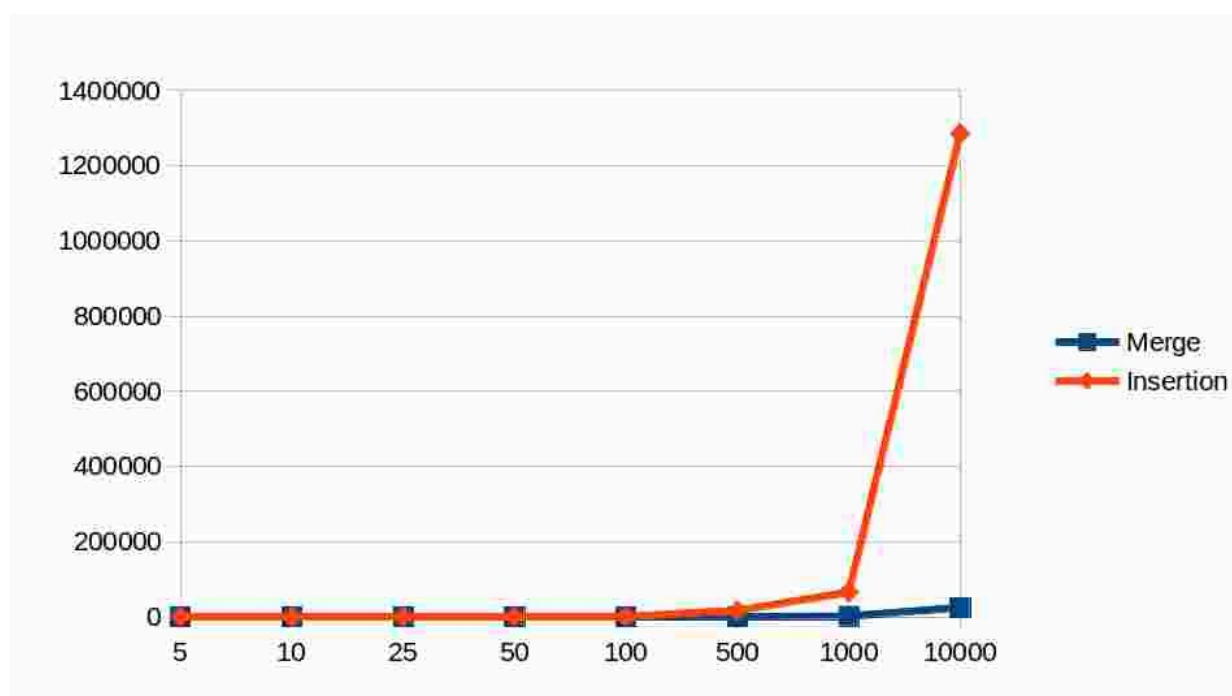


RELAZIONE DEL LABORATORIO DI ALGORITMI

Esercizio n°1: algoritmi di ordinamento



Costamagna Alberto e Gianotti Damiano

01/05/2018

Matr: 833771

Matr: 835271

INTRODUZIONE

Abbiamo implementato una libreria con i due algoritmi di ordinamento, insertion e merge sort su un generico tipo T specificando diversi criteri secondo cui ordinare i dati

MATERIALI

1. package sort
2. package sortusagejava
3. hamcrest-core.jar & junit.jar

PROCEDURA

COMPILAZIONE

----PER COMPILARE LE CLASSI PER LA STRUTTURA DATI Sort NEL PACKAGE sort---

- 1) posizionarsi in .../Sort/src
- 2) `javac -d ../classes sort/Sort.java`

---PER COMPILARE IL PACKAGE sortusagejava---

- 1) posizionarsi in .../Sort/src
- 2) `javac -d ../classes sortusagejava/SortUsageJava.java`

---PER COMPILARE LE CLASSI PER GLI UNIT TEST NEL PACKAGE sort---

- 1) posizionarsi in .../Sort/src
- 2) `javac -d ../classes -cp '.:../junit-4.12.jar;../hamcrest-core-1.3.jar' sort/*.java`

ESECUZIONE

---PER ESEGUIRE sortusagejava/SortUsageJava---

- 1) posizionarsi in .../Sort/classes
- 2) `java sortusagejava/SortUsageJava "../integers.csv"`

---PER ESEGUIRE sort/SortJava_TestsRunner---

- 1) posizionarsi in .../Sort/classes
- 2) `java -cp '.:../junit-4.12.jar;../hamcrest-core-1.3.jar' sort/SortJava_TestsRunner`

DATA

Algoritmo utilizzato/i	File utilizzato/i	Tempo medio di esecuzione/stampa
Insertion sort	"integers.csv"	circa ∞ / ∞
Merge Sort	"integers.csv"	8 ~ 11 sec / 12 ~ 13 minuti
Merge Sort + Binary Search	"integers.csv", "sums.txt"	57 sec ~ 60 sec

TODO

RISULTATI

Dopo eseguito i vari test e implementazioni richieste, non possiamo che constatare come la complessità degli algoritmi impatti moltissimo le prestazioni.

In particolare :

1. la complessità quadratica (n^2) di Insertion Sort fa risultare poco efficiente una qualsiasi implementazioni con $N > 1000$ (vedi grafico).
2. Merge Sort risulta molto più versatile e si comporta egregiamente sia con N piccoli che con N grandi.