Relazione Progetto 1

Ingegneria degli Algoritmi 2018/2019

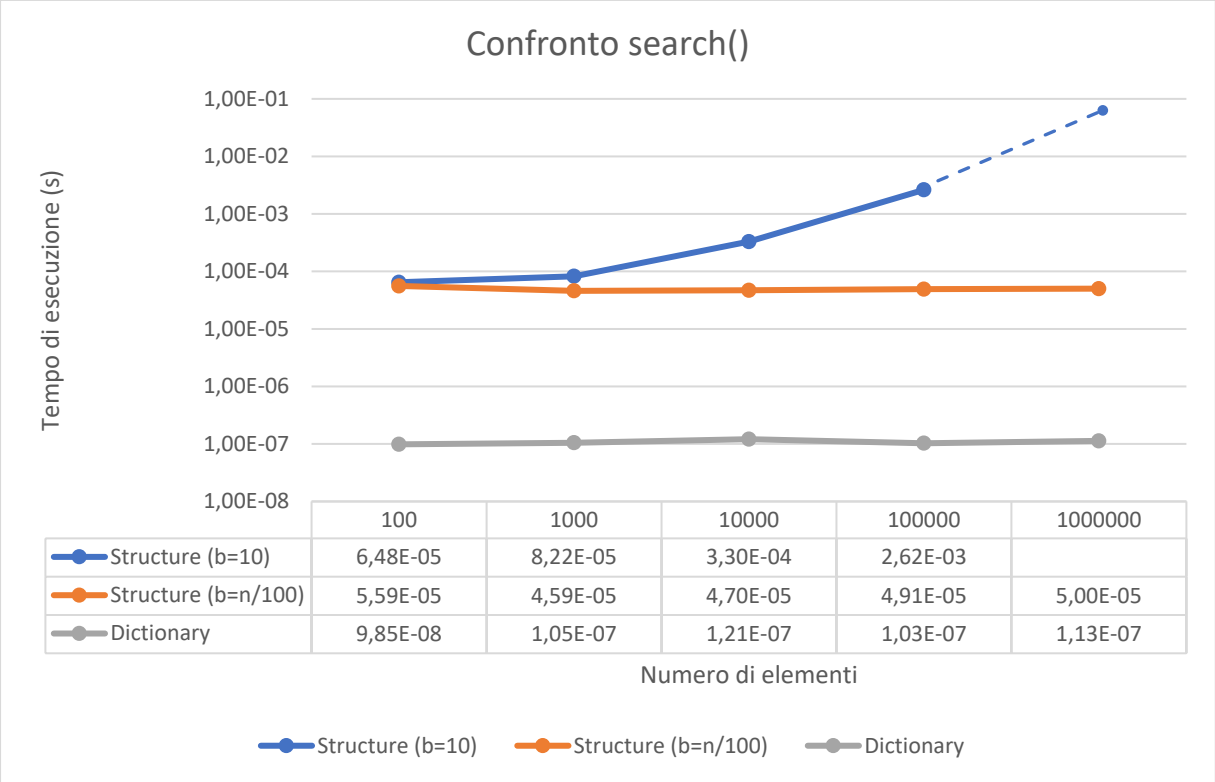
Matteo Lungo - 0253524

10 Dicembre 2018

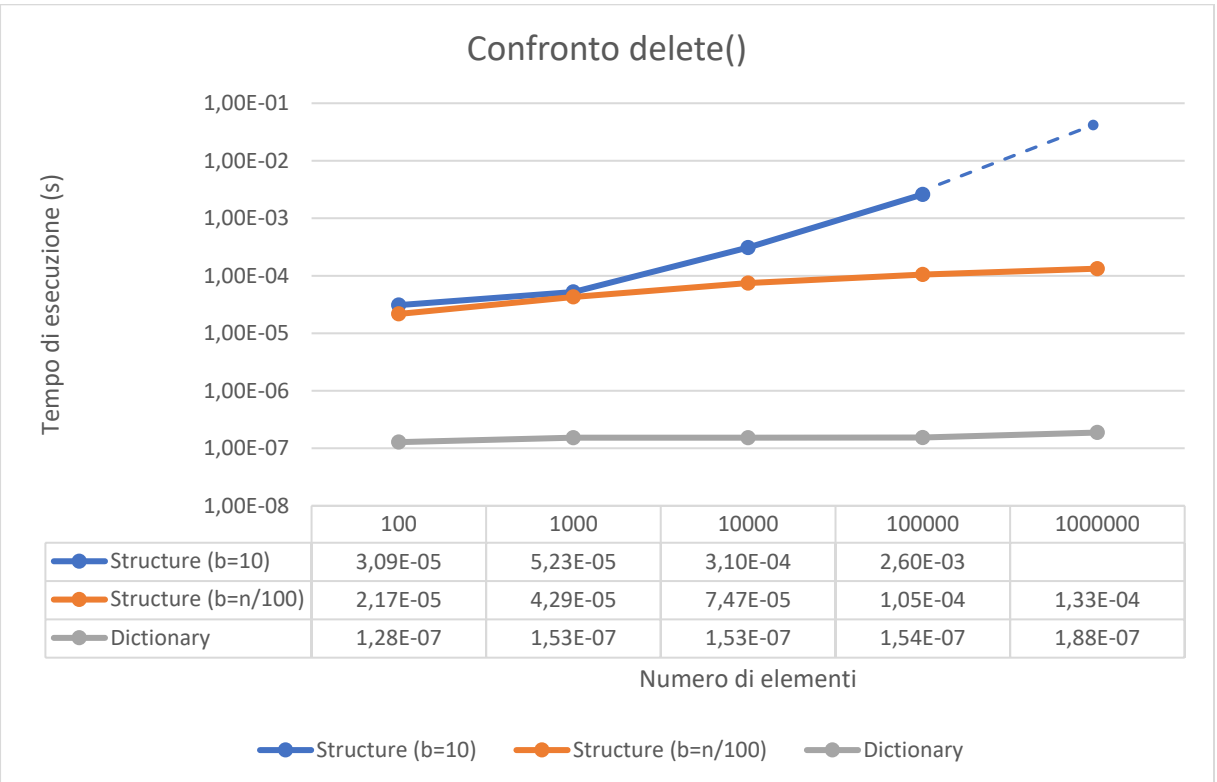
Scelte Implementative

Per la realizzazione del progetto, è stata definita una classe Structure, codificata nel file “Structure” e 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). Le strutture lista concatenata e albero AVL sono state implementate rispettivamente dai file “LinkedListDictionary” e “avlTree”. 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. Il codice dei file “LinkedListDictionary” e “avlTree” è stato modifcato con l’aggiunta di un metodo lenght() per il calcolo degli elementi presenti nelle strutture. Le stime dei tempi di esecuzioni sono state effettuate mediante l’utilizzo della libreria timeit.

Risultati sperimentali e commenti



	100	1000	10000	100000	1000000
Structure (b=10)	6,48E-05	8,22E-05	3,30E-04	2,62E-03	
Structure (b=n/100)	5,59E-05	4,59E-05	4,70E-05	4,91E-05	5,00E-05
Dictionary	9,85E-08	1,05E-07	1,21E-07	1,03E-07	1,13E-07



	100	1000	10000	100000	1000000
Structure (b=10)	3,09E-05	5,23E-05	3,10E-04	2,60E-03	
Structure (b=n/100)	2,17E-05	4,29E-05	7,47E-05	1,05E-04	1,33E-04
Dictionary	1,28E-07	1,53E-07	1,53E-07	1,54E-07	1,88E-07

Negli esperimenti effettuati la struttura Structure è stata inizializzata con i seguenti parametri:

