1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

CORSO DI LAUREA IN
INFORMATICA APPLICATA
SCUOLA DI
SCIENZE TECNOLOGIE E FILOSOFIA DELL'INFORMAZIONE

RELAZIONE PROGETTO

ALGORITMI E STRUTTURE DATI

Sessione Estiva 2020/2021

Studente: Simone Cossi
Nr. Matricola: 290796
Email: s.cossi2@campus.uniurb.it

Studente: Martina Breccia
Nr. Matricola: 292514
Email: m.breccia3@campus.uniurb.it

1. Specifica del problema

Si supponga di dover progettare un programma per un sistema informativo che gestisce i dati relativi ad un'anagrafe di autoveicoli. Il sistema permette di mantenere, per ogni veicolo, questi dati:

- veicolo: codice univoco alfanumerico rappresentato da una stringa concatenazione di quattro lettere e due numeri;
- proprietario: codice rappresentato da una stringa concatenazione di 3 lettere e tre numeri;
- modello del veicolo: stringa di massimo 20 caratteri;
- anno di immatricolazione: un numero intero.

Le informazioni sono memorizzate su un file in formato testo, secondo il seguente formato (si assumano campi separati da tabulazione o spazio):

Ad esempio:

Veicolo	Proprietario	Modello	Anno
ZZYQ48	FCA001	Fiat 850	1973
CCGH07	DEF130	Fiat 500	1971
AAAB12	ABC001	Ford T	1908
…	…	…	…

Si scriva un programma ANSI C che esegue le seguenti elaborazioni:

1. Acquisisce il file e memorizza le relative informazioni in una struttura dati di tipo dinamico.
2. Ricerca e restituisce i dati relativi ad un dato veicolo. Ad esempio: se l'utente chiede i dati relativi al veicolo CCGH07, il programma deve restituire le informazioni contenute nella riga corrispondente:

Veicolo Proprietario Modello Anno

CCGH07 DEF130 Fiat 500 1971

3. Inserimento di dati relativi ad un nuovo veicolo, specificati dall'utente.
4. Rimozione di dati relativi ad un veicolo, specificati dall'utente.
5. Calcolo del valore più grande e di quello più piccolo (secondo l'ordine lessicografico) del codice alfanumerico rappresentante il veicolo (AAAB12 e ZZYQ48, rispettivamente, nell'esempio riportato).

Per quanto riguarda l'analisi teorica si deve fornire la complessità:

- dell'algoritmo di ricerca
- dell'algoritmo di inserimento
- dell'algoritmo di rimozione
- dell'algoritmo di calcolo del valore più grande
- dell'algoritmo di calcolo del valore più piccolo

Oltre all'analisi teorica della complessità si deve effettuare uno studio sperimentale della stessa per le operazioni sopra specificate (ricerca, inserimento, rimozione, calcolo del massimo, calcolo del minimo). Come suggerimento si può operare generando un numero N di dati-veicolo casuali (dove N rappresenta il numero di veicoli). L'analisi sperimentale deve quindi valutare la complessità al variare di N.

2. Analisi del problema

Input

L'input è composto da un file di testo contenente le informazioni riguardanti dei veicoli e da alcuni comandi impartiti dall'utente nella fase di generazione del file e nella fase di esecuzione del programma.

I comandi che l'utente può impartire sono:

- Inserire il numero dei veicoli da sfruttare per l'esecuzione del programma
- Scegliere di cercare un veicolo di cui si desidera leggere tutte le informazioni
- Scegliere di inserire un nuovo veicolo con tutte le sue informazioni:
    - Id
    - Proprietario
    - Modello
    - Anno
- Scegliere di rimuovere un veicolo e tutte le sue informazioni
- Scegliere di calcolare e visualizzare il valore più grande e il valore più piccolo secondo l'ordine lessicografico

Output

L'output è composto da: dati che il generatore andrà a scrivere su file, da stampe su file e a video che il programma produrrà in fase di esecuzione, tra cui:

- Indicazioni su come e cosa l'utente deve inserire per interagire con il programma
- Stampa a video di una lista contenente tutti gli id contenuti nel file
- Stampa di tutte le informazioni del veicolo scelto dall'utente
- Ristampa a video di una lista contenente tutti gli id aggiornati contenuti nel file e ristampa del file di tutte le informazioni incluse quelle aggiornate
- Stampa a video del valore più grande e più piccolo secondo l'ordine lessicografico
- Stampa a video del numero di passi dei vari algoritmi utilizzati durante l'esecuzione
- Avvisi di eventuali errori di incompatibilità tra le scelte possibili e quelle effettuate dall'utente
- Avvisi di eventuali errori avvenuti durante l'apertura del file contenente i dati

Relazioni input – output

Partendo dal generatore, quest'ultimo genera e sovrascrive (se già dovessero essere presenti) dati riguardanti un numero N di veicoli scelti dall'utente su di un file.

Il programma, invece, utilizzando il file mostrerà all'utente una lista contenente tutti gli id dei veicoli, per poi andare a:

- Mostrare le informazioni riguardanti uno o più veicoli scelti
- Inserire un nuovo veicolo scelto dall'utente
- Rimuovere un veicolo scelto dall'utente
- Visualizzare il valore più grande e quello più piccolo, secondo l'ordine lessicografico, contenuto nel file

L'utente è dunque responsabile di scegliere:

- Il numero di veicoli di cui generare e visualizzare le informazioni
- Il veicolo di cui si vogliono leggere le informazioni
- Le informazioni riguardanti un nuovo veicolo
- Il veicolo che si vuole rimuovere
- Se visualizzare o meno il valore più grande e più piccolo
- Terminare il programma

3. Progettazione dell'algoritmo

Strutture dati

La struttura utilizzata è una di quelle previste dalla specifica del progetto. La specifica richiedeva di utilizzare una struttura dati di tipo dinamica (liste, alberi, grafi).

Abbiamo scelto di utilizzare una struttura dati di tipo albero binario di ricerca, poiché andando a confrontare gli alberi con le liste, abbiamo riscontrato che negli algoritmi di ricerca, inserimento e rimozione, la complessità nel caso ottimo e nel caso pessimo sono uguali. Nel caso ottimo troviamo in entrambi i casi la complessità pari a $T(n) = O(1)$, mentre nel caso pessimo la complessità è $T(n) = O(n)$. *M*entre nel caso medio l'albero essendo una struttura ramificata e non lineare come la lista, permette di avere una minore complessità rispetto alla lista.

Non è stato scelto di utilizzare un grafo più complesso (l'albero rientra in un caso particolare di un grafo), poiché avevamo interesse nel creare una struttura ordinata, in modo da migliorare le prestazioni in fasi successive e non avevamo necessità di altre funzionalità dei grafi.

Albero: le informazioni vengono memorizzate in una struttura ad albero binario di ricerca, quest'ultimo prevede che ogni nodo abbiamo al massimo due figli e che data una chiave *c* appartenente ad un nodo, ad ogni nodo del suo sottoalbero destro appartiene a una chiave di valore ≥ *c*, mentre ad ogni nodo del suo sottoalbero sinistro appartiene una chiave di valore < *c*.

La struttura ad albero utilizzata è ordinata in base ai nomi dei veicoli, non è possibile, l'inserimento di un veicolo con lo stesso nome di uno già presente.

Scelte di progetto

La generazione dei nomi e dei dati di *N* veicoli scelti dall'utente è stata affidata da un programma separato dal resto. Questa scelta è stata effettuata per evitare conflitti tra dati e semplificare il lavoro in fase di sviluppo, oltre al fatto di ritrovarsi con un generatore di dati su file separato.

Per quanto riguarda la generazione dei modelli, è stato scelto di creare un file contenente vari modelli di veicoli che verranno scelti randomicamente all'interno del generatore.

Il resto del progetto è invece contenuto in un altro programma che utilizza il file contenente i dati per delle operazioni, che vengono effettuate utilizzando una struttura ad albero binario di ricerca (scelta per le sue buone prestazioni in fase di inserimento, ricerca e rimozione).

Il programma, dopo aver stampato la lista dei nomi dei veicoli contenuti nel file, chiede all'utente di scegliere l'operazione da effettuare, inclusa quella di terminare il programma. Finché l'utente non sceglierà di terminare il programma, ogni volta che un'operazione termina, il programma richiederà nuovamente all'utente di effettuare una scelta.

