Relazione Progetto 1

Ingegneria degli Algoritmi 2018/2019

Matteo Lungo - 0253524

10 Dicembre 2018

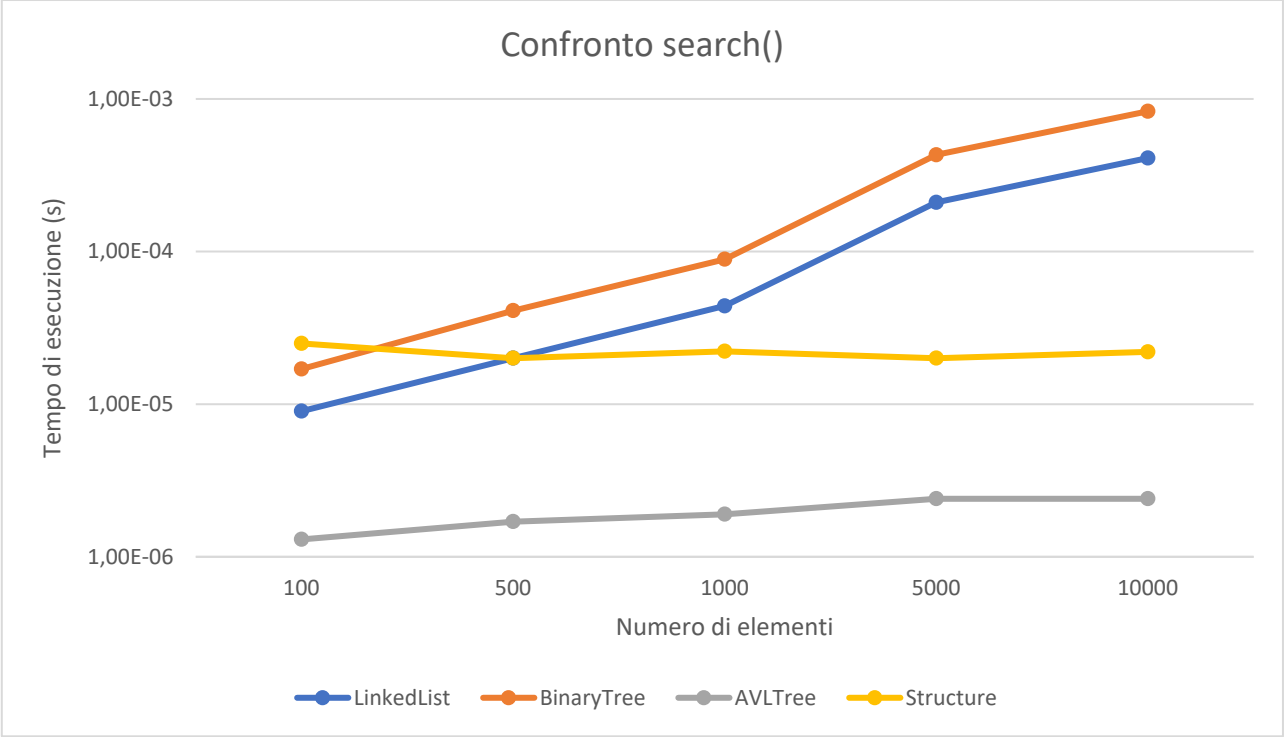
Scelte Implementative

Per la realizzazione del progetto, è stata definita una classe Structure, inizializzata secondo i parametri forniti dal testo e comprendente una lista array (contenente gli indici delle sottostrutture) e una lista structList contenente le sottostrutture stesse (come sottoliste). All'interno della classe sono stati implementati i metodi insert(), search() e delete() del Dictionary, un metodo index() che verifichi l'indice corrispondente alla struttura in cui andrà la chiave inserita e un metodo control() che verifichi se necessario trasformare la struttura da lista concatenata ad albero AVL e viceversa. A tale scopo sono stati definiti ulteriori due metodi: listToAVL che, basandosi sul metodo printOrdered della LinkedList, copia gli elementi della lista concatenata in una lista temporanea per poi sostituirla con una struttura di tipo albero AVL e inserirli al suo interno e AVLToList che, al contrario, appoggiandosi su una pila come nel metodo print del binaryTree, copia gli elementi dell'albero all'interno di una lista temporanea e una volta sostituito l'albero con una lista concatenata li inserisce al suo interno.

