Ingegneria degli Algoritmi (A.A. 2012/2013)

Progetto Software

(Curato da Simone Minasola e Federica Merola)

Indice:

1. Scelta del Progetto.

La funzione sotto presentata, è servita a determinare quale progetto svolgere. La stringa passata alla funzione è stata `simoneminasolafedericamerola` e il risultato ottenuto è stato `4`. Abbiamo quindi provveduto ad implementare gli algoritmi relativi alle strutture dati BST e AVL.

```
def scelta_progetto(nomi):
    numero=0
    for i in nomi:
        if i=='a': numero+=0
        elif i=='b': numero+=1
        elif i=='c': numero+=2
        elif i=='d': numero+=3
        elif i=='e': numero+=4
        elif i=='f': numero+=5
        elif i=='g': numero+=6
        elif i=='h': numero+=7
        elif i=='i': numero+=8
        elif i=='j': numero+=9
        elif i=='k': numero+=10
        elif i=='l': numero+=11
        elif i=='m': numero+=12
        elif i=='n': numero+=13
        elif i=='o': numero+=14
        elif i=='p': numero+=15
        elif i=='q': numero+=16
        elif i=='r': numero+=17
        elif i=='s': numero+=18
        elif i=='t': numero+=19
        elif i=='u': numero+=20
        elif i=='v': numero+=21
        elif i=='w': numero+=22
        elif i=='x': numero+=23
        elif i=='y': numero+=24
        elif i=='z': numero+=25
    return (numero%9)+1
```

2. Introduzione

2.1 Le strutture dati: BST e AVL.

Un `Dizionario` è un tipo di dato realizzato per implementare efficientemente (almeno) le tre operazioni elementari di `ricerca, inserimento e cancellazione` di elementi. Esistono diverse strutture dati per implementare un dizionario, tra le quali: BST e AVL.

Un BST (**B**inary **S**earch **T**ree) è un albero binario che soddisfa le seguenti proprietà:

1. è basato su *nodi* che contengono un *elemento* a cui è associato una *chiave*[1] presa da un dominio totalmente ordinato;
2. le chiavi del sottoalbero sinistro sono minori della chiave della radice del sottoalbero stesso;
3. le chiavi del sottoalbero destro sono maggiori della chiave della radice del sottoalbero stesso.

Il vantaggio principale nell'uso di un albero binario di ricerca è senz'altro la sua facile implementazione, che garantisce una discreta efficienza a fronte di un tempo di progettazione relativamente breve.

Le operazioni che caratterizzano tale struttura sono, tuttavia, teoricamente inefficienti; il problema principale sussiste nel fatto che l'albero risultante da inserimenti e cancellazioni di nodi, potrebbe essere totalmente sbilanciato in altezza[2], rendendo molto difficile le successive operazioni.

Operazioni Principali	Tempo di esecuzione
`search` (*chiave k*) ⟶ *elem*	O(h)
`insert` *(elem e, chiave k)*	O(h)
`delete` *(elem e)*	O(h)

Un AVL (**A**del'son – **V**el'skiĭ e **L**andis) è un albero binario di ricerca che si mantiene bilanciato in altezza a seguito di qualunque operazione effettuata. Ad ogni nodo, oltre che ad una chiave e una informazione, viene associato un *fattore di bilanciamento* calcolato come differenza assoluta tra l'altezza del suo sottoalbero sinistro e destro. Ogniqualvolta si procede all'inserimento o alla cancellazione di un nodo, questi fattori vengono aggiornati; nel momento in cui uno di questi risultasse maggiore o uguale a 2[3], si procede a ribilanciare l'albero secondo opportune rotazioni.

[1] Le chiavi possono essere valori di qualunque tipo, purché su di esse sia definita una relazione di ordine totale.
[2] Altezza (h): massima profondità a cui si trova una foglia.
[3] Se il fattore di bilanciamento di un nodo risultasse ≥ 2, significa che uno dei suoi sottoalberi è sbilanciato in altezza.

Il vantaggio principale nell'utilizzo di questa struttura dati consiste in un tempo di esecuzione delle operazioni proporzionale all'altezza dell'albero stesso, la quale rimane costantemente bilanciata e logaritmica nel numero di nodi[4], al contrario di quanto avviene per i BST.

Tuttavia il tempo di progettazione e realizzazione di tale struttura, seppur basata sugli alberi binari di ricerca, è lungo, e devono essere implementate molte più operazioni di quelle richieste dagli stessi BST. Le nuove operazioni da implementare sono le rotazioni che permettono il corretto bilanciamento dell'albero, ma che lo consentono in un tempo costante.

Operazioni Principali	Tempo di esecuzione
`search` (*chiave k*) ⟶ *elem*	O(log n)
`insert` *(elem e, chiave k)*	O(log n)
`delete` *(elem e)*	O(log n)

Esistono due tipi di rotazioni applicabili ad un nodo con fattore di bilanciamento maggiore o uguale a 2. La *Rotazione Base* (o rotazione semplice), consiste nel ruotare su un nodo perno verso destra o verso sinistra in modo da mantenere la proprietà ricerca dopo la rotazione. La *Rotazione Doppia* si ottiene componendo opportunamente due rotazioni base.

Rotazioni	Tempo di esecuzione
Sinistra – Sinistra	O(1)
Destra – Destra	O(1)
Sinistra – Destra	O(1)
Destra – Sinistra	O(1)

2.2 Valutazione a priori dell'andamento dei test.

Come richiesto, andremo ad effettuare dei test su entrambe le strutture dati, per determinarne i benefici. I test sono basati su sequenze random e sequenze già ordinate di inserimenti, cancellazioni e ricerche di stringhe (da un minimo di 5000 ad un massimo di 50000). Per ogni stringa si decide (con probabilità $0 \leq p1 < 1$) se inserirla, (con probabilità $0 \leq p2 < 1$) se cercarla o (con probabilità p3=1-p1-p2) se cancellarla.

In prima analisi, confrontando le caratteristiche delle due strutture dati, intuiamo una maggior efficienza computazionale da parte degli alberi AVL, derivata dal fatto che mantengono un' altezza logaritmica nel numero di nodi. L'effettuare continui ribilanciamenti sui nodi non comporta un eccessivo tempo addizionale alle normali operazioni[5], ma rendono estremamente efficienti tutte quelle successive. Per i BST, il non ricorrere alle rotazioni, non dovrebbe comportare un minor tempo di esecuzione: a fronte

[4] h = O(log n)
[5] Perché eseguibili in tempo costante O(1)

di un'esecuzione più veloce e lineare, il tempo totale di tutte le operazioni (random) risulterebbe maggiore di quello degli AVL, poiché i BST non prevedono il ribilanciamento dei nodi. Infatti per inserire, cancellare o cercare una chiave all'interno di un BST, si potrebbe percorrere tutto l'albero (che potrebbe risultare totalmente sbilanciato) dalla radice sino all'ultima foglia, eseguendo così le operazioni in un tempo lineare e non logaritmico come avverrebbe negli AVL. Deduciamo quindi che, per un numero uguale di operazioni, su un insieme ordinato o meno di stringhe, il tempo totale di esecuzione sarà sicuramente a favore degli alberi AVL in qualsiasi condizione.

NOTA: I dettagli delle implementazioni relative alle operazioni sono riportate nelle docstring e nei commenti dei codici relativi alle due strutture dati.

3. Test.

I test sono stati effettuati tramite il “TEST COMPLETO”, selezionabile dallo script principale del programma allegato alla relazione. Questo, comprende operazioni random tra inserimenti, cancellazioni e ricerche su un insieme di stringhe, ordinate e non, estrapolate da testi .txt presenti nella cartella `/dataset` del CD-ROM. Tali testi sono codificati in UTF - 8 per una perfetta individuazione dei caratteri speciali. Le operazioni vengono effettuate su un insieme che varia da un minimo di 5000 ad un massimo di 50000 stringhe (parole), a passi di 5000 (5000, 10000, 15000, … , 50000). Per evidenziare maggiormente la differenza tra le due strutture dati, è stato messo a disposizione anche un “TEST PARZIALE”, il quale permette di scegliere l’input, la sua dimensione e il numero di operazioni da effettuare su una singola struttura dati, lasciando libero arbitrio all’utente finale. Entrambi i test producono un file in output che racchiude tutti i tempi di esecuzione divisi per tipologia.

3.1 Piattaforme di Test.

I grafici che presenteremo, sono relativi all’esecuzione del programma su 4 differenti calcolatori:

1. "Desktop-1"
   CPU: [Quad Core] AMD Phenom II X4 965 B.E. @ 3.4 Ghz
   RAM: 4 Gb DDR3 @ 1600 Mhz
   O.S: Linux Ubuntu 12.04.2 | 64 bit

2. "Desktop-2"
   CPU: [Quad Core] AMD Phenom II X4 965 B.E. @ 3.4 Ghz
   RAM: 4 Gb DDR3 @ 1600 Mhz
   O.S: Microsoft Windows 8 | 64 bit

3. "iMac"
   CPU: [Quad Core] Intel core i5 @ 2,8 Ghz
   RAM: 4 Gb DDR3 @ 1333 Mhz
   O.S: Mac Os X Lion 10.7.5 | 64 bit

4. "Raspberry Pi"
   CPU: [Single Core] ARM11 (ARMv6) @ 1 Ghz
   RAM: 512 Mb SDRAM
   O.S: Linux Raspbian (Debian) | 32 bit

Python 2.7 comune a tutte le piattaforme

I tempi relativi ai computer 1, 2 e 3, sono uniti in un solo grafico, mentre per il 4, dati i notevoli tempi di esecuzione, si è deciso di riportarlo separatamente.