- $min = 10$
- $max = n$
- $b = 10,\ \frac{n}{100},\ max - min,\ \frac{max-min}{2}$

I metodi insert(), search() e delete() utilizzati sono stati effettuati con valori casuali.

Per b grandi rispetto all'intervallo (es. b=10), l'analisi dei tempi di esecuzione dei metodi search() e delete() dimostra che, al crescere degli elementi n, la struttura Dictionary di Python riporta tempi pressoché costanti secondo una stima $O(1)$, mentre la struttura Structure cresce secondo una stima $log(n)$.

Per valori di b più bassi (es. b=n/100), la struttura riporta tempi di esecuzione che rimangono costanti anche all'aumentare del numero di elementi presenti. Il caso migliore è stato individuato in $b = \frac{max-min}{2}$, per l'applicazione dei metodi su elementi $n \mid min \leq n < max$ e in $b = max - min$ su elementi $n \mid n < \min \vee n \geq max$.

In un caso medio, effettuando i metodi con valori casuali (in un intevallo [x,y]), i migliori parametri di configurazione sono risultati i seguenti:

- $min = mediano(x, mediano(x, y))$
- $max = mediano(mediano(x, y), y)$
- $b = \frac{max-min}{2}$

In conclusione la struttura Dictionary di Python riporta tempi di esecuzione molto minori rispetto alla struttura Structure per entrambi i metodi search() e delete() anche nel suo caso migliore (tra i 10^2 e 10^3 secondi in meno per operazione).