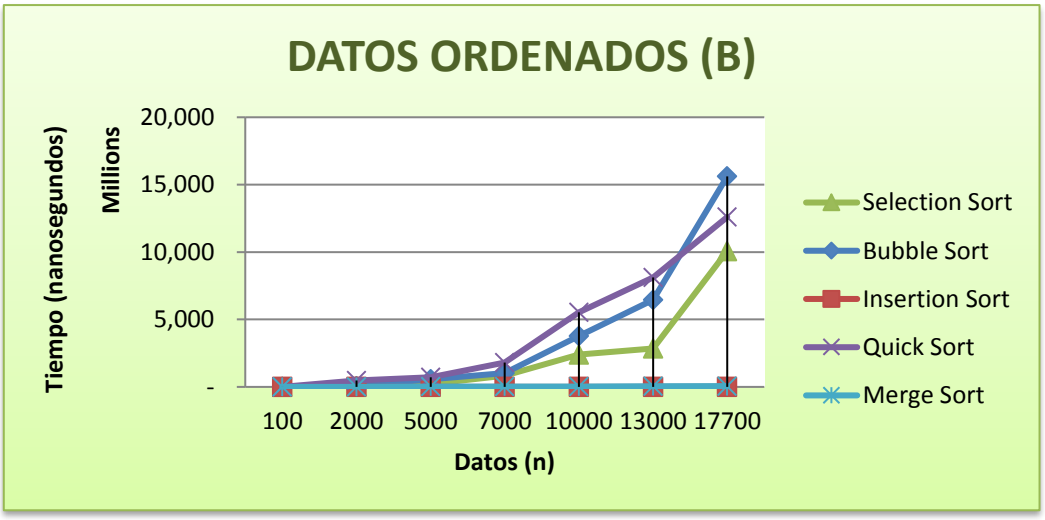
Adelina Trejo Espinoza (168135)