3.2 Analisi dei test relativi a BST e AVL su insiemi di elementi non ordinati.

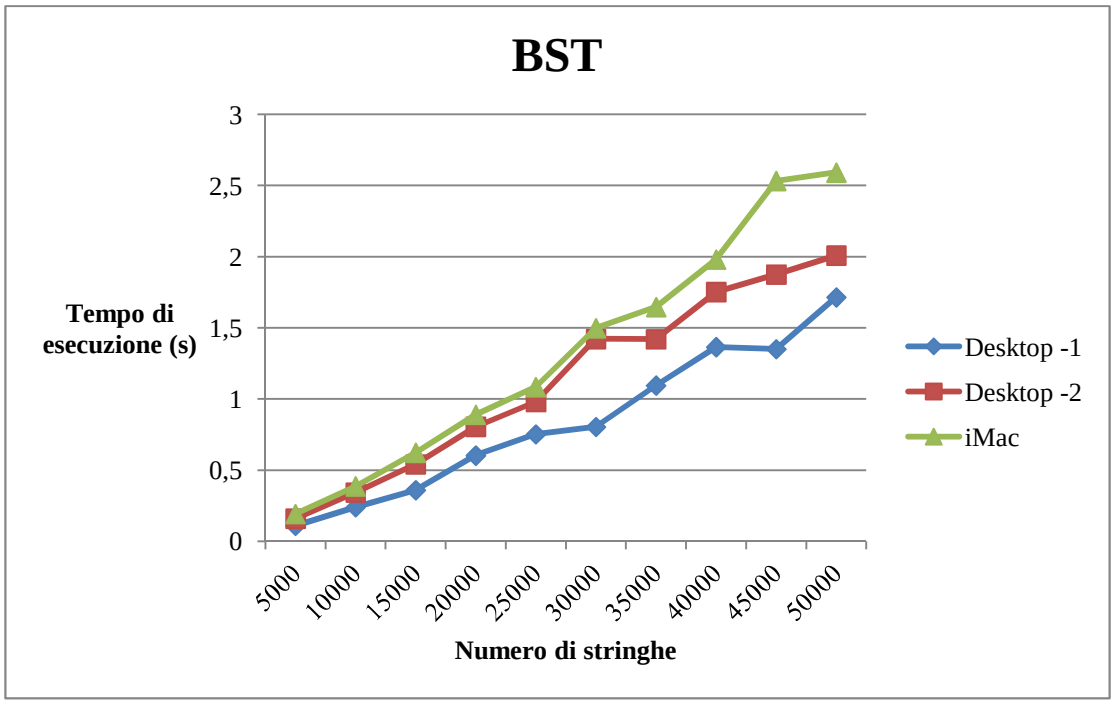

Figura 1

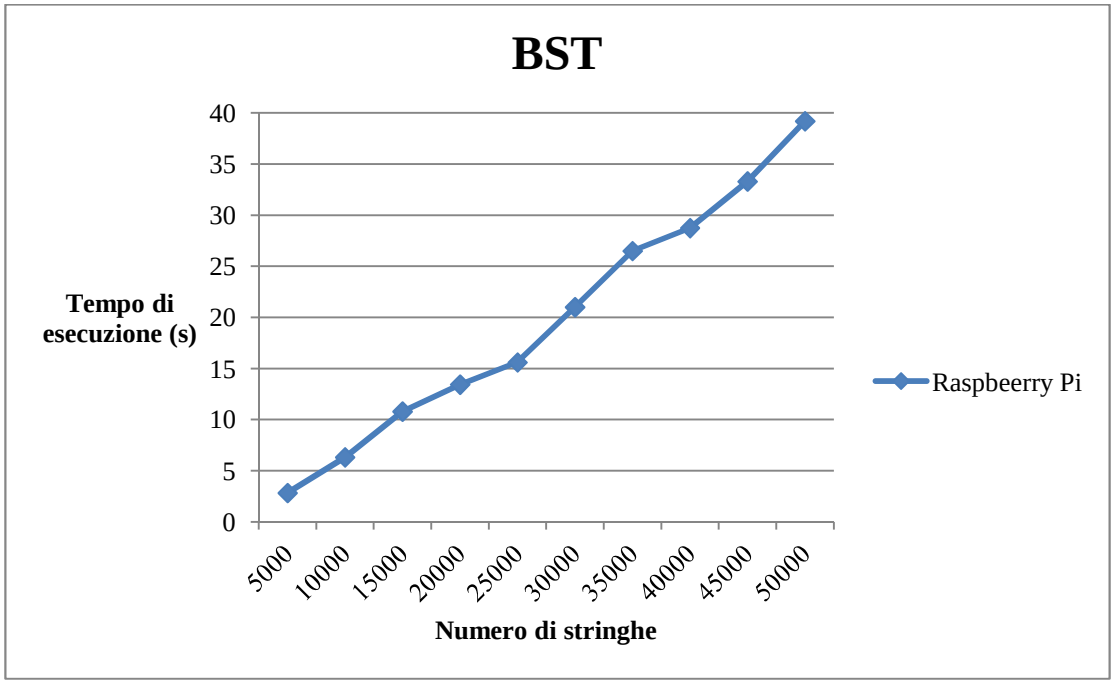

Figura 2

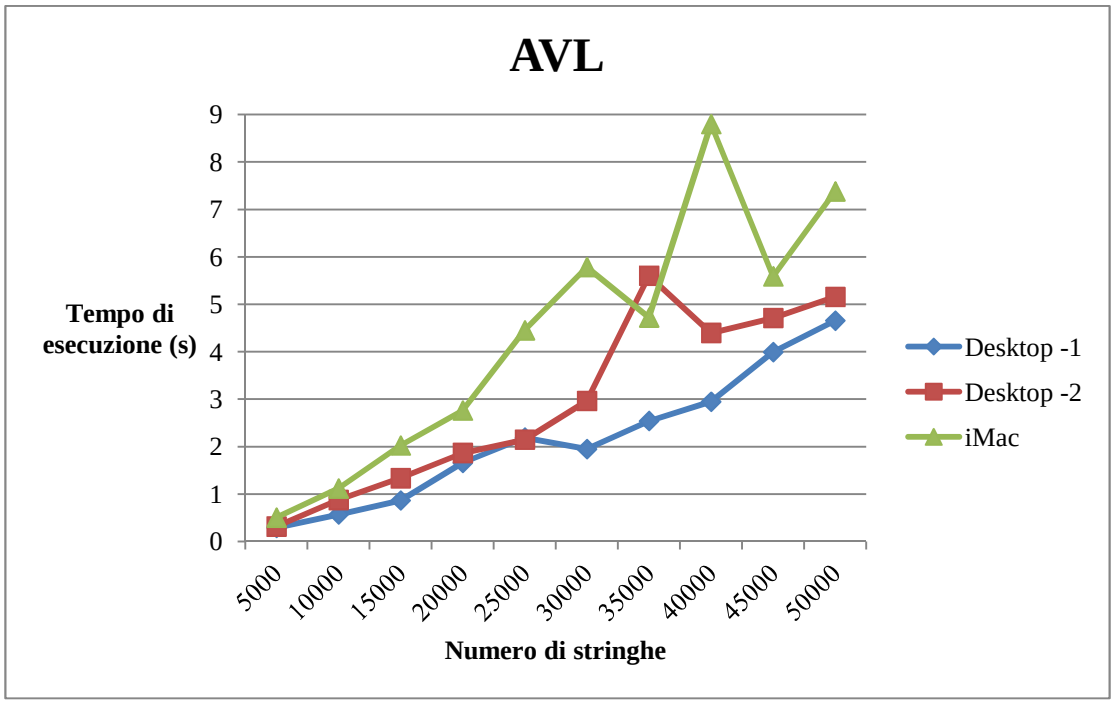

Figura 3

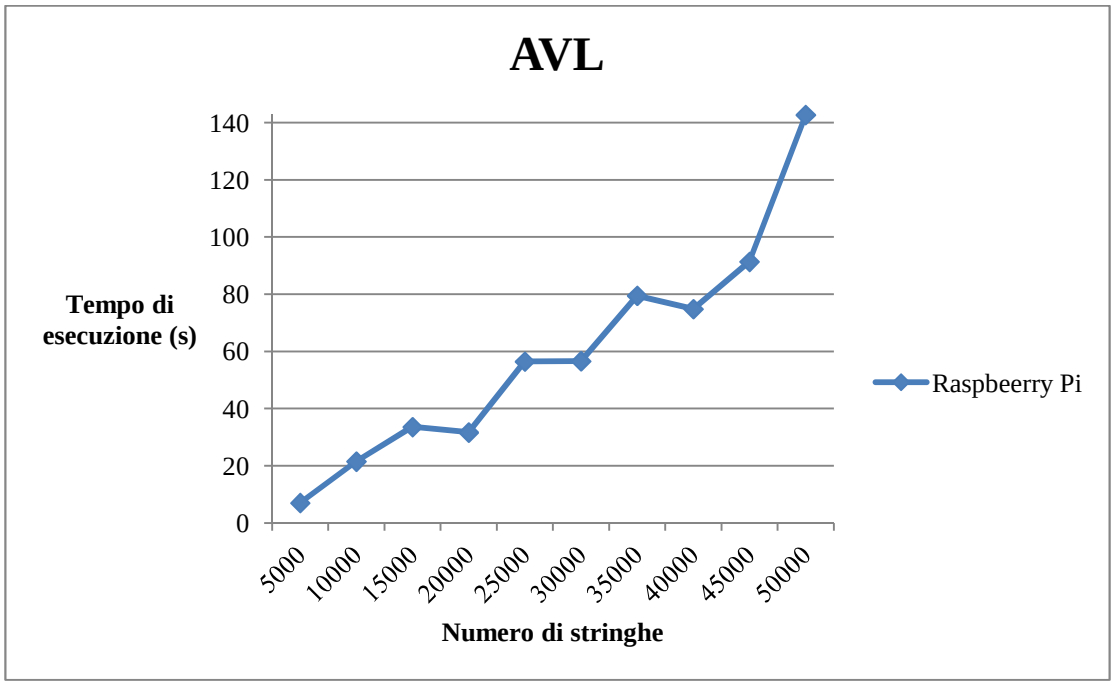

Figura 4

Presi singolarmente, i tempi di esecuzione delle due strutture dati, rispecchiano in pieno le nostre aspettative: come si può notare dalle Figure 1 e 2, 3 e 4, con l'aumentare del numero di stringhe su cui effettuare le operazioni, aumenta (in generale) il tempo totale di esecuzione. Un dato importante da considerare, però, lo si ottiene confrontando le due strutture. Per quanto riguarda le cancellazioni e gli inserimenti, si può notare che queste avvengono in modo più rapido sui BST, che non garantiscono il bilanciamento dell'albero,

rispetto agli AVL. Questo è sicuramente da imputare alle operazioni di rotazione che un AVL effettua ogniqualvolta un nodo presenti un fattore di bilanciamento maggiore o uguale a 2. I BST, non avendo l'onere di mantenersi bilanciati, effettuano cancellazioni e inserimenti in modo più rapido, a discapito della ricerca che ha tempi maggiori nei BST che negli AVL. Per dimostrare questa tesi, riportiamo un estratto dei tempi di esecuzione di "Desktop -1" per inserimenti, cancellazioni e ricerche su insiemi non ordinati di stringhe.

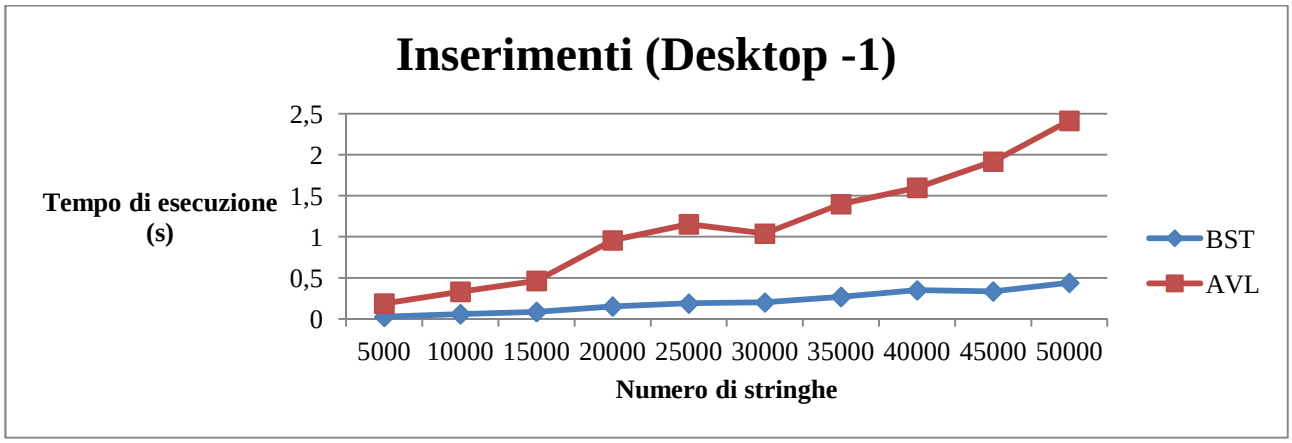

Figura 5

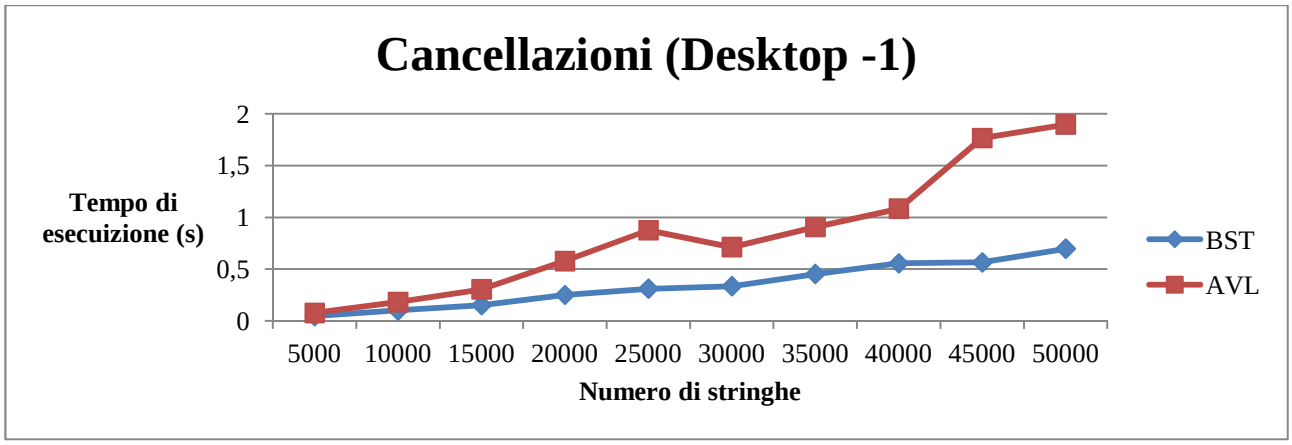

Figura 6

Come si può vedere dai grafici in Figura 5 e 6, le operazioni di inserimento e cancellazione vengono eseguite in tempo più rapido nei BST; questo perché stiamo usando degli insiemi di stringhe non ordinate tra loro. La probabilità che più stringhe vicine siano ordinate, e che quindi sbilancino di molto un BST, è relativamente piccola. Per dimostrare questa ipotesi possiamo prendere in considerazione un insieme di N parole ipotizzando queste come tutte differenti tra loro; questo dato non influenzerà affatto il risultato, dato che entrambi gli algoritmi sono impostati per collocare come figlio destro un nodo il cui padre ha la sua stessa chiave. Tutte le possibili disposizioni semplici di N parole prese a gruppi di K, sono date dalla formula:

$$D_{N,K} = \frac{N!}{(N-K)!}$$

Sappiamo, ora, che – tra tutte le disposizioni – solo due rendono ogni gruppo di K parole totalmente ordinato (crescente o decrescente). Dunque la probabilità P che un gruppo di K parole all'interno dell'insieme N sia totalmente ordinato è:

$$P = \frac{2}{D_{N,K}}$$

Con N molto grande tale probabilità diventa estremamente piccola; ancora di più se si considerano molti gruppi di K parole (magari contigui fra loro). L'albero BST risultante sarà quindi sì sempre sbilanciato, ma non tanto da rendere il divario con l'AVL (in termini di tempo) a favore di quest'ultimo. Infatti l'albero AVL, indistintamente dall'ordine o meno delle stringhe, effettua delle rotazioni per ribilanciare l'albero (qual'ora il fattore di bilanciamento di un nodo presenti un valore maggiore o uguale a 2) le quali possono portare sì benefici per le ricerche, ma anche un tempo di esecuzione maggiore per inserimenti e cancellazioni, che spesso vanifica il lavoro stesso delle operazioni di bilanciamento. Ci sentiamo di affermare che questo "vantaggio" deriva esclusivamente dalla possibilità di avere stringhe più o meno ordinate tra loro e che quindi rimane confinato a un dato puramente statistico.

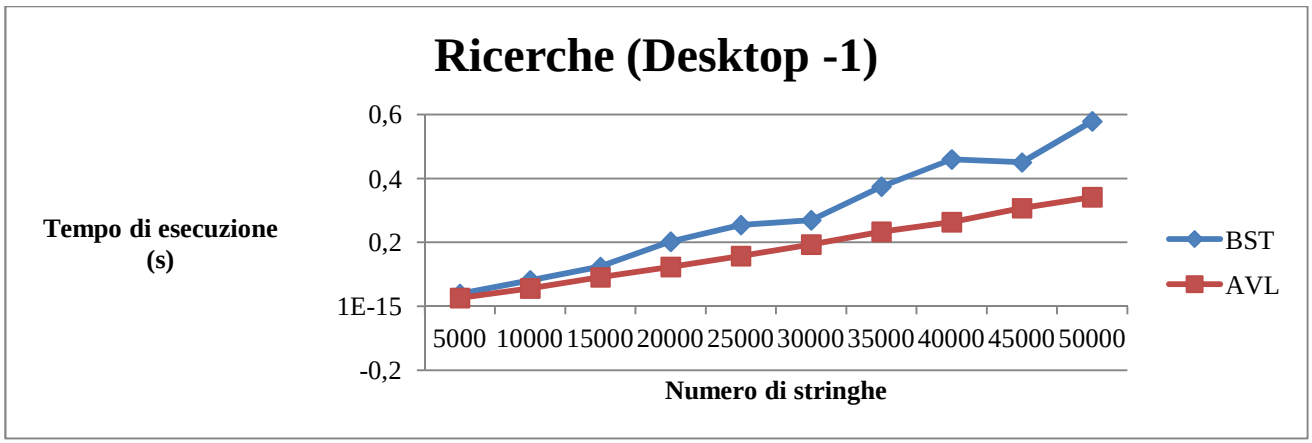

Figura 7

Per le ricerche il solo dato probabilistico di distribuzione delle stringhe, non basta a avvantaggiare il BST rispetto all' AVL: quest'ultimo (come ampiamente discusso) viene costantemente bilanciato riuscendo ad ottenere il tempo di ricerca più basso possibile. Infatti, un albero BST al massimo potrà essere bilanciato come un AVL (con opportuni inserimenti) e quindi potrà eseguire una ricerca in un tempo al più uguale a un AVL. In

questo caso, è statisticamente impossibile che stringhe casualmente importate siano disposte in modo tale da formare un BST totalmente bilanciato. Ne consegue un tempo di esecuzione minore per l'albero AVL, come mostrato dal grafico in Figura 7.