Ad ogni input inserito da tastiera viene effettuata una validazione stretta, in base a ciò che il programma richiede di inserire.

Algoritmi:

Ricerca: per la ricerca basta partire dalla radice e proseguire verso destra o sinistra in base al risultato del confronto tra il valore scelto e quello della chiave contenuta nel nodo.

Se il valore scelto è maggiore di quello della chiave del nodo si procederà verso destra, altrimenti verso sinistra.

Inserimento: l'algoritmo di inserimento viene utilizzato per l'inserimento dei dati, contenuti nel file, all'interno dell'albero binario di ricerca. Ogni volta che è necessario inserire un nuovo elemento, bisogna scorrere l'albero fino ad arrivare al nodo corretto. Durante lo scorrimento dell'albero avvengono anche dei confronti tra i dati già inseriti e quello da inserire, poiché

se il nuovo elemento ha valore superiore di quello già inserito, lo scorrimento sarà verso il sottoalbero destro, viceversa con il sinistro.

Le comparazioni avvengono tramite due funzioni: una verifica se i valori sono uguali, mentre l'altra verifica quale dei due è minore.

Ovviamente se l'albero dovesse essere vuoto il primo inserimento corrisponderà alla radice.

Rimozione: l'algoritmo di rimozione viene utilizzato per rimuovere un l'elemento scelto dall'utente. L'algoritmo, partendo dalla radice, scorre verso sinistra se il valore cercato è minore del nodo corrente, altrimenti verso destra, finché non trova il nodo desiderato, per poi:

- se il nodo è una foglia senza figli, basta cancellarlo dall'albero;
- se il nodo ha un figlio solo, si elimina sostituendolo nella struttura dell'albero con il suo unico figlio;
- se il nodo ha due figli si ricerca il suo successore, e si scambia il valore del nodo da cancellare e del successore trovato, cancellando poi solo quest'ultimo (che ha zero o un figlio solo).

Ricerca valore più grande: è una funzione ricorsiva che scorre l'albero sempre verso sinistra fino al raggiungimento dell'estremo, per poi trovare il valore più grande.

Ricerca valore più piccolo: è una funzione ricorsiva che scorre l'albero sempre verso destra fino al raggiungimento dell'estremo, per poi trovare il valore più piccolo.

4. Implementazione dell'algoritmo

GeneratoreDati.c

Inizialmente viene cambiato il seme della funzione *rand()* in modo tale da cambiare ogni volta la generazione randomica dei nomi e dei dati.

- Viene richiesto all'utente quanti veicoli si vogliono generare.
- Viene allocata la memoria necessaria per l'acquisizione dei nomi dei veicoli che vengono successivamente calcolati randomicamente.
- Viene generato/aperto il file in scrittura, su cui andranno inseriti tutti i dati.
- Vengono generati randomicamente tutti i dati dei veicoli e vengono così scritte tutte le informazioni sul file.
- Alla fine della scrittura si procede con la chiusura del file.

ValidazioneNumeroVeicoli:
funzione che chiede all'utente il numero di veicoli, acquisisce la risposta, e la verifica.
Nel caso l'input inserito dall'utente non dovesse essere compatibile con il tipo richiesto verrà comunicato e effettuata nuovamente la richiesta dell'input.

GeneraAnno:
funzione che genera randomicamente l'anno all'interno di un intervallo (1900 – 2021).

GeneraIdVeicoli:
funzione che genera randomicamente tutti i nomi dei veicoli nel formato richiesto dalla specifica.
I nomi generati saranno formati da 4 lettere maiuscole e 2 numeri.

GeneraModelli:
funzione che legge da un file diversi nomi di modelli di auto, li memorizza in una stringa di caratteri, genera randomicamente un numero all'interno dell'intervallo della lunghezza della stringa e restituisce vari modelli memorizzati.

GeneraProprietari:
funzione che genera randomicamente tutti i nomi dei proprietari nel formato richiesto dalla specifica.
I nomi generati saranno formati da 3 lettere maiuscole e 3 numeri.

ApriFile:
funzione che ci permette di aprire il file "*modelli.txt*".
Se l'apertura del file non va a buon fine ritorna come valore 0, altrimenti ritorna 1.