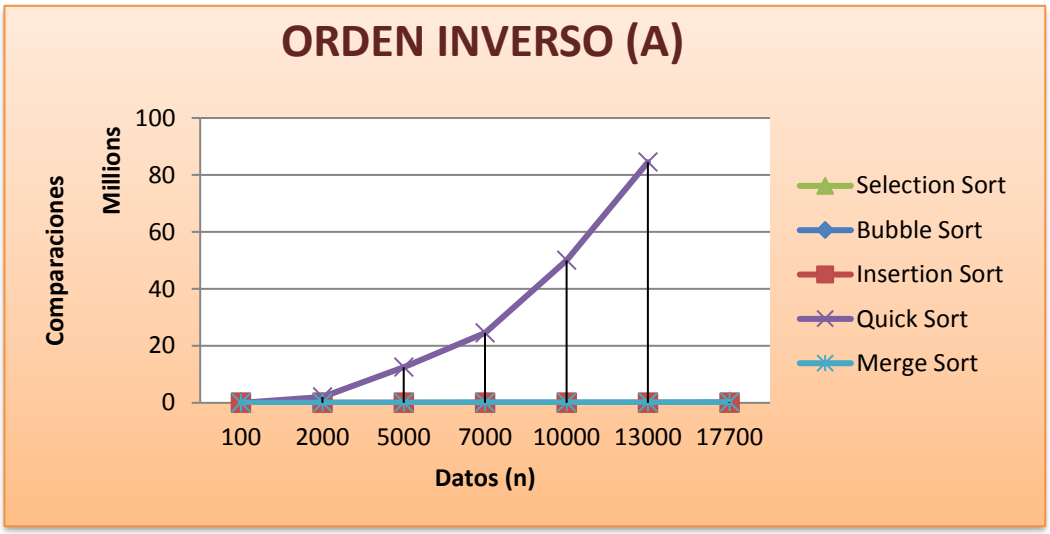
Instrucciones

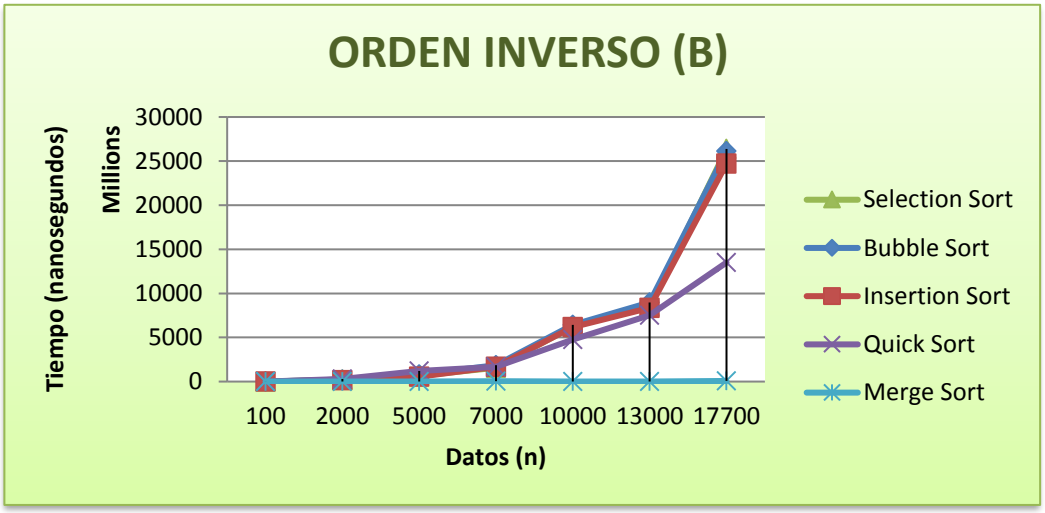
Implementar cada uno de los algoritmos de ordenamiento vistos en clase (implemente la versión de MergeSort vista en clase). La implementación debe de ser genérica, independiente del tipo de datos a ordenar Adicionalmente, por conveniencia, escriba todos los algoritmos como miembros de una sola clase. Para cada uno de los siguientes incisos entregar dos figuras (A y B), una para el número de comparaciones y otra para el tiempo de ejecución (medido en una escala apropiada). Cada figura debe de tener el resultado de todos los algoritmos para poder comparar su desempeño. Entregue un documento en .pdf con las gráficas y su interpretación, sus observaciones acerca del desempeño.

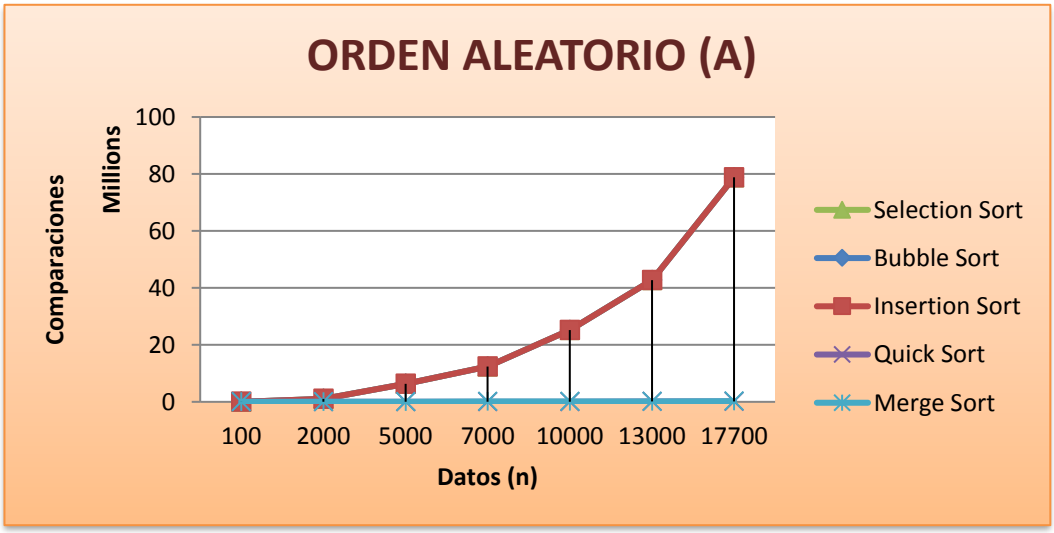
Resultados (Gráficas)