3.3 Analisi dei test relativi a BST e AVL su insiemi di elementi ordinati.

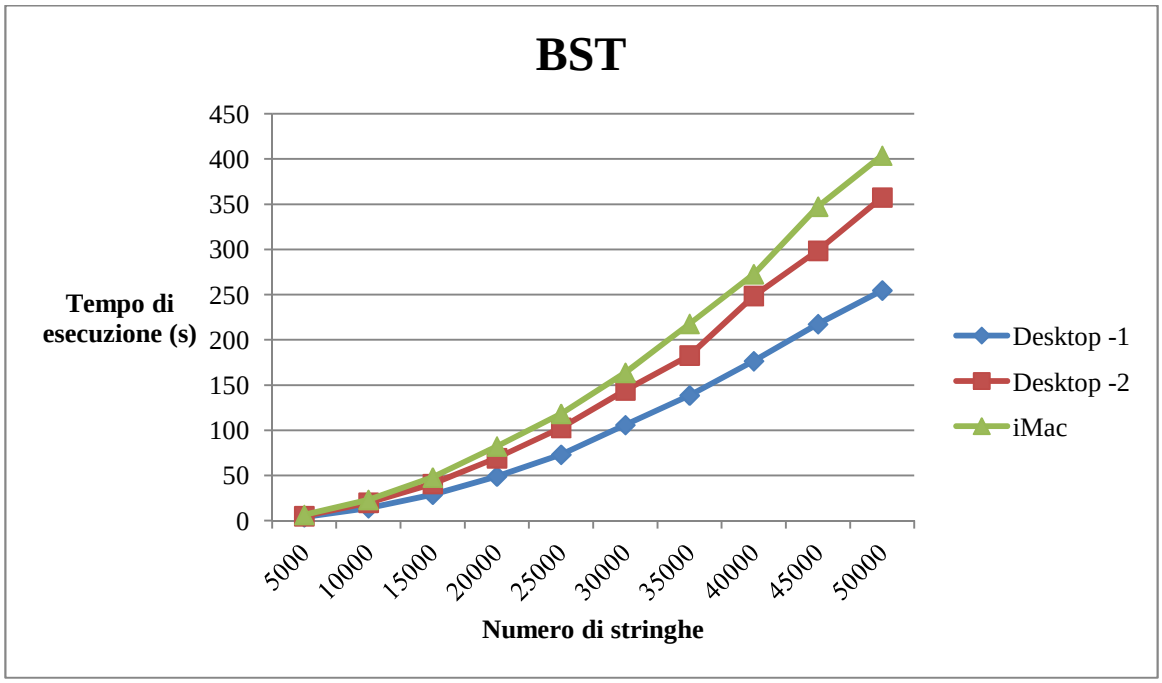

Figura 8

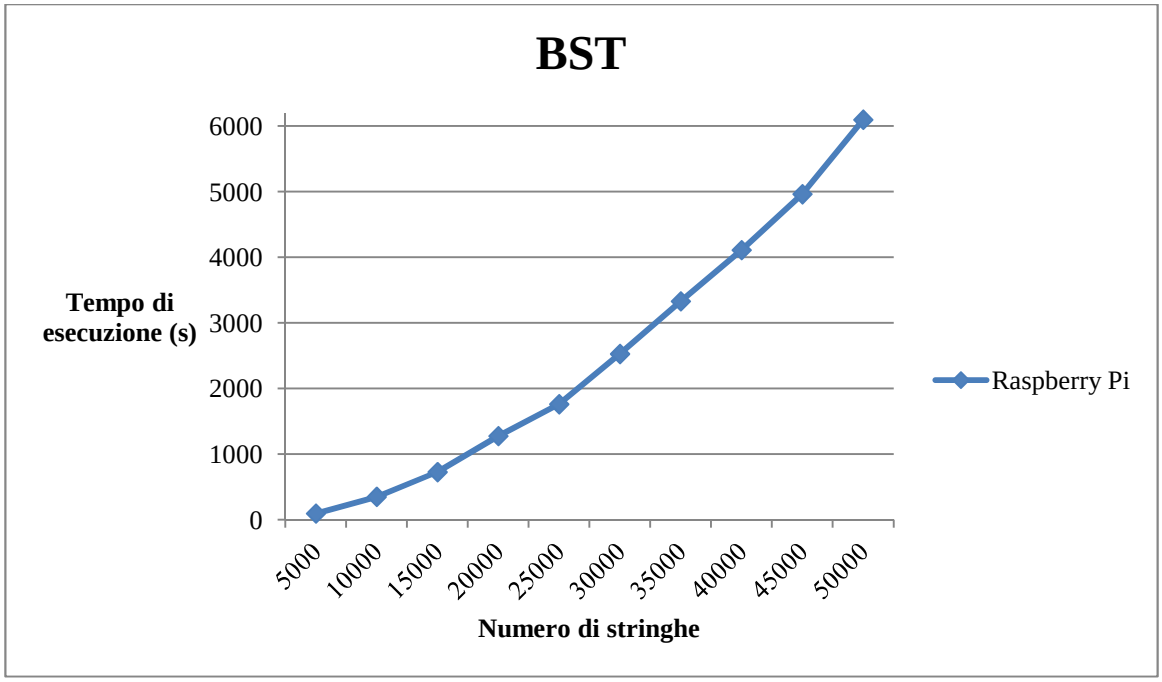

Figura 9

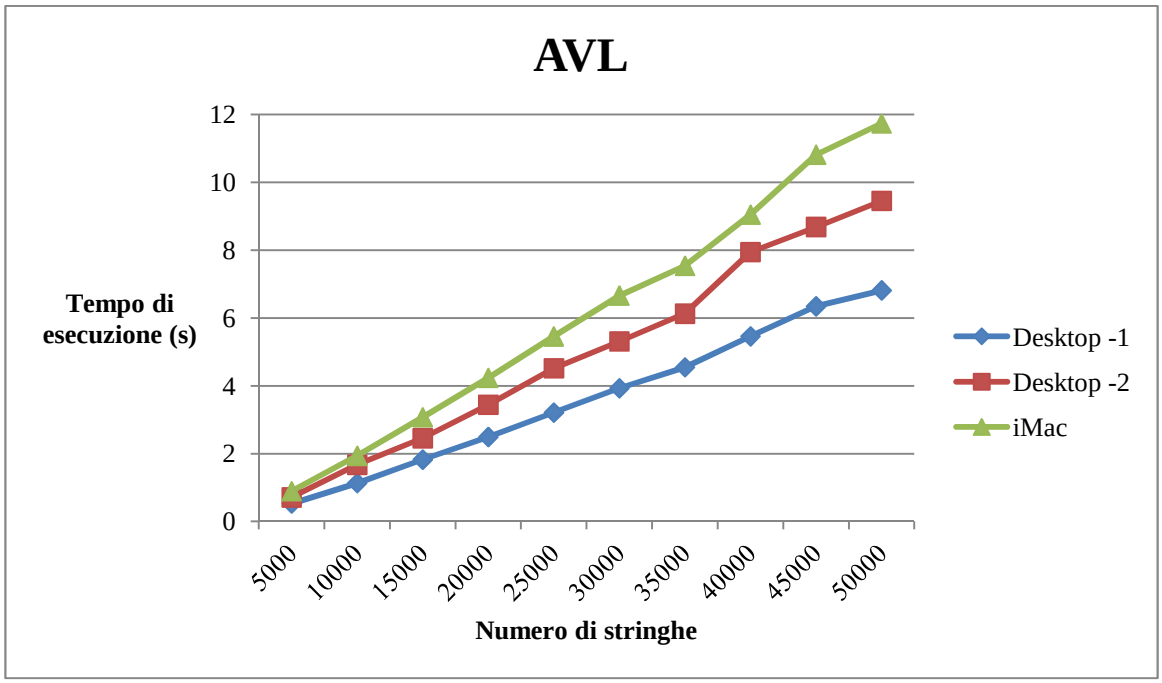

Figura 10

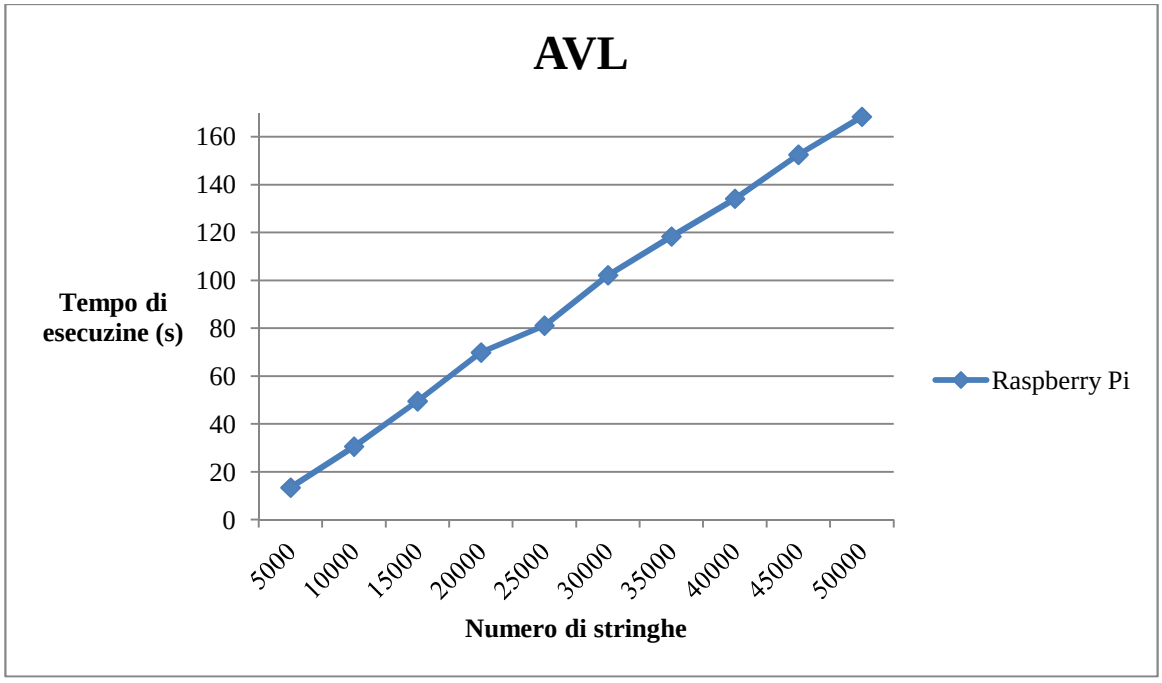

Figura 11

A differenza delle operazioni su insiemi non ordinati, il divario tra BST e AVL, rispetto al tempo di esecuzione delle stesse su insiemi ordinati di stringhe, è molto evidente. Il fatto di lavorare su stringhe ordinate tra loro, rende l'utilizzo di una struttura dati come il BST totalmente inefficiente. Infatti qualunque stringa i-esima sarà sempre maggiore della stringa (i-1)-esima; ciò indurrà il BST a posizionare ogni nuovo nodo come figlio destro del precedente, portando l'altezza dell'albero a una profondità corrispondente al numero stesso

dei nodi. Questo comporterà che ad ogni operazione di ricerca, inserimento e cancellazione, si percorrerà tutto l'albero dalla radice sino all'ultima foglia, producendo un tempo di esecuzione enormemente elevato su insiemi molto grandi di stringhe. Questo fatto si può facilmente notare confrontando tra loro i grafici di Figura 8 e 9 con quelli di Figura 10 e 11: a uno stesso numero di stringhe corrispondo tempi estremamente diversi fra loro, a favore degli alberi AVL. Infatti si nota come l'andamento del grafico sia abbastanza lineare per gli AVL, mentre assume una forma "parabolica" per le operazioni svolte sui BST. Per evidenziare tale aspetto si propone un estratto dei tempi di esecuzione di "iMac" per le ricerche su insiemi ordinati di stringhe.

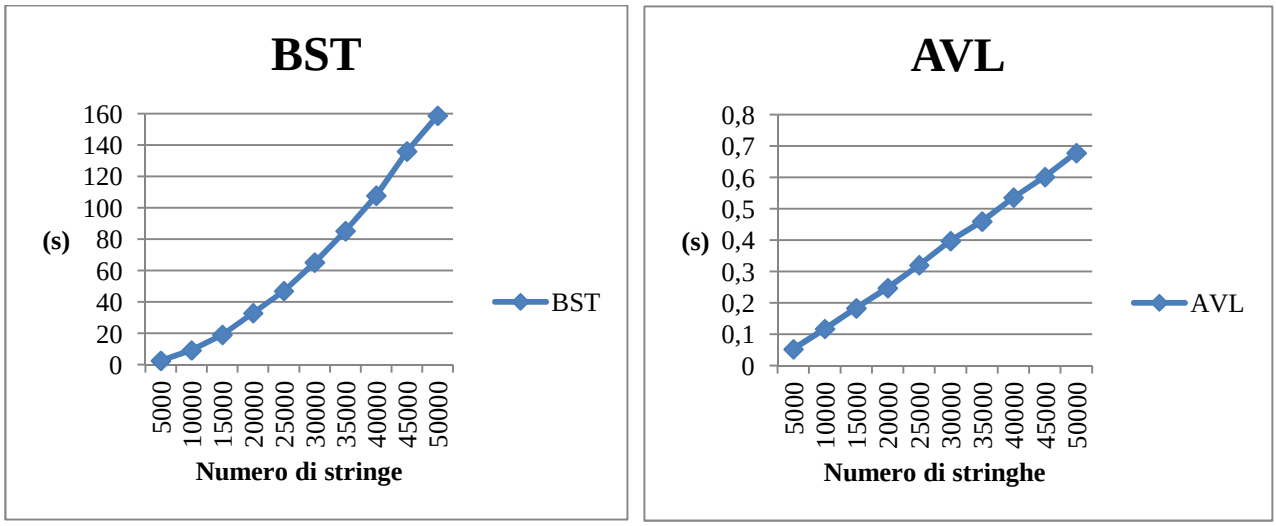

Figura 12

Come si evince dai grafici in Figura 12, i tempi relativi alle operazioni di ricerca su un BST sono estremamente maggiori rispetto a quelli di un AVL. La spiegazione, come già esposto, deriva dall'ordinamento delle stringhe nell'insieme da cui vengono prelevate. L'AVL, tenendo sempre aggiornato il fattore di bilanciamento dei nodi, non risente dello stesso problema. Si può notare dal secondo grafico, in Figura 12, che l'andamento rimane lineare qualunque sia il numero di stringhe utilizzate.

3.4 Conclusioni

Arrivati a questo punto, descriviamo in sintesi il lavoro svolto.

- ✓ Implementate entrambe le strutture dati ripercorrendo esattamente quanto spiegato a lezione;
- ✓ Strutturati i test in modo da evidenziare le principali differenze tra le due strutture dati;
- ✓ Costruiti i grafici affinché risaltino gli aspetti critici di ogni operazione;

- ✓ Descritte le motivazioni per le quali si è vista una struttura dati "primeggiare" rispetto all'altra in determinate circostanze.

Dai test svolti e dai dati sopra riportati (nonché da quelli allegati a fine documento), si può senz' altro affermare che, per qualsiasi insieme di elementi da organizzare tramite un dizionario, la struttura dati più efficiente da utilizzare tra BST e AVL è senza alcun dubbio quest'ultima. I benefici che derivano dall'utilizzo di questa struttura dati superano di gran lunga i contro; inoltre si adatta a qualsiasi gruppo di dati, agevolando l'operazione di ricerca, ossia la più importante del tipo di dato dizionario.

3.5 Problematiche relative alla dimensione della memoria principale (RAM)

Durante i nostri test, l'esecuzione dei codici relativi al programma principale non ha generato alcun tipo di problema. Esiste però la possibilità, benché minima, di incorrere in un inconveniente che può causare disturbi, e che si verifica quando la memoria principale non è in grado di contenere interamente le strutture dati utilizzate. In questa situazione il sistema operativo è progettato in modo tale da ovviare a tale inconveniente; è creata su memoria non volatile un'area di *swap*, ossia un'estensione della capacità della memoria volatile complessiva del computer. Quando si verifica la situazione sopra descritta, il S.O. tende a liberare la RAM spostando i dati meno utilizzati nell'area di swap (usando uno degli algoritmi di swapping: FIFO, LRU ecc.) per far avanzare il processo in esecuzione. Questo comporta che, ogniqualvolta il sistema richieda un dato presente nell'area di swap, questo deve essere spostato dall'hard-disk alla RAM con conseguente spreco di tempo. Infatti dal punto di vista della CPU, prelevare un dato dalla memoria principale costa circa 80ns, un tempo estremamente breve in confronto ai 16ms corrispondenti al tempo di accesso a disco rigido. I calcolatori moderni hanno un quantitativo di memoria principale estremamente elevato che rende meno probabile, ma non impossibile, questa eventualità. Nel caso in cui ciò si verificasse, assisteremmo ad un drastico aumento del tempo di esecuzione del programma e ad una diminuzione complessiva delle prestazioni della macchina.