Progetto.c

Apertura del file in lettura.
Nel caso in cui il file non venga aperto correttamente, si termina il programma.
Se il file viene aperto correttamente, si procede con la creazione dell'albero, leggendo i dati da file.
Verranno visualizzati gli id dei veicoli, presenti all'interno del file ("*dati.txt*").
All'interno del codice si ha un ciclo che permette all'utente di poter selezionare una funzione del programma che si desidera utilizzare. Tutto ciò che viene inserito tramite tastiera da utente verrà validato, il ciclo non termina finché l'utente non inserisce un valore accettabile.
All'interno del ciclo si ha uno switch che in base alla scelta dell'utente richiama determinate funzioni in modo da eseguire le operazioni desiderate.

- Case 1: ci permette di ricercare un id scelto dall'utente, stampando a video tutte le informazioni riguardanti quell'id. Stampa, inoltre, i passi dell'algoritmo di ricerca.
- Case 2: ci permette di inserire un nuovo veicolo. Stampa, inoltre, i passi dell'algoritmo di inserimento.
- Case 3: ci permette di rimuovere un veicolo scelto dall'utente. Stampa, inoltre, i passi dell'algoritmo di rimozione.
- Case 4, restituisce il valore più grande e quello più piccolo presenti nel file dati. Stampa, inoltre, i passi dell'algoritmo di calcolo del valore più grande e più piccolo.

Alla fine dello switch, prima di terminare il programma, abbiamo un'assegnazione a nodo e radice dei valori = NULL, per evitare conflitti quando viene riletto il file e ricreata la struttura.

ApriFileLettura:
apertura del file in lettura. Se il file viene aperto correttamente verrà ritornato il valore 1, altrimenti ritorna 0.

ApriFileScrittura:
apertura del file in scrittura. Se il file viene aperto correttamente verrà ritornato il valore 1, altrimenti ritorna 0.

Inserimento:
funzione che inserisce i valori passatole, all'interno dell'albero, scorrendolo e andando a decidere la posizione dei dati da inserire.
Per decidere la posizione dei dati da inserire, si avvale di due funzioni, che controllano se il nome del veicolo sia minore o maggiore di quello già presente.
Lo scorrimento avviene, come già detto, verso destra, se dovesse essere maggiore, altrimenti verso sinistra.

NodiUguali e **NodoMinore**:
funzioni che mi permettono di confrontare due nodi e comunicare quale sia il maggiore o il minore, e se dovessero essere uguali.

StampaIdAlbero:
funzione che stampa a video tutti i nomi dei veicoli utilizzando la struttura ad albero, sviluppata in modo da non ripetere nomi già presenti ed è ricorsiva, per andare ad analizzare tutti i sottoalberi.

CercaIdAlbero:
funzione che serve per scorrere l'albero finché non viene trovato il nome del veicolo richiesto, trovato il veicolo ne stampa tutti i valori.
Ha un valore di ritorno perché nel caso non dovesse essere stato inserito un nome presente nell'albero lo comunicherà all'utente.

VerificaIdVeicolo:
funzione che chiede all'utente di inserire il nome di un veicolo sopra-visualizzato, di cui si vogliono visualizzare le informazioni.
Esegue anche tutti i controlli necessari sul nome inserito.

InserisciNuovoElemento:
funzione che chiede all'utente di inserire le informazioni del nuovo veicolo.
Ogni input da tastiera verrà controllato in base a cosa l'utente sta inserendo (id veicolo, proprietario, modello, anno).
Una volta acquisiti tutti i dati vengono richiamate due funzioni:

- *Inserimento* (per inserire il nuovo nodo nell'albero);
- *StampaAlbero* (ristampa su file tutto l'albero aggiornato).

CreaAlbero:
funzione che leggendo dati da file li passa alla funzione *Inserimento* per creare la struttura ad albero.

StampaAlbero:
funzione che apre il file in scrittura, richiama la funzione *StampaAlberoParziale*, e chiude il file una volta finita l'esecuzione.

StampaAlberoParziale:
funzione che stampa su file un nodo per volta richiamando sé stessa.
Sono funzioni separate poiché l'apertura e la chiusura all'interno della stessa funzione richiamata, avrebbero causato conflitti.

RimuoviElemento:
algoritmo che scorrendo l'albero cerca il nodo in cui è presente lo stesso id desiderato, per andarlo poi a rimuovere.