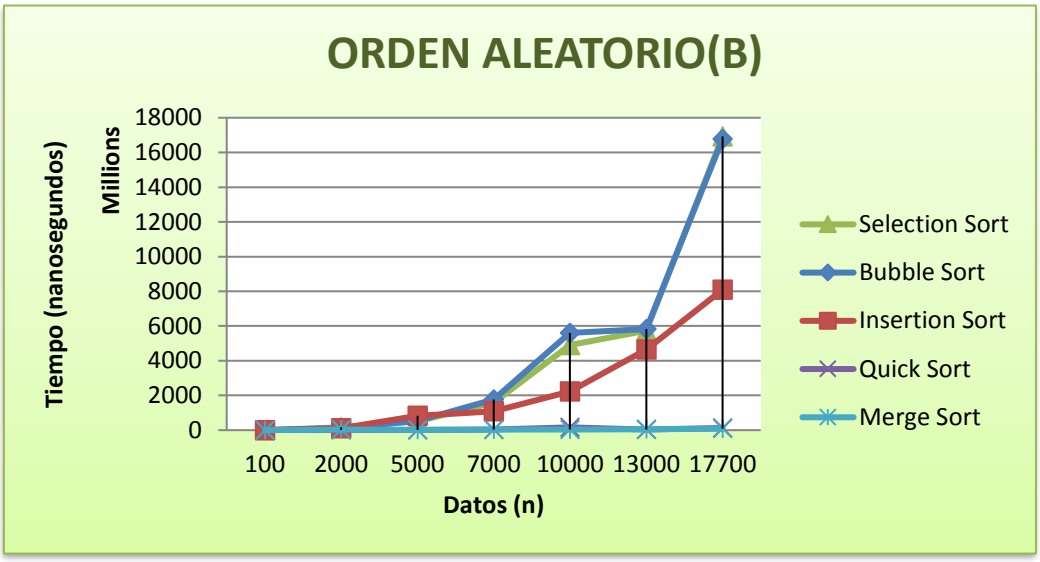





ORDEN INVERSO (A)
Comparaciones
Millions
100
80
60
40
20
0
100
2000
5000
7000
10000
13000
17700
Datos (n)
Selection Sort
Bubble Sort
Insertion Sort
Quick Sort
Merge Sort


ORDEN INVERSO (B)
Tiempo (nanosegundos)
Millions
30000
25000
20000
15000
10000
5000
0
100
2000
5000
7000
10000
13000
17700
Datos (n)
Selection Sort
Bubble Sort
Insertion Sort
Quick Sort
Merge Sort


ORDEN ALEATORIO (A)
Comparaciones
Millions
100
80
60
40
20
0
100
2000
5000
7000
10000
13000
17700
Datos (n)
Selection Sort
Bubble Sort
Insertion Sort
Quick Sort
Merge Sort



Observaciones y Conclusiones

Cuando los datos se encuentran ordenados Quick Sort, es el algoritmo que se mantiene menos constante en cuanto al número de comparaciones que requiere hacer en función al aumento de la cantidad de datos. Merge Sort es más constante, y el resto lo son totalmente. En cuanto al tiempo todos aumentan progresivamente el tiempo conforme al aumento en la cantidad de datos, sin embargo Quick Sort, a la par de Bubble Sort, y seguido por Selecion Sort son más susceptibles a este aumento. Por lo tanto, Merge Sort e Insertion Sort parecen ser más eficientes en este caso determinado.

Bubble, Insertion, Selection Sort tienen un número de comparaciones muy semejante cuando los datos se encuentran en orden inverso, mostrándose constantes en la gráfica. Sin embargo no se puede decir lo mismo en cuanto al tiempo de ejecución, ya que en este caso determinado Merge Sort se muestra más eficiente en cuanto a la susceptibilidad de cambio tanto para tiempo, como para comparaciones.

Para el caso en que los datos se iniciaban en orden aleatorio, Quick Sort y Merge Sort mostraron más constancia también en este caso. El resto de los algoritmos se mostró muy susceptible de cambio tanto en tiempo como en número de comparaciones, conforme al número de datos aumentaba.