4. Codice.

Il codice del progetto ha il seguente schema:

```
• Progetto_4.py
• /src
    • __init__.py
    • BinarySearchTree.py
        - class BinarySearchTree:
            __init__(self)
            search(self, key)
            insertNode(self, newNode)
            findMax(self, node=None)
            findPred(self, node)
            delete(self, element)
            deleteNode(self, node)
            swapInformations(self, node1, node2)
    • AVLTree.py
        - class AVLTree(BinarySearchTree):
            updateNode(self, node)
            update(self, node)
            setFlag(self, node)
            selectRotation(self, node)
            rotationLL(self, a, flag)
            rotationRR(self, a, flag)
            rotationLR(self, node, flag)
            rotationRL(self, node, flag)
            insertAVLNode(self, newNode)
            deleteAVLNode(self, element)
    • extraFunctions.py
        - color(colorType, string)
        - printDate()
        - ask(prompt, retries=4, complaint="...")
        - ask1(prompt, numbers=2, retries=4, complaint="...")
        - class Stack:
            __init__(self)
            push(self, element)
            pop(self)
            isEmpty(self)
        - rand()
        - class elapsedTime:
            __init__(self)
            addTime(self, start, stop)
            getTime(self)
    • Nodes.py
        - class BinaryNode:
            __init__(self, info, key)
        - class AVLNode:
            __init__(self, info, key)
    • TxtInput.py
        - isBoring(char)
        - class TxtInput:
            __init__(self, filename)
            splitText(self, words, dimList, dimString)
            splitTextToWords(self, words, dimList)
```

Presentiamo di seguito il codice da noi creato e utilizzato per effettuare i test da cui abbiamo ottenuto i risultati sopra descritti. Ogni modulo è comprensivo di un piccolo programma per testarlo (anche se non riportato in questo documento); si invita il lettore ad importare come principale ogni modulo di cui si intende verificarne la corretta implementazione. Inoltre in tutto il codice sono presenti commenti esplicativi, e ogni classe e funzione ha le proprie *docstring* informative per una corretta interpretazione del codice.

NOTA: ogni paragrafo del capitolo 4 corrisponde ad un modulo del programma, completi di opportuni commenti, e codice indentato.

4.1 BinarySearchTree.py

```
# -*- coding: utf-8 -*-
"""
Module name: BinarySearchTree
Dependences: Nodes, extraFunctions
Author: Simone Minasola
simone.minasola@gmail.com
CHANGELOG

Versione 0.2 (beta) - 15/05/2013
-Aggiunte e sistemate le docstring per ogni funzione
-Migliorato il codice per i test di debug quando il modulo viene importato come
principale

Versione 0.1 (beta) - 13/05/2013
-Eliminati elementi di debug
-Sdoppiata la funzione delete in delete e deleteNode
-Aggiunta la funzione stampaAlbero per facilitare nella fase di test

Versione 0.0 (alpha) - 10/05/2013
"""

from Nodes import BinaryNode
from extraFunctions import Stack

class BinarySearchTree:
	"""
	Questa classe implementa un albero binario di ricerca (BST).
	La creazione dei nodi è affidata alla classe BinaryNode.
	Si è deciso di inserire a destra i nodi con chiave uguale a    quella
	del loro futuro padre.
	"""
	def __init__(self):
		self.root=None

	def search(self, key):
		"""
		Funzione per la ricerca di un nodo tramite la sua chiave.
		Questa funzione ritorna una tupla "t" così composta:
		1)t[0]==None se la chiave immessa non appartiene a nessun nodo;
		in questo caso, t[1]==l'ultima foglia analizzata durante
		la fase di ricerca.
		2)t[0]==l'info del nodo trovato tramite la chiave immessa (nel caso
		in cui esista); in questo t[1]==nodo con la chiave cercata
		"""
		if key==None:
```

```
                return None

        currentNode=self.root #Comincio la ricerca dalla radice
        while(currentNode!=None):
                if key==currentNode.key: #Ho trovato il nodo
                        return currentNode.info, currentNode
                else:
                        if key < currentNode.key:
                                if currentNode.leftSon!=None:
                                        currentNode=currentNode.leftSon #Mi sposto al
                                figlio sinistro
                                else:
                                        return None, currentNode
                        else:
                                if currentNode.rightSon!=None:
                                        currentNode=currentNode.rightSon #Mi sposto al
                                figlio destro
                                else:
                                        return None, currentNode

    def insertNode(self, newNode):
        """
        Questa funzione inserisce un nuovo nodo nel BST rispettando
        le proprietà che contraddistinguono questa struttura dati.
        Nel caso in cui la chiave del nodo da inserire A sia uguale alla
        chiave di un altro nodo gia esistente B, si è scelto di inserire
        A a destra di B.
        """
        if self.root==None: #Se la radice è vuota, inserisci il nodo come radice
                self.root=newNode
        else:
                currentNode=self.root #Comincio dalla radice
                predNode=None
                while currentNode!=None: #Cerco il nodo che lo precede
                        predNode=currentNode
                        if newNode.key < currentNode.key:
                                currentNode=currentNode.leftSon
                        else:
                                currentNode=currentNode.rightSon

                #Attacco newNode come una foglia
                if newNode.key < predNode.key:
                        predNode.leftSon=newNode
                        newNode.father=predNode
                else:
                        predNode.rightSon=newNode
                        newNode.father=predNode

    def findMax(self, node=None):
        """
        Funzione per la ricerca del nodo con chiave massima all'interno
        del BST. Se la variabile "node" viene passata alla funzione, il
        nodo con chiave massima verrà ricercato a partire dal sottoalbero
        che ha come radice proprio il nodo passato; altrimenti la ricerca
        verrà effettuata a partire dalla radice.
        """
        if node==None:
                node=self.root
        while node.rightSon!=None: #Scendo sempre al figlio destro del nodo
                node=node.rightSon
        return node

    def findPred(self, node):
        """
        Funzione per la ricerca del predecessore di un nodo.
        Viene restituito il nodo predecessore di quello passato
        alla funzione; nel caso quest'ultimo non esista, viene
        restituito None.
        """
```

```
        if node==None:
                return None
        else:
                if node.leftSon != None: #Il nodo ha figlio sinistro
                        return self.findMax(node.leftSon) #Il predecessore è il massimo
                        del sottoalbero sx
                else: #Il nodo non ha figlio sinistro
                        while node.father!=None and node==node.father.leftSon:
                                node=node.father #Si sale fino alla prima svolta a
                        sinistra
                        return node.father

def delete(self, element):
        """
        Funzione per la cancellazione di un nodo con chiave=="element" dal BST.
        Se non esiste un nodo con chiave "element" nel BST, la funzione
        restituirà False; altrimenti, verrà invocata la funzione deleteNode
        e restituito il valore father_, che rappresenta il padre del nodo eliminato.
        Questo sarà utile per la classe AVLTree, dato che erediterà questa funzione.
        """
        node=self.search(element)
        if node==None or node[0]==None:
                return False
        node=node[1]
        father_=self.deleteNode(node)
        return father_

def deleteNode(self, node):
        """
        Funzione che determina il modo corretto per eliminare un nodo dal BST.
        Si distinguono 3 casi:
        1) il nodo è una foglia
        2) il nodo ha 1 figlio
        3) il nodo ha 2 figli
        """
        father_=node.father

        #1) se node==foglia
        if node.leftSon==None and node.rightSon==None:
                if node.father!=None: #Il nodo da eliminare non è la radice
                        father=node.father
                        son=node
                        if node==node.father.leftSon:
                                father.leftSon=None #Stacco la foglia
                        elif node==node.father.rightSon:
                                father.rightSon=None #Stacco la foglia
                        son.father=None #Elimino i legami di parentela
                else: #Il nodo da eliminare è radice del BST
                        self.root=None #Elimino la radice

        #2) se node==padre di un solo figlio...
        elif (node.leftSon==None and node.rightSon!=None) or (node.leftSon!=None and
        node.rightSon==None):
                if node.leftSon!=None: #Ha solo il figlio sinistro
                        son=node.leftSon
                elif node.rightSon!=None: #oppure ha solo il figlio destro
                        son=node.rightSon
                if node==self.root: #Se il nodo da cancellare è la radice
                        son.father=None #Stacca il padre dal figlio
                        self.root=son #Il figlio diventa la nuova radice
                elif node!=self.root: #se il nodo da cancellare non è la radice
                        father=node.father
                        #Si attaccano i 'nipoti' ai 'nonni'
                        if node==node.father.rightSon:
                                father.rightSon=son
                        elif node==node.father.leftSon:
                                father.leftSon=son
                        son.father=father
```

```
        #3) se node==padre di due figli
        elif node.leftSon!=None and node.rightSon!=None:
                pred=self.findPred(node) #Trovo il predecessore
                self.swapInformations(pred, node) #Si scambiano info e key dei nodi
                        #Mi ritrovo in uno dei due casi precedenti
                father_=self.deleteNode(pred)

        return father_

def swapInformations(self, node1, node2):
        "Funzione che scambia info e key di due nodi."
        tmp1, tmp2=node1.info, node1.key
        node1.info=node2.info
        node1.key=node2.key
        node2.info=tmp1
        node2.key=tmp2

#ONLY FOR DEBUG
def stampaAlbero(self):
        stack = Stack()
        if self.root!=None:
                stack.push([self.root, 0, None])
        else:
                print "Albero vuoto!"
        while not stack.isEmpty():
                current = stack.pop()
                level = current[1]
                if current[2]==None:
                        print "|---"*level + (",current[0].info,",",current[0].key,")
                RADICE"
                if current[2]!=None:
                        print "|---"*level + "
                (",current[0].info,",",current[0].key,")"+current[2]
                if current[0].rightSon != None:
                        stack.push([current[0].rightSon, level + 1, " figlio DX"])
                if current[0].leftSon != None:
                        stack.push([current[0].leftSon, level + 1, " figlio SX"])
                level += 1
```

4.2 AVLTree.py

```
# -*- coding: utf-8 -*-
"""
Module name: AVLTree
Dependences: Nodes, extraFunctions
Author: Simone Minasola
simone.minasola@gmail.com
CHANGELOG

Versione 0.1 (beta) - 07/06/2013
-Aggiunti elementi di debug per test
-Aggiunte le docstring per le funzioni di classe
-Risolto un bug relativo alle rotazioni SX-DX e DX-SX

Versione 0.0 (alpha) - 06/06/2013
"""

from BinarySearchTree import BinarySearchTree
from Nodes import AVLNode
from extraFunctions import Stack

class AVLTree(BinarySearchTree):
	"""
	Questa classe implementa un albero AVL.
	La creazione dei nodi è affidata alla classe AVLNode
	"""

	def updateNode(self, node):
		"""
		Funzione che si occupa di aggiornare i dati di un nodo, ovvero
		l'altezza del sottoalbero sx, l'altezza del sottoalbero dx, e
		il fattore di bilanciamento (q). Quando quest'ultimo risulta
		essere maggiore di 2, viene invocata la funzione per bilanciare
		l'albero.
		"""
		if node.leftSon!=None:
			node.sx=max(node.leftSon.sx, node.leftSon.dx)+1
		else:
			node.sx=0
		if node.rightSon!=None:
			node.dx=max(node.rightSon.sx, node.rightSon.dx)+1
		else:
			node.dx=0
		node.q=abs(node.sx-node.dx)
		if node.q>=2:
			self.selectRotation(node)

	def update(self, node):
		"""
		Questa funzione aggiorna i dati di tutti i nodi incontrati durante il
		percorso tra il nodo di riferimento passato alla funzione e la radice
		dell'albero.
		"""
		while node!=None:
			self.updateNode(node)
			node=node.father

	def setFlag(self, node):
		"""
		Questa funzione accetta in entrata un nodo A e restituisce:
		- 1: se A è la radice dell'albero
		- 2: se A è figlio sinistro di suo padre
		- 3: se A è figlio destro di suo padre
		"""
		if node==self.root:
			flag=1
		else:
```

```
                if node==node.father.leftSon:
                        flag=2
                else:
                        flag=3
        return flag

    def selectRotation(self, node):
        """
        Questa funzione si occupa di selezionare la giusta rotazione
        da eseguire quando il fattore di bilanciamento di un nodo è
        maggiore o uguale a 2
        """

        flag=self.setFlag(node)

        #Rotazione sx-sx
        if node.leftSon!=None and node.leftSon.sx>=node.leftSon.dx:
                temp=self.rotationLL(node, flag)
                self.updateNode(node)
                self.updateNode(temp)

        #Rotazione sx-dx
        elif node.leftSon!=None and node.leftSon.sx<node.leftSon.dx:
                temp=self.rotationLR(node, flag)

        #Rotazione dx-dx
        elif node.rightSon!=None and node.rightSon.sx<=node.rightSon.dx:
                temp=self.rotationRR(node, flag)
                self.updateNode(node)
                self.updateNode(temp)

        #Rotazione dx-sx
        else:
                temp=self.rotationRL(node, flag)

def rotationLL(self, a, flag):
    """
    Funzione che si occupa della rotazione SX-SX
    """
    b=a.leftSon
    temp=b
    a.leftSon=b.rightSon #Attacco il figlio dx di b come figlio sx di a
    if a.leftSon!=None:
            a.leftSon.father=a #Imposto a come padre del suo nuovo figlio sx
    temp.rightSon=a #Attacco a come figlio dx di b
    temp.father=a.father #Attacco a b il padre che era di a
    a.father=temp #Ora il padre di a è temp
    if flag==1: #Se il nodo sbilanciato era la radice: root=temp
            self.root=temp
    else:
            if flag==2: #Altrimenti attacca temp come figlio sx o dx di suo padre
                    temp.father.leftSon=temp
            else:
                    temp.father.rightSon=temp
    return temp

def rotationRR(self, a, flag):
    """
    Funzione che si occupa della rotazione DX-DX
    """
    b=a.rightSon
    temp=b
    #Speculare alla rotazione sx-sx
    a.rightSon=b.leftSon
    if a.rightSon!=None:
            a.rightSon.father=a
    temp.leftSon=a
    temp.father=a.father
    a.father=temp
```

```
        if flag==1:
                self.root=temp
        else:
                if flag==2:
                        temp.father.leftSon=temp
                else:
                        temp.father.rightSon=temp
        return temp

def rotationLR(self, node, flag):
        """
        Funzione che si occupa della rotazione SX-DX
        """
        #Eseguo una rotazione dx-dx sul figlio sx del nodo sbilanciato
        a=self.rotationRR(node.leftSon, self.setFlag(node.leftSon))
        #Eseguo una rotazione sx-sx sul nodo sbilanciato
        b=self.rotationLL(node, flag)
        #Aggiorno i nodi
        self.updateNode(b.leftSon)
        self.updateNode(b.rightSon)
        self.updateNode(b)

def rotationRL(self, node, flag):
        """
        Funzione che si occupa della rotazione DX-SX
        """
        #Speculare alla rotazione sx-dx
        a=self.rotationLL(node.rightSon, self.setFlag(node.rightSon))
        b=self.rotationRR(node, flag)
        self.updateNode(b.leftSon)
        self.updateNode(b.rightSon)
        self.updateNode(b)

def insertAVLNode(self, newNode):
        """
        Funzione che esegue l'inserimento di un nodo in un albero AVL.
        Si appoggia alla funzione di inserimento di un nodo ereditata
        dalla classe BinarySearchTree.
        """
        self.insertNode(newNode) #Inserisci nodo come in un BST
        self.update(newNode) #Aggiorna i nodi fino alla radice

def deleteAVLNode(self, element):
        """
        Funzione che esegue l'eliminazione di un nodo da un albero AVL.
        Si appoggia alla funzione di eliminazione di un nodo ereditata
        dalla classe BinarySearchTree.
        """
        father=self.delete(element) #Elimina il nodo come in un BST
        if father==False or father==None:
                return False #Ritorna False se il nodo non esisteva
        self.update(father) #Aggiorna i nodi fino alla radice

#ONLY FOR DEBUG
def stampaAlbero(self):
        stack = Stack()
        if self.root!=None:
                stack.push([self.root, 0, None, None, None]) #Aggiunti altezza sottoalbero
        sx | dx
        else:
                print "Albero vuoto!"
        while not stack.isEmpty():
                current = stack.pop()
                level = current[1]
                if current[2]==None:
                        print "|---"*level+" (",current[0].info,",",current[0].key,")
                RADICE"+" "+str(current
        [0].sx) + "|" + str(current[0].dx)
                if current[2]!=None:
```

```
            print "|---"*level+"
      (",current[0].info,",",current[0].key,")"+current[2]+" " + str
(current[0].sx) + "|" + str(current[0].dx)
      if current[0].rightSon != None:
            stack.push([current[0].rightSon, level + 1, " figlio DX"])
      if current[0].leftSon != None:
            stack.push([current[0].leftSon, level + 1, " figlio SX"])
      level += 1
```