RicercaMax / RicercaMin:
Scorrimento verso sinistra o verso destra per trovare l'estremo sinistro e l'estremo destro, in cui sono memorizzati il valore maggiore e il valore minore.

5. Testing del programma

Testing "GeneratoreDati.c":

```
Inserire il numero di veicoli che si vogliono analizzare
-5

Il valore inserito non e' accettabile
Perfavore inserire un numero naturale diverso da 0
Reinserire il valore:   15

File Generato con Successo
```

Viene chiesto dal programma quanti veicoli si vogliono analizzare.
Quando vengono inseriti dei valori non accettabili il programma ce lo comunica e ci chiede nuovamente di inserire valori.
Al primo valore accettabile che inseriamo, il programma procede con l'esecuzione e termina.

Testing "Progetto.c":

```
Lista degli id dei veicoli che sono stati analizzati:
        BHQE12
        CXQX33
        DDWW26
        HPWR06
        MNIP80
        OGTR51
        QNKJ54
        RQCU54
        RWXR55
        SRHS72

Selezionare l'opzione che si vuole eseguire:
(1) Cerca
(2) Inserisci nuovo elemento
(3) Rimuovi un elemento
(4) Calcola il valore piu' grande e quello piu' piccolo
(0) Termina

Numero: L
Opzione non valida...   Riprovare

Selezionare l'opzione che si vuole eseguire:
(1) Cerca
(2) Inserisci nuovo elemento
(3) Rimuovi un elemento
(4) Calcola il valore piu' grande e quello piu' piccolo
(0) Termina

Numero: 58
Opzione non valida...   Riprovare

Selezionare l'opzione che si vuole eseguire:
(1) Cerca
(2) Inserisci nuovo elemento
(3) Rimuovi un elemento
(4) Calcola il valore piu' grande e quello piu' piccolo
(0) Termina

Numero: 1

Inserire l'id di un veicolo sopra-visualizzato:
```

Si richiede di selezionare l'opzione che si vuole eseguire.

Se l'utente inserisce un qualsiasi valore al di fuori dei cinque possibili, viene comunicato l'errore e rieseguita la domanda all'utente.

```
Numero: 1

Inserire l'id di un veicolo sopra-visualizzato:
bhqe12

Non e' stato trovato alcun veicolo, (attenzione inserire lettere maiuscole!)

I passi compiuti dall'algoritmo per trovare e stampare i dati del veicolo richiesto sono: 3

--------------------------------------------------------------------------------
```

Opzione 1: Se l'utente inserisce sei caratteri con lettere minuscole o un id veicolo non presente, il programma procede con la ricerca senza restituire alcun veicolo, comunicando che non è presente un veicolo con quel nome.

```
Numero: 1

Inserire l'id di un veicolo sopra-visualizzato:
ABCDEFGH25

Errore nell'aquisizione dell'id del veicolo scelto...
Controllare la lunghezza dell'id inserito
ABC54

Errore nell'aquisizione dell'id del veicolo scelto...
Controllare la lunghezza dell'id inserito
OGTR51

Ecco i dati riguardanti il veicolo selezionato:
OGTR51  TDD272  Toyota-Hybrid   1956

I passi compiuti dall'algoritmo per trovare e stampare i dati del veicolo richiesto sono: 3

--------------------------------------------------------------------------------
```

Se l'utente invece inserisce qualsiasi altra lunghezza, il programma lo rileva come errore e chiede all'utente di reinserire il nome del veicolo desiderato.
Quando inserisce un nome corretto e presente, il programma procede con la ricerca e la stampa le informazioni riguardanti quel veicolo.

```
Numero: 2

Inserire l'id del veicolo:
        ahsd98

Il valore immesso non e' corretto
Inserire solo 4 lettere maiuscole e 2 numeri

Inserire l'id del veicolo:
        2594D55DF

Il valore immesso non e' corretto
Inserire solo 4 lettere maiuscole e 2 numeri

Inserire l'id del veicolo:
        MKOI98

Inserire il nome del proprietario
        sjd987

Il valore immesso non e' corretto
Inserire solo 3 lettere maiuscole e 3 numeri

Inserire il nome del proprietario
        159LO

Il valore immesso non e' corretto
Inserire solo 3 lettere maiuscole e 3 numeri

Inserire il nome del proprietario
        ABC123

Inserire il modello del veicolo
NB. I caratteri dopo uno spazio non verranno acquisiti
        Fiat-500

Inserire l'anno del veicolo
        2056

Il valore immesso non e' corretto
Inserire correttamente l'anno

Inserire l'anno del veicolo
        LKJO

Il valore immesso non e' corretto
Inserire correttamente l'anno

Inserire l'anno del veicolo
        2015

I passi compiuti dall'algoritmo per inserire i dati sono: 5

--------------------------------------------------------------------
```

