Nome: Carmine
Cognome: Monaco
Matricola: S271339

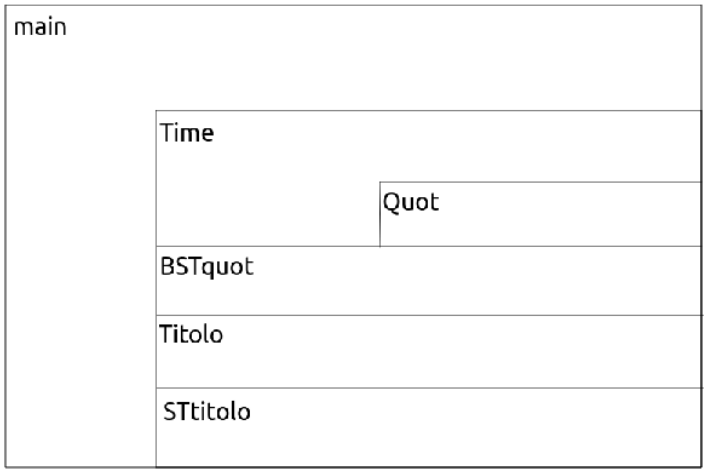
Laboratorio 12

Struttura dati:

-Time è un quasi ADT
-Quot è un quasi ADT, non viene allocato con malloc

Time e quot sono composti per valore, non allocati dinamicamente
-BSTquot è un ADT di I classe
-Titolo è un ADT di I classe, allocato con malloc
-STTitolo è un ADT di I classe

Time e Quot sono composti per valore e non allocati dinamicamente.



Struttura dati	Chiave	Item
BSTquot	Time	Quot
STtitolo	Stringa (char name[T_NAME_SIZE])	Titolo

Quot è l'item client di BSTquot, esso contiene la chiave usata da BSTquot accessibile tramite funzione Time Q_KEYget(Quot quot).
Titolo è l'item client di Sttitolo, esso contiene la chiave usata da Sttitlo accessibile tramite funzione char* T_KEYget(Titolo titolo);

Time prima di essere confrontata viene trasformata ad intero con la funzione int TIMEget(), che viene usata in TIMEcmp(Time time1, Time time2);
Il titolo invece usa la strcmp(char* str1,char* str2) da <string.h> per confrontare 2 stringhe

Scelte algoritmiche:

Bstquot:
Ho esteso il BST con 3 funzioni visibili dai client: **BSTupdatequot**, **BSTsearchLocalMax** **BSTsearchAbsoluteMax**.

BSTupdatequot() viene usata per aggiornare il valore dell'item del BST (Quotazione giornaliera) leggendo le nuove transazioni di azioni da file. La funzione è una BSTsearch che al successo chiama QUOTupdate passando l'indirizzo della quotazione giornaliera dal BST per aggiornare i dati. La parte di search avrà sempre successo perchè prima di chiamare questa funzione viene controllato se la Quot esiste, e viene creata in caso contrario.

Non ho usato Quot BSTsearch perchè per modificare la Quot salvata nel BST serviva una funzione Quot* BSTsearch, che ho deciso di non implementare

Complessità: come BSTsearch, BST bilanciato: O(logn), BST sbilanciato O(n) | guardo QUOTupdate come O(1), perde tempo nella fscanf del file, ma effettua operazioni O(1) come addizioni.

BSTsearchLocalMax() è un algoritmo di ricorsione che percorre I nodi inOrder. Cerca il Quot con valore massimo controllando tutte le Quot dentro il BST, però controllando se siano idonei verificando che le loro date siano comprese tra data 1 e data 2.
Complessità: sempre O(n)

BSTsearchAbsoluteMax() un caso particolare della funzione BSTsearchLocalMax: cerco le Quot con data meno recente e piu' recente con minR e maxR rispettivamente (meno recente => valore Key minore => sarà minR) ed applico la BSTsearchLocalMax con questi 2 estremi.
Complessità: sempre O(n)

STtitolo:
contiene una semplice lista ordinata per nome ascendente, con la STinsert che gestisce l'ordinamento.

Time:
contiene 2 funzioni read: TIMEread(FILE *fp, Time *time) che legge data e l'orario e viene usata per la lettura del file, e TIMEreadsimple(FILE *fp, Time *time) che legge solo la data e viene usata quando il client chiede informazioni per svolgere un compito.