4.3 extraFunctions.py

```
"""
Module name: extraFunctions
Dependences: platform, random, time
Author: Simone Minasola
simone.minasola@gmail.com
DESCRIPTION:
Modulo contenente funzioni ausiliarie per il programma principale.
"""

from platform import system
from random import uniform
from time import strftime

def color(colorType, string):
	"""
	Funzione usata per colorare il testo dello standard output su bash.
	ATTENZIONE: colora il testo solamente su terminali di sistemi OS X o Linux!
	"""
	if system()=="Windows": #controllo l'ambiente in cui viene eseguito il programma
		print string
		return

	#background
	if colorType=='BBLACK': code="\033[40m" #NERO
	elif colorType=='BRED': code="\033[41m" #ROSSO
	elif colorType=='BGREEN': code="\033[42m" #VERDE
	elif colorType=='BYELLOW': code="\033[43m" #GIALLO
	elif colorType=='BBLUE': code="\033[44m" #BLU
	elif colorType=='BMAGENTA': code="\033[45m"#MAGENTA
	elif colorType=='BACQUA': code="\033[46m" #ACQUA
	elif colorType=='BWHITE': code="\033[47m" #BIANCO

	#foreground
	elif colorType=='BLACK': code="\033[30m" #NERO
	elif colorType=='RED': code="\033[31m" #ROSSO
	elif colorType=='GREEN': code="\033[32m" #VERDE
	elif colorType=='YELLOW': code="\033[33m" #GIALLO
	elif colorType=='BLUE': code="\033[34m" #BLU
	elif colorType=='MAGENTA': code="\033[35m" #MAGENTA
	elif colorType=='ACQUA': code="\033[36m" #ACQUA
	elif colorType=='WHITE': code="\033[37m" #BIANCO

	else:
		print string
		return

	END = "\033[0m" #RIPRISTINA COLORE
	print code+string+END

def printDate():
	"Ritorna la data corrente come una stringa di testo."
	return 'File generato il ' + strftime('%x') + ' alle ore '+strftime('%X')+'\n\n'

def ask(prompt, retries=4, complaint="Rispondere esclusivamente con y/Y o n/N"):
	"""
	Funzione che stampa a schermo "prompt" e ritorna True se viene dato in input
	y/Y, False se viene dato in input n/N. La variabile retries individua il
	numero di tentativi massimi di inserimento: ad ogni inserimento sbagliato,
	viene scritto a schermo il "complaint".
	"""
	while True:
		say=raw_input(prompt)
		if say in ('y', 'Y'): return True
		if say in ('n', 'N'): return False
		retries=retries-1
```

```
            if retries < 0: raise IOError, color("BRED","Troppi tentativi falliti!!
      Chiusura del
      programma."), exit()
            print complaint

def ask1(prompt, numbers=2, retries=4, complaint="Scegliere esclusivamente fra le opzioni
date"):
      """
      Funzione 'copia' di ask. In questo caso, si possono introdurre un numero variabile
      di scelte
      (indicato da numbers). Ogni scelta deve essere identificata da un numero. Se entro
      "retries"
      tentativi l'utente seleziona una scelta giusta, questa viene ritornata.
      """
      while True:
            say=raw_input(prompt)
            try:
                  s=int(say)
                  for i in range(1,numbers+1):
                        if s==i:
                              return i
                  #retries=retries-1
                  print complaint
            except ValueError:
                  print complaint
            retries=retries-1
            if retries < 0: raise IOError, color("BRED","Troppi tentativi falliti!!
      Chiusura del
      programma."), exit()

class Stack:
      """
      Classe che implementa una semplice pila.
      """

      def __init__(self):
            self.elements = []

      def push(self, element) :
            self.elements.append(element)

      def pop(self):
            return self.elements.pop()

      def isEmpty(self):
            return (self.elements == [])

      def rand():
      """
      Questa funzione viene utilizzata per decidere se una stringa deve essere
      inserita, cercata o cancellata in un albero. Viene generato un numero in
      virgola P tale che 0<=p<1. Il segmento [0,1) verrà spezzato in 3 parti
      uguali: p1, p2, p3. Se il numero p si trova in
      [0, p1): la stringa verrà inserita, in questo caso la funzione ritorna 1
      [p1, p2): la stringa verra cercata, in questo caso la funzione ritorna 2
      [p2, 1]: la stringa verrà eliminata, in questo caso la funzione ritorna 3
      """
      p=uniform(0,1)
      p1=1.0/3.0
      p2=p1+p1
      if p<p1:
            return 1
      elif p>=p1 and p<p2:
            return 2
      else:
            return 3

class elapsedTime:
      """
```

```
Una semplice classe per misurare accuratamente il tempo di esecuzione del
programma.
"""

def __init__(self):
      self.initialTime=0

def addTime(self, start, stop):
      self.initialTime+=stop-start

def getTime(self):
      return self.initialTime

if __name__ == "__main__":
      print "Modulo importato come principale!!!"
      print "Questo modulo contiene solamente funzioni ausiliarie per il corretto
funzionamento del
programma."
```

4.4 Nodes.py

```
"""
Module name: Nodes
No dependences
Author: Simone Minasola
simone.minasola@gmail.com
DESCRIPTION:
Modulo che implementa due classi nodo, essenziali per le
strutture dati del programma.
"""

class BinaryNode:

    def __init__(self, info, key):
        self.key=key
        self.info=info
        self.father=None
        self.leftSon=None
        self.rightSon=None

class AVLNode:

     def __init__(self, info, key):
        self.key=key
        self.info=info
        self.father=None
        self.leftSon=None
        self.rightSon=None
        self.sx=0 #Altezza sottoalberi sx e dx
        self.dx=0
        self.q=0 #Fattore di bilanciamento
```

4.5 TxtInput.py

```
# -*- coding: utf-8 -*-
"""

########################################
# ONLY FOR FILES ENCODED IN UTF-8     #
########################################

Module name: TxtInput
Dependences: extraFunctions
Author: Simone Minasola
simone.minasola@gmail.com
CHANGELOG

Versione 0.3 (beta) - 15/05/2013
-Aggiunta la funzione splitTextToWords
-Eliminata del tutto la classe TxtOutput per inefficienza
-Eliminate variabili globali utilizzate per le classi e migliorato il codice del
costruttore della classe

Versione 0.2 (beta) - 05/04/2013
-Aggiunta la classe TxtOutput per gestire la scrittura su file di qualsiasi tipo di dato
-Modificato i nomi di funzioni, classe e variabili secondo uno stile di scrittura più
appropriato
-Aggiunto controllo in TxtInput (grossolano) per verificare che il file non sia composto
da soli caratteri noiosi(definiti nella funzione isBoring)
-Ora i caratteri accentati e i simboli non verranno presi in considerazione (in input)
-Eliminati elementi di debug
-Aggiunte le docstring per ogni funzione
Versione 0.1 (alpha) - 25/03/2013
-Aggiunto controllo in TxtInput per verificare quando la lettura del file arriva alla
fine
-Aggiunto controllo in TxtInput per verificare se un file è vuoto

Bug conosciuti
1)i caratteri accentati e alcuni simboli non vengono riconosciuti
2)se un file è composto da soli caratteri noiosi (definiti nella funzione isBoring) si
incorre in un loop infinito

Versione 0.0 (alpha) - 23/03/2013
"""
#import sys, time
from extraFunctions import ask, color, printDate

def isBoring(char):
	"""
	Questa funzione restituisce True se un carattere preso in considerazione risulta
	essere 'noioso' per il programma, False altrimenti. La variabile string contiene
	una stringa di tutti i caratteri che non verranno presi in considerazione.
	"""
	string="ï<>«»@#{}èéòóàùì|[]!£$%&/()=?'^\+*§°_-.:,;"+'"'+"\r"
	for i in string:
		if char==i:
			return True
	return False

class TxtInput:
	"""
	Questa classe permette di gestire i file di testo in input, la sua divisione in
	stringhe di una certa dimensione decisa dall'utente e l'inserimento di
	quest'ultime in liste di grandezza anch'essa stabilita dall'utilizzatore.
	"""

	def __init__(self, filename):
		try: #Si cerca di aprire il file
			self.filename=filename
			txt=open(self.filename, 'r')
```

```
                self.offset=0
            except IOError, OSError: #Viene catturata l'eccezione
                color("BRED", "Il file non esiste o non hai i permessi sufficienti
                per aprirlo.\n"+\
                "Ricordati di non omettere l'estensione del file.")
                exit()
            else:
                lines=len(txt.readlines())
                if lines==0: #Se il file è vuoto, esci
                    color("BRED", "Il file è vuoto!\nRiavviare il programma con un
                file txt pieno.")
                txt.close()
                exit()
            color("BGREEN", "File aperto con successo")
            txt.close()

        def setOffset(self, offset):
            """
            Questa funzione imposta l'offset dell'oggetto TxtInput.
            """
            self.offset=offset

        def splitText(self, words, dimList, dimStrings):
            """
            Questa funzione accetta come argomenti words: lista che verrà riempita con
            le stringhe dimList: numero di stringhe che dovranno essere inserite nella
            lista dimString: dimensione in byte di ogni stringa di testo
            La funzione restituirà la lista di partenza con gli elementi aggiunti
            durante il processo di riempimento.
            """
            txt=open(self.filename, 'r')
            txt.seek(self.offset) #Imposta l'indicatore di posizione
            text=txt.read().strip() #Estrapola tutto il testo
            txt.close()
            lenght=len(text)
            position=0

            for i in range(dimList):
                cont = ctrl = 0 #ctrl variabile di controllo
                word=""
                while cont < dimStrings:
                    if lenght==position: #Ricomincia se si raggiunge la fine di
                words
                        position=0
                    char=text[position] #Estrapola un carattere alla volta
                    if not isBoring(char) and char!='\n' and char!=' ': #Controlla
                il carattere
                        word=word+char
                        cont+=1
                    else:
                        ctrl+=1 #Incrementa ctrl se rileva un carattere non
                    idoneo
                    if ctrl==30:
                        color("YELLOW", "Sicuro che il file non sia composto da
                    soli caratteri o simboli
                    non riconosciuti(y/n)\n"+\
                    "-Guardare la funzione isBoring nel modulo TxtInput per
                maggiori dettagli-\n")
                        if ask(""):
                            ctrl=0
                        else:
                            color("BRED", "Chiusura del programma.")
                            exit()
                    position+=1
            words.append(word)
        self.offset=position #Setta l'indicatore di posizione
        return words

def splitTextToWords(self, words, dimList):
```

```
"""
Questa funzione accetta come argomenti
words: lista che verrà riempita con le stringhe
dimList: numero di stringhe che dovranno essere inserite nella lista
La funzione restituirà la lista di partenza con gli elementi aggiunti durante il
processo di riempimento, che prevede l'inserimento di parole complete (ovvero
parole formate da tutte le lettere antecedenti ad uno spazio o un ritorno a capo).
"""

txt=open(self.filename, 'r')
txt.seek(self.offset)

#ctrl=0
cont=0 #Contatore
while cont < dimList:
	word=""
	ctrl=0
	while True:
		char=txt.read(1) #Estaggo un carattere alla volta
		if char=='': #Se arrivo alla fine del file...
			self.offset=0 #...setto l'offseto a 0
			txt.seek(self.offset)
		elif char!='\n' and char!=' ':
			if not isBoring(char):
				word=word+char
			else: #Se è un carattere'noioso' lo scarto...
				ctrl+=1 #...e incremento ctrl
		elif ctrl==30:
			color("YELLOW", "Sicuro che il file non sia composto da soli
		caratteri o simboli
		non riconosciuti(y/n)\n"+\ "-Guardare la funzione isBoring nel modulo
	TxtInput per maggiori dettagli-\n")
			if ask(""):
				ctrl=0 #Reimposto ctrl a 0
			else:
				color("BRED", "Chiusura del programma.")
				exit()
		else:
			break #Esco dal ciclo
	if word!='':
		words.append(word)
		cont+=1
self.offset=txt.tell() #Setto l'offset alla posizione corrente
txt.close()
return words
```

5. Dataset.

Abbiamo messo a disposizione diversi tipi di file .txt da usare come input da cui estrarre le stringhe per generare i nodi delle strutture dati. Ogni file è codificato in UTF -8 per una perfetta interpretazione dei caratteri speciali. L'utente può in qualsiasi caso usare un file .txt non presente nel CD-ROM a patto che sia codificato come richiesto. I test sono stati effettuati con diversi file presenti nella cartella `/dataset` il cui contenuto è elencato di seguito.

- /dataset
    - I Malavoglia (Verga).txt
    - Amleto (Shakespeare).txt
    - Odissea (Omero).txt
    - Il piacere (D'Annunzio).txt
    - Iliade (Omero).txt
    - Il fu Mattia Pascal (Pirandello).txt
    - Mandragola (Machiavelli).txt
    - Pinocchio (Collodi).txt
    - Don Chisciotte della Mancia (de Cervantes).txt
    - Decameron (Boccaccio).txt
    - Divina Commedia (Dante).txt
    - Mastro don Gesualdo (Verga).txt
    - Quaderni di Serafino Gubbio operatore (Pirandello).txt
    - Fermo e Lucia (Manzoni).txt

6. Tabulati dei Test.

Come abbiamo già detto, i Test sono stati effettuati su quattro diversi calcolatori[6]. Riportiamo qui di seguito i risultati, in forma schematica, dei computer utilizzati. I dati riportati di seguito sono presenti nella cartella `/tests` del CD-ROM.