Opzione 2:
Quando si sceglie l'opzione numero due, quindi di inserire un nuovo veicolo, viene richiesto all'utente di inserire dei dati:
l'id del veicolo: quando l'utente inserisce dei dati non compatibili con il nome del veicolo specificato, il programma richiede all'utente di inserire il nome del veicolo fino a che l'utente non ne inserisce uno accettabile.

Stessa cosa accade per il resto dei dati come il nome del proprietario, il modello, e l'anno.

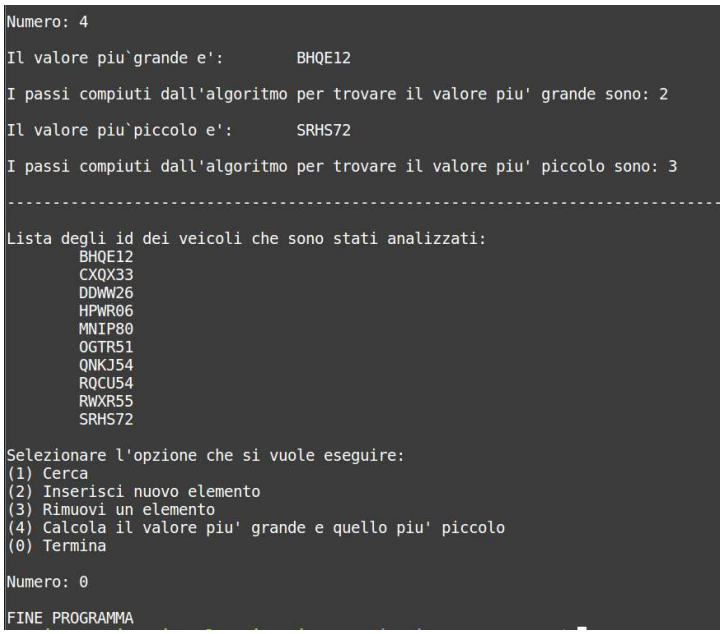
```
Numero: 3

Inserire l'id di un veicolo sopra-visualizzato:
ASDF45

Non e' stato trovato nessun elemento da eliminare

I passi compiuti dall'algoritmo per rimuovere l'elemento selezionato sono: 2

--------------------------------------------------------------------------------
```

Opzione 3:
Quando l'utente sceglie la terza opzione, ovvero rimuovere un veicolo, si ritrova a dover inserire il nome di uno dei veicoli sopracitati.
Se l'utente inserisce un nome di 6 caratteri non presente, il programma procede con l'esecuzione e avvisa che non è stato trovato alcun elemento da eliminare.

```
Numero: 3

Inserire l'id di un veicolo sopra-visualizzato:
ASDFGFG56

Errore nell'aquisizione dell'id del veicolo scelto...
Controllare la lunghezza dell'id inserito
23AS

Errore nell'aquisizione dell'id del veicolo scelto...
Controllare la lunghezza dell'id inserito
MKOI98

Veicolo rimosso

I passi compiuti dall'algoritmo per rimuovere l'elemento selezionato sono: 5

--------------------------------------------------------------------------------
```

Quando invece, inserisce un nome più lungo di 6 caratteri o più corto di 6 caratteri il programma da errore.

Mentre se l'utente inserisce un nome presente, il programma procede con la rimozione del veicolo.



Opzione 4:
Quando l'utente seleziona la quarta opzione, il programma va a trovare il valore più grande e quello più piccolo, stampandoli poi a schermo, insieme al numero di passi che sono stati necessari per trovarli.

Premendo poi lo zero il programma termina.