Risultati sperimentali e commenti

search()

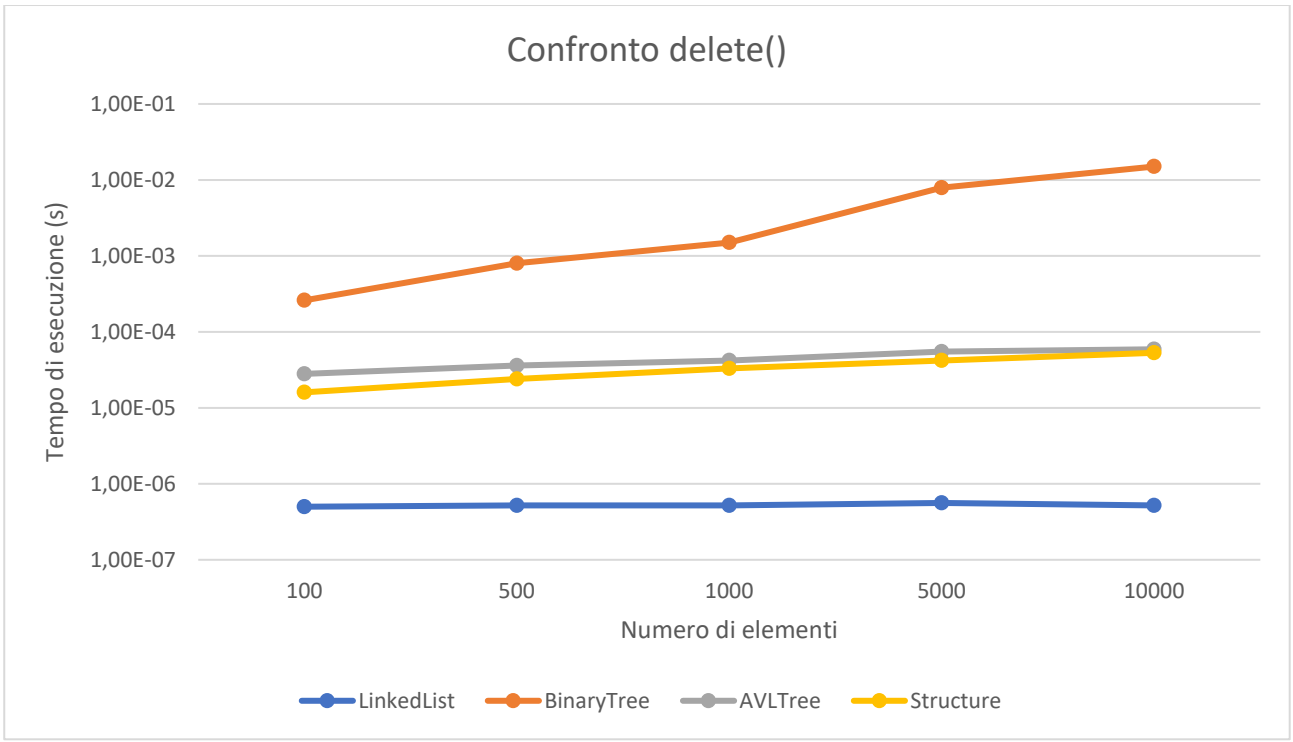
Elementi n	LinkedList	BinaryTree	AVLTree	Structure
100	9,00E-06	1,70E-05	1,30E-06	2,50E-05
500	2,00E-05	4,10E-05	1,70E-06	2,00E-05
1000	4,40E-05	8,90E-05	1,90E-06	2,22E-05
5000	2,10E-04	4,30E-04	2,40E-06	2,00E-05
10000	4,10E-04	8,30E-04	2,40E-06	2,20E-05



Confrontando i tempi di esecuzione medi del metodo search() , la struttura Structure ha mantenuto valori pressoché costanti all'aumentare del numero di elementi n presenti (sull'ordine dei 20 microsecondi). La struttura AVLTree si è dimostrata la più efficiente, mantenendo valori sull'ordine del microsecondo (anch'essa costante). Le strutture LinkedList e BinaryTree hanno dimostrato una crescenza logaritmica più elevata del tempo di esecuzione all'aumentare del valore di elementi presenti, partendo dai 10 microsecondi circa (100 elementi) e arrivando al millisecondo (10000 elementi).

delete()

Elementi n	LinkedList	BinaryTree	AVLTree	Structure
100	5,0E-07	2,6E-04	2,8E-05	1,6E-05
500	5,2E-07	8,0E-04	3,6E-05	2,4E-05
1000	5,6E-07	1,5E-03	4,2E-05	3,3E-05
5000	5,2E-07	7,9E-03	5,5E-05	4,2E-05
10000	5,2E-07	1,5E-02	5,9E-05	5,3E-05



Confrontando i tempi di esecuzione medi del metodo delete() , la struttura Structure ha mantenuto valori che si aggirano tra i 16 e i 53 microsecondi (100-10000 elementi), con prestazioni simili a quelle di AVLTree (28-59 millisecondi). La struttura LinkedList si è dimostrata la più efficiente, mantenendo valori sull'ordine del mezzo microsecondo. La struttura BinaryTree si è dimostrata la meno efficiente, con una curva di crescita che va dai 26 millisecondi al centesimo di secondo.