6.1 Risultati di operazioni su insiemi di elementi ordinati.

1. Desktop -1.

```
File generato il 06/19/13 alle ore 21:19:23
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=1710
RICERCHE=1623
CANCELLAZIONI=1667

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.888860940933
       tempo per le ricerche       1.60009670258
       tempo per le cancellazioni  1.60332202911
       TEMPO TOTALE PER BST        4.09227967262

AVL:   tempo per gli inserimenti   0.427196979523
       tempo per le ricerche       0.0313243865967
       tempo per le cancellazioni  0.0693690776825
       TEMPO TOTALE PER AVL        0.527890443802

TEMPO TOTALE TRASCORSO:
4.62017011642

-------------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=3286
RICERCHE=3299
CANCELLAZIONI=3415

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   2.85077929497
       tempo per le ricerche       5.40872335434
       tempo per le cancellazioni  5.69899582863
       TEMPO TOTALE PER BST        13.9584984779

AVL:   tempo per gli inserimenti   0.893500566483
       tempo per le ricerche       0.0700359344482
       tempo per le cancellazioni  0.166889429092
       TEMPO TOTALE PER AVL        1.13042593002

TEMPO TOTALE TRASCORSO:
15.088924408
-------------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:
```

[6] Vedi Capitolo 4: Codice

```
INSERIMENTI=4997
RICERCHE=4928
CANCELLAZIONI=5075

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   6.15873599052
       tempo per le ricerche       11.2570543289
       tempo per le cancellazioni  11.5587978363
       TEMPO TOTALE PER BST        28.9745881557

      AVL:   tempo per gli inserimenti   1.43121266365
             tempo per le ricerche       0.111270904541
             tempo per le cancellazioni  0.280495882034
             TEMPO TOTALE PER AVL        1.82297945023

TEMPO TOTALE TRASCORSO:
30.797567606
-------------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=6624
RICERCHE=6697
CANCELLAZIONI=6679

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   10.2727982998
       tempo per le ricerche       19.4231543541
       tempo per le cancellazioni  19.4589214325
       TEMPO TOTALE PER BST        49.1548740864

AVL:   tempo per gli inserimenti   1.91970086098
       tempo per le ricerche       0.159377813339
       tempo per le cancellazioni  0.41021156311
       TEMPO TOTALE PER AVL        2.48929023743

TEMPO TOTALE TRASCORSO:
51.6441643238
-------------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=8178
RICERCHE=8387
CANCELLAZIONI=8435

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   15.0495095253
       tempo per le ricerche       28.7928407192
       tempo per le cancellazioni  29.3863613605
       TEMPO TOTALE PER BST        73.2287116051

AVL:   tempo per gli inserimenti   2.48894238472
       tempo per le ricerche       0.198714971542
       tempo per le cancellazioni  0.522057533264
       TEMPO TOTALE PER AVL        3.20971488953

TEMPO TOTALE TRASCORSO:
76.4384264946
-------------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=10064
RICERCHE=9975
CANCELLAZIONI=9961

Tempo trascorso per le operazioni diviso per tipologia:
```

```
BST:   tempo per gli inserimenti   22.6017575264
       tempo per le ricerche       41.2909817696
       tempo per le cancellazioni  42.0013020039
       TEMPO TOTALE PER BST        105.8940413

AVL:   tempo per gli inserimenti   3.0190050602
       tempo per le ricerche       0.243676185608
       tempo per le cancellazioni  0.665330648422
       TEMPO TOTALE PER AVL        3.92801189423

TEMPO TOTALE TRASCORSO:
109.822053194
-------------------------------------------------------------------------------------------

Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=11661
RICERCHE=11530
CANCELLAZIONI=11809

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   29.2067928314
       tempo per le ricerche       53.9001364708
       tempo per le cancellazioni  55.5142307281
       TEMPO TOTALE PER BST        138.62116003

AVL:   tempo per gli inserimenti   3.45122885704
       tempo per le ricerche       0.285393714905
       tempo per le cancellazioni  0.809519767761
       TEMPO TOTALE PER AVL        4.54614233971

TEMPO TOTALE TRASCORSO:
143.16730237
-------------------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=13410
RICERCHE=13292
CANCELLAZIONI=13298

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   37.5192456245
       tempo per le ricerche       69.1503460407
       tempo per le cancellazioni  69.9658062458
       TEMPO TOTALE PER BST        176.635397911

AVL:   tempo per gli inserimenti   4.10959386826
       tempo per le ricerche       0.342764616013
       tempo per le cancellazioni  1.00683188438
       TEMPO TOTALE PER AVL        5.45919036865

TEMPO TOTALE TRASCORSO:
182.09458828
-------------------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=15022
RICERCHE=15050
CANCELLAZIONI=14928

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   46.0058121681
       tempo per le ricerche       85.9357943535
       tempo per le cancellazioni  85.5517275333
       TEMPO TOTALE PER BST        217.493334055

AVL:   tempo per gli inserimenti   4.82777523994
```

```
       tempo per le ricerche       0.393766403198
       tempo per le cancellazioni  1.12201213837
       TEMPO TOTALE PER AVL        6.34355378151

TEMPO TOTALE TRASCORSO:
223.836887836
--------------------------------------------------------------------------------
Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=16600
RICERCHE=16765
CANCELLAZIONI=16635

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   53.7545936108
       tempo per le ricerche       100.437724352
       tempo per le cancellazioni  100.567257643
       TEMPO TOTALE PER BST        254.759575605

AVL:   tempo per gli inserimenti   5.10396933556
       tempo per le ricerche       0.44113278389
       tempo per le cancellazioni  1.26763558388
       TEMPO TOTALE PER AVL        6.81273770332

TEMPO TOTALE TRASCORSO:
261.572313309
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91552
ricerche:      91546
cancellazioni  91902

TEMPO TOTALE TRASCORSO
BST: 1099.13452673
AVL: 36.2699370384

TOTALE: 1135.40446377

FINE
```

---

2. Desktop -2.

```
File generato il 06/19/13 alle ore 21:52:20
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=1674
RICERCHE=1663
CANCELLAZIONI=1663

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   1.15799951553
       tempo per le ricerche       1.99200201035
       tempo per le cancellazioni  2.11799836159
       TEMPO TOTALE PER BST        5.26799988747

AVL:   tempo per gli inserimenti   0.574000597
       tempo per le ricerche       0.0490009784698
       tempo per le cancellazioni  0.0859994888306
       TEMPO TOTALE PER AVL        0.709001064301

TEMPO TOTALE TRASCORSO:
5.97700095177
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=3377
RICERCHE=3259
CANCELLAZIONI=3364

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   4.28199529648
       tempo per le ricerche       7.69200181961
       tempo per le cancellazioni  8.06199979782
       TEMPO TOTALE PER BST        20.0359969139

AVL:   tempo per gli inserimenti   1.36700105667
       tempo per le ricerche       0.0879993438721
       tempo per le cancellazioni  0.216001987457
       TEMPO TOTALE PER AVL        1.671002388

TEMPO TOTALE TRASCORSO:
21.7069993019
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=4956
RICERCHE=5065
CANCELLAZIONI=4979

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   8.5719974041
       tempo per le ricerche       16.1010093689
       tempo per le cancellazioni  16.1509971619
       TEMPO TOTALE PER BST        40.8240039349

AVL:   tempo per gli inserimenti   1.92800092697
       tempo per le ricerche       0.156997919083
       tempo per le cancellazioni  0.371999263763
       TEMPO TOTALE PER AVL        2.45699810982

TEMPO TOTALE TRASCORSO:
43.2810020447
--------------------------------------------------------------------------------
```

```
Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=6676
RICERCHE=6613
CANCELLAZIONI=6711

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   14.6159999371
       tempo per le ricerche       27.166000843
       tempo per le cancellazioni  27.5719909668
       TEMPO TOTALE PER BST        69.3539917469

AVL:   tempo per gli inserimenti   2.69500732422
       tempo per le ricerche       0.195001125336
       tempo per le cancellazioni  0.553001403809
       TEMPO TOTALE PER AVL        3.44300985336

TEMPO TOTALE TRASCORSO:
72.7970016003
-------------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=8254
RICERCHE=8375
CANCELLAZIONI=8371

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   21.5310111046
       tempo per le ricerche       40.5919926167
       tempo per le cancellazioni  40.7259964943
       TEMPO TOTALE PER BST        102.849000216

AVL:   tempo per gli inserimenti   3.57299780846
       tempo per le ricerche       0.256002426147
       tempo per le cancellazioni  0.689000606537
       TEMPO TOTALE PER AVL        4.51800084114

TEMPO TOTALE TRASCORSO:
107.367001057
-------------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=9924
RICERCHE=10031
CANCELLAZIONI=10045

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   30.2970013618
       tempo per le ricerche       56.732998848
       tempo per le cancellazioni  57.2389931679
       TEMPO TOTALE PER BST        144.268993378

AVL:   tempo per gli inserimenti   4.10600233078
       tempo per le ricerche       0.333002567291
       tempo per le cancellazioni  0.870001077652
       TEMPO TOTALE PER AVL        5.30900597572

TEMPO TOTALE TRASCORSO:
149.577999353
-------------------------------------------------------------------------------------

Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=11634
RICERCHE=11550
CANCELLAZIONI=11816
```

```
Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   38.838996172
       tempo per le ricerche       70.8720145226
       tempo per le cancellazioni  73.1229948997
       TEMPO TOTALE PER BST        182.834005594

AVL:   tempo per gli inserimenti   4.6969935894
       tempo per le ricerche       0.376999855042
       tempo per le cancellazioni  1.05300211906
       TEMPO TOTALE PER AVL        6.12699556351

TEMPO TOTALE TRASCORSO:
188.961001158
-------------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=13407
RICERCHE=13370
CANCELLAZIONI=13223

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   52.6279957294
       tempo per le ricerche       98.1040008068
       tempo per le cancellazioni  98.0160059929
       TEMPO TOTALE PER BST        248.748002529

AVL:   tempo per gli inserimenti   6.18400096893
       tempo per le ricerche       0.46999835968
       tempo per le cancellazioni  1.2909963131
       TEMPO TOTALE PER AVL        7.94499564171

TEMPO TOTALE TRASCORSO:
256.692998171
-------------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=15081
RICERCHE=14919
CANCELLAZIONI=15000

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   63.1270179749
       tempo per le ricerche       117.380983591
       tempo per le cancellazioni  118.237998009
       TEMPO TOTALE PER BST        298.745999575

AVL:   tempo per gli inserimenti   6.68300271034
       tempo per le ricerche       0.501996517181
       tempo per le cancellazioni  1.49800467491
       TEMPO TOTALE PER AVL        8.68300390244

TEMPO TOTALE TRASCORSO:
307.429003477
-------------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con stringhe ordinate:

INSERIMENTI=16682
RICERCHE=16632
CANCELLAZIONI=16686

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   75.2430019379
       tempo per le ricerche       140.537995338
       tempo per le cancellazioni  141.803004742
       TEMPO TOTALE PER BST        357.584002018
```

```
AVL:   tempo per gli inserimenti   7.20499968529
       tempo per le ricerche       0.564998626709
       tempo per le cancellazioni  1.68500065804
       TEMPO TOTALE PER AVL        9.45499897003

TEMPO TOTALE TRASCORSO:
367.039000988
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91665
ricerche:      91477
cancellazioni  91858

TEMPO TOTALE TRASCORSO
BST: 1520.95800877
AVL: 50.31701231

TOTALE: 1571.27502108

FINE
```

---

3. iMac.

```
File generato il 06/19/13 alle ore 22:57:08
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=1641
RICERCHE=1690
CANCELLAZIONI=1669

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   1.34876489639
       tempo per le ricerche       2.62223005295
       tempo per le cancellazioni  2.56643915176
       TEMPO TOTALE PER BST        6.5374341011

AVL:   tempo per gli inserimenti   0.73033952713
       tempo per le ricerche       0.0531187057495
       tempo per le cancellazioni  0.108461618423
       TEMPO TOTALE PER AVL        0.891919851303

TEMPO TOTALE TRASCORSO:
7.42935395241
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=3313
RICERCHE=3357
CANCELLAZIONI=3330

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   4.77842545509
       tempo per le ricerche       9.24874782562
       tempo per le cancellazioni  9.18525195122
       TEMPO TOTALE PER BST        23.2124252319

AVL:   tempo per gli inserimenti   1.54655742645
       tempo per le ricerche       0.117969036102
       tempo per le cancellazioni  0.271628141403
       TEMPO TOTALE PER AVL        1.93615460396

TEMPO TOTALE TRASCORSO:
25.1485798359
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=4987
RICERCHE=5020
CANCELLAZIONI=4993

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   9.93222379684
       tempo per le ricerche       19.1097488403
       tempo per le cancellazioni  18.8075070381
       TEMPO TOTALE PER BST        47.8494796753
```

```
AVL:   tempo per gli inserimenti   2.43083953857
       tempo per le ricerche       0.183656930923
       tempo per le cancellazioni  0.459859132767
       TEMPO TOTALE PER AVL        3.07435560226

TEMPO TOTALE TRASCORSO:
50.9238352776
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=6769
RICERCHE=6651
CANCELLAZIONI=6580

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   17.4803752899
       tempo per le ricerche       32.9509835243
       tempo per le cancellazioni  31.9539752007
       TEMPO TOTALE PER BST        82.3853340149

AVL:   tempo per gli inserimenti   3.33610057831
       tempo per le ricerche       0.248053073883
       tempo per le cancellazioni  0.651886463165
       TEMPO TOTALE PER AVL        4.23604011536

TEMPO TOTALE TRASCORSO:
86.6213741302
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=8232
RICERCHE=8413
CANCELLAZIONI=8355

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   24.359711647
       tempo per le ricerche       47.0036966801
       tempo per le cancellazioni  47.0054247379
       TEMPO TOTALE PER BST        118.368833065

AVL:   tempo per gli inserimenti   4.25245857239
       tempo per le ricerche       0.321151256561
       tempo per le cancellazioni  0.882473707199
       TEMPO TOTALE PER AVL        5.45608353615

TEMPO TOTALE TRASCORSO:
123.824916601
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=9820
RICERCHE=10108
CANCELLAZIONI=10072

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   33.5349199772
```

```
       tempo per le ricerche       65.2126300335
       tempo per le cancellazioni  65.010535717
       TEMPO TOTALE PER BST        163.758085728

AVL:   tempo per gli inserimenti   5.19268918037
       tempo per le ricerche       0.398014307022
       tempo per le cancellazioni  1.07148647308
       TEMPO TOTALE PER AVL        6.66218996048

TEMPO TOTALE TRASCORSO:
170.420275688
--------------------------------------------------------------------------------

Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=11673
RICERCHE=11574
CANCELLAZIONI=11753

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   45.5691947937
       tempo per le ricerche       85.2699964046
       tempo per le cancellazioni  87.0222856998
       TEMPO TOTALE PER BST        217.861476898

AVL:   tempo per gli inserimenti   5.74629092216
       tempo per le ricerche       0.460030794144
       tempo per le cancellazioni  1.33297038078
       TEMPO TOTALE PER AVL        7.53929209709

TEMPO TOTALE TRASCORSO:
225.400768995
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=13222
RICERCHE=13337
CANCELLAZIONI=13441

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   56.5239620209
       tempo per le ricerche       107.80296731
       tempo per le cancellazioni  108.440214634
       TEMPO TOTALE PER BST        272.767143965

AVL:   tempo per gli inserimenti   6.93365097046
       tempo per le ricerche       0.53636598587
       tempo per le cancellazioni  1.57670092583
       TEMPO TOTALE PER AVL        9.04671788216

TEMPO TOTALE TRASCORSO:
281.813861847
--------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=15222
RICERCHE=14835
```

```
CANCELLAZIONI=14943

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   73.7403550148
       tempo per le ricerche       135.983318806
       tempo per le cancellazioni  137.759271383
       TEMPO TOTALE PER BST        347.482945204

AVL:   tempo per gli inserimenti   8.34486937523
       tempo per le ricerche       0.602157354355
       tempo per le cancellazioni  1.86885237694
       TEMPO TOTALE PER AVL        10.8158791065

TEMPO TOTALE TRASCORSO:
358.29882431
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con
stringhe ordinate:

INSERIMENTI=16678
RICERCHE=16563
CANCELLAZIONI=16759

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   84.6734509468
       tempo per le ricerche       158.782330751
       tempo per le cancellazioni  160.489099264
       TEMPO TOTALE PER BST        403.944880962

AVL:   tempo per gli inserimenti   8.97997331619
       tempo per le ricerche       0.678458213806
       tempo per le cancellazioni  2.07889938354
       TEMPO TOTALE PER AVL        11.7373309135

TEMPO TOTALE TRASCORSO:
415.682211876
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91557
ricerche:      91548
cancellazioni  91895

TEMPO TOTALE TRASCORSO
BST: 1745.61530757
AVL: 61.3959636688

TOTALE: 1807.01127124

FINE
```

