Algoritmi e Strutture Dati

Progetto di Laboratorio Esercizio 1
Anno Accademico: 2022/2023

AbdelHamid Badrane[matricola: 902154]
abdelhamid.badrane@edu.unito.it
Marika Torrente[matricola: 916315]
marika.torrente@edu.unito.it

Sommario

1.Esercizio 1
1.1 Introduzione
1.2 Guida all'utilizzo
1.3 Tempi di esecuzione
1.3.1 Confronto tra diversi pivot
1.4 Conclusioni

1.1 Introduzione

In questo esercizio andremo a implementare una libreria che offre un algoritmo di ordinamento Merge-BinaryInsertion Sort, si tratta di un algoritmo ibrido che combina *Merge Sort* e *BinaryInsertion Sort.*
Sfruttiamo il valore di k in modo da ottimizzare l'ordinamento sfruttando il potenziale di entrambi gli algoritmi, `k=0` implica che *Merge-BinaryInsertion Sort* si comporta esattamente come il *Merge Sort* classico, mentre `k>>0` aumenta l'utilizzo del *BinaryInsertion Sort*.
Con Binary-Insertion Sort ci riferiamo a una versione dell'algoritmo Insertion Sort in cui la posizione, all'interno della sezione ordinata del vettore, in cui inserire l'elemento corrente è determinata tramite ricerca binaria.

1.2 Guida all'utilizzo

Comandi make per l'utilizzo del programma:

-make all: compila e crea l'eseguibile per il main e gli unit tests
-make test: compila, crea ed esegue il programma per gli unit tests
-make main: compila, crea ed esegue il programma per il main
-make run: controlla la presenza di un argomento src, dst, k e field che indicano rispettivamente il file da ordinare, il file ordinato di uscita, il valore di k e in base a quale field vogliamo ordinare, se non è presente uno dei precedenti segnala la mancanza e in caso contrario viene avviato il main

1.2 Tempi di esecuzione

Per l'ordinamento dei 20000000 di record il Merge-BinaryInsertion riporta fallimento in caso di k tendente a 20000000 in quanto ci mette tempo maggiore di 10 minuti perchè viene utilizzato gran parte il BinaryInsertion sort rispetto al merge sort,
||||||||||||tuttavia per numero di record ridotti si presta ad essere ottimo.
Tabella con i tempi di esecuzione del Merge-BinaryInsertion

valori di k	String	Int	Double
0	~24,92 s	~18,87 s	~19,44 s
50	~22,51 s	~18,68 s	~18,53 s
1000	~ 25,78 s	~21,10 s	~20,27 s
5000	~24,86 s	~23,04 s	~21,33 s
50k	~46,70 s	~44,57 s	~45,46 s
500k	~345 s	~345 s	~345 s

1.3 Conclusioni

Come abbiamo potuto notare dai diversi test effettuati la scelta del k ha un'importanza decisiva nell'esecuzione del Merge-BinaryInsertion sort.
Per sfruttare al meglio l'algoritmo è quindi preferibile utilizzare valori di k compresi tra 50 e 1000.
E infatti conviene utilizzare il BinaryInsertion sort al posto del Merge sort per l'ordinamento di array di dimensione particolarmente ridotte, mentre per array grandi è più conveniente utilizzare il Merge sort con complessità $O(n\ log\ n)$ rispetto al BinaryInsertion sort con $O(n^2)$