6. Valutazione della complessità del programma

Analisi teorica della complessità:

Algoritmo di ricerca:

Per trovare il veicolo scelto dall'utente, viene utilizzata la funzione *CercaIdAlbero*.
Quest'ultima scorre l'albero grazie a un ciclo, fino a quando non trova l'elemento scelto e arriva alla fine dell'albero senza trovarne alcuno. Quando ciò che cerca viene trovato, vengono stampate tutte le informazioni riguardanti quel nodo.
La complessità di questo algoritmo è:

In un caso ottimale (albero vuoto) risulta:

$\mathrm{T}(n) = O(1)$ complessità costante

In un caso pessimo (albero non vuoto e cerca un elemento non presente, deve perciò essere percorso interamente) risulta:

$\mathrm{T}(n) = O(\mathrm{n})$ complessità lineare

In un caso medio (albero non vuoto, il numero di iterazioni è proporzionale alla lunghezza media del percorso per raggiungere un nodo dalla radice) risulta:

$\mathrm{T}(n) = \mathrm{O}(\log \mathrm{n})$ complessità logaritmica

Algoritmo di inserimento:

Per inserire un nuovo elemento, l'algoritmo di inserimento, scorre l'albero relativamente al valore che desideriamo inserire.
Così facendo si sposta tra nodi sinistri e nodi destri, in base a un confronto tra i valori dei nodi già presenti e quello da inserire fino a raggiungere la posizione dove inserirà poi il nuovo valore.

In un caso ottimale (albero vuoto) risulta:

$\mathrm{T}(n) = O(1)$ complessità costante

In un caso pessimo (albero non vuoto e inserimento di un valore per cui bisogna compiere tutto il percorso dalla radice alla foglia più distante da essa) risulta:

$\mathrm{T}(n) = O(n)$ complessità lineare

In un caso medio (albero non vuoto, il numero di iterazioni è proporzionale alla lunghezza media del percorso per raggiungere un nodo dalla radice) risulta:

$\mathrm{T}(n) = \mathrm{O}(\log \mathrm{n})$ complessità logaritmica

Algoritmo di rimozione:

Per rimuovere un elemento, l'algoritmo di rimozione, scorre l'albero relativamente al valore che desideriamo rimuovere.
Così facendo si sposta tra nodi sinistri e nodi destri, in base a un confronto tra i valori dei nodi già presenti e quello da rimuovere fino a raggiungere il valore che andrà poi rimosso.

In un caso ottimale (albero vuoto) risulta:

$\mathrm{T}(n) = O(1)$ complessità costante

In un caso pessimo (albero non vuoto e rimozione di un valore per cui bisogna compiere tutto il percorso dalla radice alla foglia più distante da essa) risulta:

$\mathrm{T}(n) = O(n)$ complessità lineare

In un caso medio (albero non vuoto, il numero di iterazioni è proporzionale alla lunghezza media del percorso per raggiungere un nodo dalla radice) risulta:

$\mathrm{T}(n) = \mathrm{O}(\log \mathrm{n})$ complessità logaritmica

Algoritmi di calcolo del valore più grande e più piccolo:

Per trovare l'elemento con il valore più grande e più piccolo, bisogna scorrere l'albero in una direzione e nell'altra, fino al raggiungimento degli estremi.

In un caso ottimale (albero vuoto) risulta:

$\mathrm{T}(n) = O(1)$ complessità costante

In un caso pessimo (albero non vuoto, in quanto bisogna compiere tutto il percorso dalla radice alla foglia più distante da essa) risulta:

$\mathrm{T}(n) = O(n)$ complessità lineare

Analisi sperimentale della complessità

Con l'analisi sperimentale andremo a studiare l'andamento degli algoritmi all'aumentare del numero dei veicoli.

Per quanto riguarda gli algoritmi di ricerca e rimozione, abbiamo utilizzato sempre dei nomi diversi ma che fossero sempre all'incirca nella media dei valori dei nomi.