---

4. Raspberry Pi.

```
File generato il 06/19/13 alle ore 19:13:38

Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=1614
RICERCHE=1686
CANCELLAZIONI=1700

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   19.6036877632
       tempo per le ricerche       37.312615633
       tempo per le cancellazioni  38.0425665379
       TEMPO TOTALE PER BST        94.9588699341

AVL:   tempo per gli inserimenti   10.7126126289
       tempo per le ricerche       0.845395803452
       tempo per le cancellazioni  1.80764889717
       TEMPO TOTALE PER AVL        13.3656573296

TEMPO TOTALE TRASCORSO:
108.324527264
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=3370
RICERCHE=3307
CANCELLAZIONI=3323

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   75.7878296375
       tempo per le ricerche       136.656614065
       tempo per le cancellazioni  136.695060253
       TEMPO TOTALE PER BST        349.139503956

AVL:   tempo per gli inserimenti   24.2066149712
       tempo per le ricerche       1.82037949562
       tempo per le cancellazioni  4.5046453476
       TEMPO TOTALE PER AVL        30.5316398144

TEMPO TOTALE TRASCORSO:
379.67114377
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=4944
RICERCHE=5017
CANCELLAZIONI=5039

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   152.597358704
       tempo per le ricerche       282.800977707
       tempo per le cancellazioni  291.783007383
       TEMPO TOTALE PER BST        727.181343794

AVL:   tempo per gli inserimenti   39.2451124191
       tempo per le ricerche       2.92533707619
       tempo per le cancellazioni  7.34149456024
       TEMPO TOTALE PER AVL        49.5119440556

TEMPO TOTALE TRASCORSO:
```

```
776.693287849
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=6830
RICERCHE=6571
CANCELLAZIONI=6599

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   281.029872656
       tempo per le ricerche       494.307784796
       tempo per le cancellazioni  500.088979959
       TEMPO TOTALE PER BST        1275.42663741

AVL:   tempo per gli inserimenti   55.345968008
       tempo per le ricerche       3.89735245705
       tempo per le cancellazioni  10.5619575977
       TEMPO TOTALE PER AVL        69.8052780628

TEMPO TOTALE TRASCORSO:
1345.23191547
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=8274
RICERCHE=8349
CANCELLAZIONI=8377

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   373.530453444
       tempo per le ricerche       691.204578876
       tempo per le cancellazioni  696.569898844
       TEMPO TOTALE PER BST        1761.30493116

AVL:   tempo per gli inserimenti   62.0198259354
       tempo per le ricerche       4.98263573647
       tempo per le cancellazioni  14.0655994415
       TEMPO TOTALE PER AVL        81.0680611134

TEMPO TOTALE TRASCORSO:
1842.37299228
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=10028
RICERCHE=9956
CANCELLAZIONI=10016

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   540.654002666
       tempo per le ricerche       990.721424818
       tempo per le cancellazioni  994.993954182
       TEMPO TOTALE PER BST        2526.36938167

AVL:   tempo per gli inserimenti   78.7088110447
       tempo per le ricerche       6.08257293701
       tempo per le cancellazioni  17.3051996231
       TEMPO TOTALE PER AVL        102.096583605

TEMPO TOTALE TRASCORSO:
2628.46596527
--------------------------------------------------------------------------------
```

```
Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=11588
RICERCHE=11756
CANCELLAZIONI=11656

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   702.736054659
       tempo per le ricerche       1315.56823373
       tempo per le cancellazioni  1310.52786851
       TEMPO TOTALE PER BST        3328.8321569

AVL:   tempo per gli inserimenti   90.3953123093
       tempo per le ricerche       7.35026431084
       tempo per le cancellazioni  20.5477883816
       TEMPO TOTALE PER AVL        118.293365002

TEMPO TOTALE TRASCORSO:
3447.1255219
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=13361
RICERCHE=13376
CANCELLAZIONI=13263

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   877.779755831
       tempo per le ricerche       1621.53695107
       tempo per le cancellazioni  1609.17467332
       TEMPO TOTALE PER BST        4108.49138021

AVL:   tempo per gli inserimenti   101.341372728
       tempo per le ricerche       8.33154845238
       tempo per le cancellazioni  24.3900997639
       TEMPO TOTALE PER AVL        134.063020945

TEMPO TOTALE TRASCORSO:
4242.55440116
--------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:

INSERIMENTI=14961
RICERCHE=15131
CANCELLAZIONI=14908

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   1061.96614099
       tempo per le ricerche       1960.32452059
       tempo per le cancellazioni  1938.84386992
       TEMPO TOTALE PER BST        4961.1345315

AVL:   tempo per gli inserimenti   115.033508778
       tempo per le ricerche       9.47476625443
       tempo per le cancellazioni  27.9560482502
       TEMPO TOTALE PER AVL        152.464323282

TEMPO TOTALE TRASCORSO:
5113.59885478
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con stringhe
ordinate:
```

```
INSERIMENTI=16614
RICERCHE=16624
CANCELLAZIONI=16762

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   1295.39036965
       tempo per le ricerche       2390.19544649
       tempo per le cancellazioni  2408.92757297
       TEMPO TOTALE PER BST        6094.51338911

AVL:   tempo per gli inserimenti   126.466065407
       tempo per le ricerche       10.5956315994
       tempo per le cancellazioni  31.2439978123
       TEMPO TOTALE PER AVL        168.305694818

TEMPO TOTALE TRASCORSO:
6262.81908393
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91584
ricerche:      91773
cancellazioni  91643

TEMPO TOTALE TRASCORSO
BST: 26148.3296146
AVL: 919.505568027

TOTALE: 27067.8351827

FINE
```

6.2 Risultati di operazioni su insiemi di elementi non ordinati.

1. Desktop -1.

```
File generato il 06/19/13 alle ore 21:39:29
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe
non ordinate:

INSERIMENTI=1661
RICERCHE=1673
CANCELLAZIONI=1666

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.026082277298
       tempo per le ricerche       0.038999080658
       tempo per le cancellazioni  0.0455665588379
       TEMPO TOTALE PER BST        0.110647916794

AVL:   tempo per gli inserimenti   0.18633890152
       tempo per le ricerche       0.0264658927917
       tempo per le cancellazioni  0.0757460594177
       TEMPO TOTALE PER AVL        0.288550853729

TEMPO TOTALE TRASCORSO:
0.399198770523
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=3342
RICERCHE=3314
CANCELLAZIONI=3344

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.0587809085846
       tempo per le ricerche       0.0812451839447
       tempo per le cancellazioni  0.100658178329
       TEMPO TOTALE PER BST        0.240684270859

AVL:   tempo per gli inserimenti   0.331712007523
       tempo per le ricerche       0.0561022758484
       tempo per le cancellazioni  0.18089056015
       TEMPO TOTALE PER AVL        0.568704843521

TEMPO TOTALE TRASCORSO:
0.80938911438
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=4915
RICERCHE=5057
CANCELLAZIONI=5028

Tempo trascorso per le operazioni diviso per tipologia:
```

```
BST:   tempo per gli inserimenti   0.0853183269501
       tempo per le ricerche       0.123928308487
       tempo per le cancellazioni  0.151346206665
       TEMPO TOTALE PER BST        0.360592842102

AVL:   tempo per gli inserimenti   0.464520215988
       tempo per le ricerche       0.0908203125
       tempo per le cancellazioni  0.308387517929
       TEMPO TOTALE PER AVL        0.863728046417

TEMPO TOTALE TRASCORSO:
1.22432088852
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=6709
RICERCHE=6567
CANCELLAZIONI=6724

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.151336431503
       tempo per le ricerche       0.202478408813
       tempo per le cancellazioni  0.250075340271
       TEMPO TOTALE PER BST        0.603890180588

AVL:   tempo per gli inserimenti   0.957918167114
       tempo per le ricerche       0.123527288437
       tempo per le cancellazioni  0.576816797256
       TEMPO TOTALE PER AVL        1.65826225281

TEMPO TOTALE TRASCORSO:
2.2621524334
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=8363
RICERCHE=8307
CANCELLAZIONI=8330

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.188325881958
       tempo per le ricerche       0.254590034485
       tempo per le cancellazioni  0.310108423233
       TEMPO TOTALE PER BST        0.753024339676

AVL:   tempo per gli inserimenti   1.15476465225
       tempo per le ricerche       0.157553911209
       tempo per le cancellazioni  0.875626087189
       TEMPO TOTALE PER AVL        2.18794465065

TEMPO TOTALE TRASCORSO:
2.94096899033
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=10046
```

```
RICERCHE=9977
CANCELLAZIONI=9977

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.200725078583
       tempo per le ricerche       0.269531011581
       tempo per le cancellazioni  0.334760427475
       TEMPO TOTALE PER BST        0.805016517639

AVL:   tempo per gli inserimenti   1.0409822464
       tempo per le ricerche       0.193828105927
       tempo per le cancellazioni  0.712124347687
       TEMPO TOTALE PER AVL        1.94693470001

TEMPO TOTALE TRASCORSO:
2.75195121765
--------------------------------------------------------------------------------

Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=11659
RICERCHE=11694
CANCELLAZIONI=11647

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.268592834473
       tempo per le ricerche       0.374920606613
       tempo per le cancellazioni  0.452135801315
       TEMPO TOTALE PER BST        1.0956492424

AVL:   tempo per gli inserimenti   1.39989900589
       tempo per le ricerche       0.233494281769
       tempo per le cancellazioni  0.9056828022
       TEMPO TOTALE PER AVL        2.53907608986

TEMPO TOTALE TRASCORSO:
3.63472533226
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=13395
RICERCHE=13281
CANCELLAZIONI=13324

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.349720478058
       tempo per le ricerche       0.460337400436
       tempo per le cancellazioni  0.555624723434
       TEMPO TOTALE PER BST        1.36568260193

AVL:   tempo per gli inserimenti   1.59996747971
       tempo per le ricerche       0.26394200325
       tempo per le cancellazioni  1.08294701576
       TEMPO TOTALE PER AVL        2.94685649872

TEMPO TOTALE TRASCORSO:
4.31253910065
--------------------------------------------------------------------------------
```

```
Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=14887
RICERCHE=14926
CANCELLAZIONI=15187

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.336263656616
       tempo per le ricerche       0.450726509094
       tempo per le cancellazioni  0.564546346664
       TEMPO TOTALE PER BST        1.35153651237

AVL:   tempo per gli inserimenti   1.91893577576
       tempo per le ricerche       0.307543516159
       tempo per le cancellazioni  1.76539778709
       TEMPO TOTALE PER AVL        3.99187707901

TEMPO TOTALE TRASCORSO:
5.34341359138
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=16702
RICERCHE=16658
CANCELLAZIONI=16640

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.439868688583
       tempo per le ricerche       0.579124212265
       tempo per le cancellazioni  0.696624040604
       TEMPO TOTALE PER BST        1.71561694145

AVL:   tempo per gli inserimenti   2.41637802124
       tempo per le ricerche       0.342080831528
       tempo per le cancellazioni  1.89637875557
       TEMPO TOTALE PER AVL        4.65483760834

TEMPO TOTALE TRASCORSO:
6.37045454979
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91679
ricerche:      91454
cancellazioni  91867

TEMPO TOTALE TRASCORSO
BST: 30.0757842064
AVL: 21.6467726231

TOTALE: 51.7225568295

FINE
```

---

2. Desktop -2

```
File generato il 06/19/13 alle ore 22:18:21
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=1629
RICERCHE=1723
CANCELLAZIONI=1648

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.0399994850159
       tempo per le ricerche       0.0480003356934
       tempo per le cancellazioni  0.0720012187958
       TEMPO TOTALE PER BST        0.160001039505

AVL:   tempo per gli inserimenti   0.171000242233
       tempo per le ricerche       0.0539994239807
       tempo per le cancellazioni  0.089998960495
       TEMPO TOTALE PER AVL        0.314998626709

TEMPO TOTALE TRASCORSO:
0.474999666214
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=3376
RICERCHE=3256
CANCELLAZIONI=3368

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.0859994888306
       tempo per le ricerche       0.113002538681
       tempo per le cancellazioni  0.144999504089
       TEMPO TOTALE PER BST        0.344001531601

AVL:   tempo per gli inserimenti   0.466000080109
       tempo per le ricerche       0.0669994354248
       tempo per le cancellazioni  0.341999292374
       TEMPO TOTALE PER AVL        0.874998807907

TEMPO TOTALE TRASCORSO:
1.21900033951
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=4980
RICERCHE=5013
CANCELLAZIONI=5007

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.135997772217
       tempo per le ricerche       0.188001871109
       tempo per le cancellazioni  0.218000173569
       TEMPO TOTALE PER BST        0.541999816895

AVL:   tempo per gli inserimenti   0.704002141953
       tempo per le ricerche       0.13100028038
       tempo per le cancellazioni  0.499997854233
       TEMPO TOTALE PER AVL        1.33500027657

TEMPO TOTALE TRASCORSO:
```

```
1.87700009346
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=6622
RICERCHE=6595
CANCELLAZIONI=6783

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.201999425888
       tempo per le ricerche       0.257997989655
       tempo per le cancellazioni  0.34600019455
       TEMPO TOTALE PER BST        0.805997610092

AVL:   tempo per gli inserimenti   1.06800436974
       tempo per le ricerche       0.16700053215
       tempo per le cancellazioni  0.629997014999
       TEMPO TOTALE PER AVL        1.86500191689

TEMPO TOTALE TRASCORSO:
2.67099952698
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=8362
RICERCHE=8217
CANCELLAZIONI=8421

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.230000019073
       tempo per le ricerche       0.30699801445
       tempo per le cancellazioni  0.443002700806
       TEMPO TOTALE PER BST        0.980000734329

AVL:   tempo per gli inserimenti   1.13000202179
       tempo per le ricerche       0.197999715805
       tempo per le cancellazioni  0.817998886108
       TEMPO TOTALE PER AVL        2.1460006237

TEMPO TOTALE TRASCORSO:
3.12600135803
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=10073
RICERCHE=9955
CANCELLAZIONI=9972

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.358999490738
       tempo per le ricerche       0.502996683121
       tempo per le cancellazioni  0.561004400253
       TEMPO TOTALE PER BST        1.42300057411

AVL:   tempo per gli inserimenti   1.63400053978
       tempo per le ricerche       0.274002790451
       tempo per le cancellazioni  1.05399608612
       TEMPO TOTALE PER AVL        2.96199941635

TEMPO TOTALE TRASCORSO:
4.38499999046
--------------------------------------------------------------------------------
```

```
Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=11754
RICERCHE=11544
CANCELLAZIONI=11702

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.38399720192
       tempo per le ricerche       0.478002548218
       tempo per le cancellazioni  0.559002161026
       TEMPO TOTALE PER BST        1.42100191116

AVL:   tempo per gli inserimenti   2.72999835014
       tempo per le ricerche       0.303997755051
       tempo per le cancellazioni  2.5680000782
       TEMPO TOTALE PER AVL        5.6019961834

TEMPO TOTALE TRASCORSO:
7.02299809456
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=13251
RICERCHE=13400
CANCELLAZIONI=13349

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.436000823975
       tempo per le ricerche       0.5870013237
       tempo per le cancellazioni  0.729994535446
       TEMPO TOTALE PER BST        1.75299668312

AVL:   tempo per gli inserimenti   2.22100162506
       tempo per le ricerche       0.377000808716
       tempo per le cancellazioni  1.79699754715
       TEMPO TOTALE PER AVL        4.39499998093

TEMPO TOTALE TRASCORSO:
6.14799666405
--------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=15115
RICERCHE=15034
CANCELLAZIONI=14851

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.486001014709
       tempo per le ricerche       0.63100528717
       tempo per le cancellazioni  0.758998155594
       TEMPO TOTALE PER BST        1.87600445747

AVL:   tempo per gli inserimenti   2.47099590302
       tempo per le ricerche       0.419003009796
       tempo per le cancellazioni  1.820997715
       TEMPO TOTALE PER AVL        4.71099662781

TEMPO TOTALE TRASCORSO:
6.58700108528
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:
```

```
INSERIMENTI=16851
RICERCHE=16353
CANCELLAZIONI=16796

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.528001308441
       tempo per le ricerche       0.650999069214
       tempo per le cancellazioni  0.830002307892
       TEMPO TOTALE PER BST        2.00900268555

AVL:   tempo per gli inserimenti   2.77999711037
       tempo per le ricerche       0.466000080109
       tempo per le cancellazioni  1.9089987278
       TEMPO TOTALE PER AVL        5.15499591827

TEMPO TOTALE TRASCORSO:
7.16399860382
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   92013
ricerche:      91090
cancellazioni  91897

TEMPO TOTALE TRASCORSO
BST: 40.7349948883
AVL: 29.3609883785

TOTALE: 70.0959832668

FINE
```

---

3. iMac.

```
File generato il 06/19/13 alle ore 23:27:33
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe
non ordinate:

INSERIMENTI=1687
RICERCHE=1653
CANCELLAZIONI=1660

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.0479440689087
       tempo per le ricerche       0.0666153430939
       tempo per le cancellazioni  0.0784287452698
       TEMPO TOTALE PER BST        0.192988157272

AVL:   tempo per gli inserimenti   0.324508428574
       tempo per le ricerche       0.044007062912
       tempo per le cancellazioni  0.135003328323
       TEMPO TOTALE PER AVL        0.503518819809

TEMPO TOTALE TRASCORSO:
0.696506977081
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=3276
RICERCHE=3351
CANCELLAZIONI=3373

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.0904457569122
       tempo per le ricerche       0.132943868637
       tempo per le cancellazioni  0.163774013519
       TEMPO TOTALE PER BST        0.387163639069

AVL:   tempo per gli inserimenti   0.668268918991
       tempo per le ricerche       0.0956935882568
       tempo per le cancellazioni  0.355050563812
       TEMPO TOTALE PER AVL        1.11901307106

TEMPO TOTALE TRASCORSO:
1.50617671013
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=4795
RICERCHE=5014
CANCELLAZIONI=5191

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.142432689667
       tempo per le ricerche       0.215356111526
       tempo per le cancellazioni  0.266205072403
       TEMPO TOTALE PER BST        0.623993873596
```

```
AVL:   tempo per gli inserimenti   1.03442001343
       tempo per le ricerche       0.153985977173
       tempo per le cancellazioni  0.835256814957
       TEMPO TOTALE PER AVL        2.02366280556

TEMPO TOTALE TRASCORSO:
2.64765667915
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=6617
RICERCHE=6695
CANCELLAZIONI=6688

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.21219587326
       tempo per le ricerche       0.306666851044
       tempo per le cancellazioni  0.371451854706
       TEMPO TOTALE PER BST        0.89031457901

AVL:   tempo per gli inserimenti   1.6050388813
       tempo per le ricerche       0.210651874542
       tempo per le cancellazioni  0.945606708527
       TEMPO TOTALE PER AVL        2.76129746437

TEMPO TOTALE TRASCORSO:
3.65161204338
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=8359
RICERCHE=8353
CANCELLAZIONI=8288

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.268831253052
       tempo per le ricerche       0.366027593613
       tempo per le cancellazioni  0.450886249542
       TEMPO TOTALE PER BST        1.08574509621

AVL:   tempo per gli inserimenti   2.63546133041
       tempo per le ricerche       0.263358831406
       tempo per le cancellazioni  1.55207180977
       TEMPO TOTALE PER AVL        4.45089197159

TEMPO TOTALE TRASCORSO:
5.53663706779
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=10001
RICERCHE=10001
CANCELLAZIONI=9998

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.364314317703
```

```
       tempo per le ricerche       0.515104055405
       tempo per le cancellazioni  0.619300365448
       TEMPO TOTALE PER BST        1.49871873856

AVL:   tempo per gli inserimenti   2.52196359634
       tempo per le ricerche       0.319445371628
       tempo per le cancellazioni  2.94174790382
       TEMPO TOTALE PER AVL        5.7831568718

TEMPO TOTALE TRASCORSO:
7.28187561035
--------------------------------------------------------------------------------

Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=11715
RICERCHE=11606
CANCELLAZIONI=11679

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.41709113121
       tempo per le ricerche       0.551805973053
       tempo per le cancellazioni  0.678348779678
       TEMPO TOTALE PER BST        1.64724588394

AVL:   tempo per gli inserimenti   2.45777916908
       tempo per le ricerche       0.376486301422
       tempo per le cancellazioni  1.88623666763
       TEMPO TOTALE PER AVL        4.72050213814

TEMPO TOTALE TRASCORSO:
6.36774802208
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=13344
RICERCHE=13351
CANCELLAZIONI=13305

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.487620353699
       tempo per le ricerche       0.671265602112
       tempo per le cancellazioni  0.822376489639
       TEMPO TOTALE PER BST        1.98126244545

AVL:   tempo per gli inserimenti   3.51912021637
       tempo per le ricerche       0.437371969223
       tempo per le cancellazioni  4.84261918068
       TEMPO TOTALE PER AVL        8.79911136627

TEMPO TOTALE TRASCORSO:
10.7803738117
--------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=14912
RICERCHE=14971
```

```
CANCELLAZIONI=15117

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.620364665985
       tempo per le ricerche       0.867718935013
       tempo per le cancellazioni  1.04515480995
       TEMPO TOTALE PER BST        2.53323841095

AVL:   tempo per gli inserimenti   2.96067619324
       tempo per le ricerche       0.494936227798
       tempo per le cancellazioni  2.13369321823
       TEMPO TOTALE PER AVL        5.58930563927

TEMPO TOTALE TRASCORSO:
8.12254405022
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con
stringhe non ordinate:

INSERIMENTI=16597
RICERCHE=16756
CANCELLAZIONI=16647

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.694262504578
       tempo per le ricerche       0.851787567139
       tempo per le cancellazioni  1.04647517204
       TEMPO TOTALE PER BST        2.59252524376

AVL:   tempo per gli inserimenti   3.76162004471
       tempo per le ricerche       0.55043721199
       tempo per le cancellazioni  3.06728005409
       TEMPO TOTALE PER AVL        7.37933731079

TEMPO TOTALE TRASCORSO:
9.97186255455
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91303
ricerche:      91751
cancellazioni  91946

TEMPO TOTALE TRASCORSO
BST: 56.5936880112
AVL: 43.1297974586

TOTALE: 99.7234854698

FINE
```

---

4. Raspberry Pi.

```
File generato il 06/20/13 alle ore 09:20:15
```

```
Statistiche per 5000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=1678
RICERCHE=1625
CANCELLAZIONI=1697

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   0.70515871048
       tempo per le ricerche       0.956351995468
       tempo per le cancellazioni  1.1630449295
       TEMPO TOTALE PER BST        2.82455563545

AVL:   tempo per gli inserimenti   4.36786007881
       tempo per le ricerche       0.684838056564
       tempo per le cancellazioni  1.97605466843
       TEMPO TOTALE PER AVL        7.0287528038

TEMPO TOTALE TRASCORSO:
9.85330843925
--------------------------------------------------------------------------------

Statistiche per 10000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=3302
RICERCHE=3362
CANCELLAZIONI=3336

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   1.55671787262
       tempo per le ricerche       2.20345568657
       tempo per le cancellazioni  2.56227970123
       TEMPO TOTALE PER BST        6.32245326042

AVL:   tempo per gli inserimenti   12.291960001
       tempo per le ricerche       1.4973654747
       tempo per le cancellazioni  7.74619817734
       TEMPO TOTALE PER AVL        21.535523653

TEMPO TOTALE TRASCORSO:
27.8579769135
--------------------------------------------------------------------------------

Statistiche per 15000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=4901
RICERCHE=5019
CANCELLAZIONI=5080

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   2.61323332787
       tempo per le ricerche       3.72511935234
       tempo per le cancellazioni  4.45440840721
       TEMPO TOTALE PER BST        10.7927610874

AVL:   tempo per gli inserimenti   17.3242025375
       tempo per le ricerche       2.32865023613
       tempo per le cancellazioni  13.9479913712
       TEMPO TOTALE PER AVL        33.6008441448

TEMPO TOTALE TRASCORSO:
```

```
44.3936052322
--------------------------------------------------------------------------------

Statistiche per 20000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=6608
RICERCHE=6790
CANCELLAZIONI=6602

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   3.37635564804
       tempo per le ricerche       4.58548212051
       tempo per le cancellazioni  5.47054243088
       TEMPO TOTALE PER BST        13.4323801994

AVL:   tempo per gli inserimenti   17.3291575909
       tempo per le ricerche       3.17394804955
       tempo per le cancellazioni  11.1990544796
       TEMPO TOTALE PER AVL        31.70216012

TEMPO TOTALE TRASCORSO:
45.1345403194
--------------------------------------------------------------------------------

Statistiche per 25000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=8325
RICERCHE=8288
CANCELLAZIONI=8387

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   4.06855225563
       tempo per le ricerche       5.09976005554
       tempo per le cancellazioni  6.43122291565
       TEMPO TOTALE PER BST        15.5995352268

AVL:   tempo per gli inserimenti   32.9511630535
       tempo per le ricerche       3.98359155655
       tempo per le cancellazioni  19.5522117615
       TEMPO TOTALE PER AVL        56.4869663715

TEMPO TOTALE TRASCORSO:
72.0865015984
--------------------------------------------------------------------------------

Statistiche per 30000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=10101
RICERCHE=9972
CANCELLAZIONI=9927

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   5.53723978996
       tempo per le ricerche       6.98663234711
       tempo per le cancellazioni  8.48095726967
       TEMPO TOTALE PER BST        21.0048294067

AVL:   tempo per gli inserimenti   31.8535912037
       tempo per le ricerche       4.89355778694
       tempo per le cancellazioni  19.8632986546
       TEMPO TOTALE PER AVL        56.6104476452

TEMPO TOTALE TRASCORSO:
77.6152770519
--------------------------------------------------------------------------------
```

```
Statistiche per 35000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=11688
RICERCHE=11570
CANCELLAZIONI=11742

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   7.06105518341
       tempo per le ricerche       8.65883827209
       tempo per le cancellazioni  10.7756252289
       TEMPO TOTALE PER BST        26.4955186844

AVL:   tempo per gli inserimenti   39.0162312984
       tempo per le ricerche       5.74981069565
       tempo per le cancellazioni  34.728028059
       TEMPO TOTALE PER AVL        79.4940700531

TEMPO TOTALE TRASCORSO:
105.989588737
--------------------------------------------------------------------------------

Statistiche per 40000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=13284
RICERCHE=13366
CANCELLAZIONI=13350

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   7.67125201225
       tempo per le ricerche       9.39423418045
       tempo per le cancellazioni  11.6780121326
       TEMPO TOTALE PER BST        28.7434983253

AVL:   tempo per gli inserimenti   38.6664233208
       tempo per le ricerche       6.80503964424
       tempo per le cancellazioni  29.3626573086
       TEMPO TOTALE PER AVL        74.8341202736

TEMPO TOTALE TRASCORSO:
103.577618599
--------------------------------------------------------------------------------

Statistiche per 45000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:

INSERIMENTI=15028
RICERCHE=14992
CANCELLAZIONI=14980

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   8.67725253105
       tempo per le ricerche       11.0476605892
       tempo per le cancellazioni  13.5728983879
       TEMPO TOTALE PER BST        33.2978115082

AVL:   tempo per gli inserimenti   49.6281151772
       tempo per le ricerche       7.64054250717
       tempo per le cancellazioni  34.108453989
       TEMPO TOTALE PER AVL        91.3771116734

TEMPO TOTALE TRASCORSO:
124.674923182
--------------------------------------------------------------------------------

Statistiche per 50000 operazioni random su alberi AVL e BST vuoti con stringhe non
ordinate:
```

```
INSERIMENTI=16754
RICERCHE=16512
CANCELLAZIONI=16734

Tempo trascorso per le operazioni diviso per tipologia:
BST:   tempo per gli inserimenti   10.5026488304
       tempo per le ricerche       12.8385474682
       tempo per le cancellazioni  15.8494715691
       TEMPO TOTALE PER BST        39.1906678677

AVL:   tempo per gli inserimenti   65.5314919949
       tempo per le ricerche       8.58887934685
       tempo per le cancellazioni  68.6007099152
       TEMPO TOTALE PER AVL        142.721081257

TEMPO TOTALE TRASCORSO:
181.911749125
```

```
NUMERO TOTALE DI OPERAZIONI SVOLTE:
inserimenti:   91669
ricerche:      91496
cancellazioni  91835

TEMPO TOTALE TRASCORSO
BST: 793.916772127
AVL: 595.391077995

TOTALE: 1389.30785012

FINE
```

---

7. Struttura del CD-ROM.

- Progetto_4.py
- /src
    - _init_.py
    - AVLTree.py
    - BinarySearchTree.py
    - extraFunctions.py
    - Nodes.py
    - TxtInput.py
- /dataset
    - I Malavoglia (Verga).txt
    - Amleto (Shakespeare).txt
    - Odissea (Omero).txt
    - Il piacere (D'Annunzio).txt
    - Iliade (Omero).txt
    - Il fu Mattia Pascal (Pirandello).txt
    - Mandragola (Machiavelli).txt
    - Pinocchio (Collodi).txt
    - Don Chisciotte della Mancia (de Cervantes).txt
    - Decameron (Boccaccio).txt
    - Divina Commedia (Dante).txt
    - Mastro don Gesualdo (Verga).txt
    - Quaderni di Serafino Gubbio operatore (Pirandello).txt
    - Fermo e Lucia (Manzoni).txt
- /tests
    - Piattaforme di test.pdf
    - /Operazioni su insiemi non ordinati
        - Desktop-1.pdf
        - Desktop-2.pdf
        - iMac.pdf
        - Raspberry Pi.pdf
    - /Operazioni su insiemi ordinati
        - Desktop-1.pdf
        - Desktop-2.pdf
        - iMac.pdf
        - Raspberry Pi.pdf
- /report
    - Relazione_progetto_4.pdf