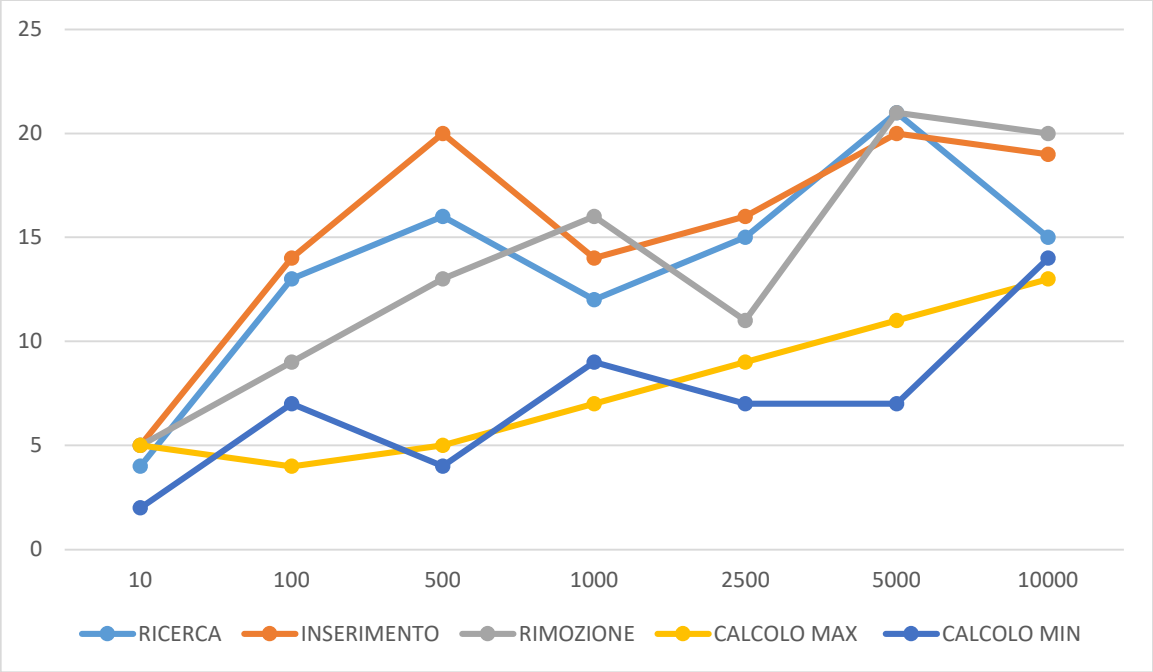
Per l'inserimento, invece, abbiamo utilizzato lo stesso nome in modo da analizzare il comportamento sempre nella stessa situazione.

NB. Capita che all'aumentare di *n*, effettuata una ricerca, un inserimento o una rimozione, si possa ottenere un numero di passi minore. Questo accade poiché alla generazione di nuovi dati e alla esecuzione del programma la struttura ad albero cambia, perciò sarà impossibile eseguire dei test che restituiranno sempre gli stessi risultati, anche ad *n* fisso:

esempio con inserimento: supponiamo di eseguire due test dove generiamo entrambe le volte 50 veicoli. Supponiamo di aggiungere entrambe le volte uno stesso id, il numero dei passi cambierà poiché alcune volte la generazione randomica potrebbe generare più id con valori bassi, mentre altre volte potrebbe generare più id con valori alti. Ciò comporterà che, l'id che noi andremo ad inserire, alcune volte, impiegherà più o meno passi rispetto ad altre per venire inserito nell'albero.

Allo stesso modo potremmo eseguire esperimenti sia per gli algoritmi di ricerca che di rimozione, sempre nel caso in cui andiamo a ricercare o rimuovere in entrambi i test gli stessi veicoli (utilizzando una generazione randomica sarà altamente improbabile poter andare a cercare o rimuovere gli stessi valori in due test con dati nati da generazioni differenti).

VEICOLI	RICERCA	INSERIMENTO	RIMOZIONE	CALCOLO MAX	CALCOLO MIN
10	4	5	5	5	2
100	13	14	9	4	7
500	16	20	13	5	4
1000	12	14	16	7	9
2500	15	16	11	9	7
5000	21	20	21	11	7
10000	15	19	20	13	14



Analizzando l'andamento del grafico e trascurando alcuni punti centrali un po' barcollanti, possiamo dire che il numero di passi aumenta all'aumentare dei numeri dei veicoli. La crescita non è lineare, anzi, la crescita del numero di passi è molto più lenta rispetto alla crescita di *n*.

Possiamo perciò dire che l'andamento dei passi rispetta l'analisi teorica della complessità ovvero è di tipo logaritmico.