Studente: Susanna Peretti
Numero di matricola: 306186
Iscritta al 1° anno del corso di Laurea Triennale di Informatica Applicata

Corso di Algoritmi e Strutture Dati
Progetto per la sessione estiva 2020/2021

Docente: Prof. Valerio Freschi

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

Analisi degli input:

Il primo input è costituito da un file di testo .txt da acquisire e memorizzare i dati relativi agli autoveicoli; per ogni autoveicolo sono forniti il codice del veicolo, il codice del proprietario, il modello del veicolo e l'anno di immatricolazione. A seguito, in base all'operazione da svolgere, si richiedono diversi input:
1) Inserire un veicolo – l'utente deve inserire le quattro informazioni richieste;
2) Rimuovere un veicolo – l'utente deve inserire il codice veicolo dell'autoveicolo da rimuovere;
3) Cercare un veicolo – l'utente deve inserire il codice veicolo del veicolo da cercare;

Analisi degli output:

Per le operazioni di acquisizione del file, inserimento e rimozione di un veicolo gli output sono costituiti da un messaggio di conferma o errore del risultato richiesto. Per le operazioni di ricerca, calcolo del valore più grande e calcolo del valore più piccolo, si restituiscono i quattro dati relativi all'autoveicolo che si sta cercando.

Relazioni intercorrenti fra input e output:

La relazione tra input e output si basa sulla stampa e modifica dell'insieme dei dati in base al codice veicolo preso in esame. Data l'unicità dei codici veicolo, si esegue una classificazione ed un ordinamento secondo i codici.

3. Progettazione dell'Algoritmo

Scelte di progetto:

Come riportato nel punto 1 della "Specifica del problema", i dati acquisiti da file devono essere inseriti in un'opportuna struttura dati. La scelta di una tra le strutture analizzate, deve comprendere un requisito di dinamicità data la necessità di inserimento e rimozione di dati durante l'esecuzione del programma e la necessità di dover esaminare i codici veicolo per poter cercare un determinato autoveicolo.
La struttura dati migliore per questo tipo di elaborato è l'albero binario di ricerca, una struttura dati tale che, per ogni nodo n dell'albero tutte le chiavi dei nodi contenuti nel sottoalbero sinistro di n hanno valore minore della chiave contenuta in n, e tutte le chiavi dei nodi contenuti nel sottoalbero destro di n hanno valore maggiore della chiave contenuta in n. Questa proprietà viene mantenuta anche a seguito di algoritmi di inserimento e rimozione, permettendo un algoritmo di ricerca con una complessità minore.
Ogni elemento dell'albero contiene il codice veicolo, il codice proprietario, la marca, il modello dell'autoveicolo e l'anno di immatricolazione. Il codice veicolo è salvato come array di caratteri, ossia una stringa, di cui i primi quattro caratteri devono essere lettere maiuscole e gli ultimi due numeri; il codice proprietario, salvato anch'esso come stringa, deve avere i primi tre caratteri come lettere maiuscole e gli ultimi tre caratteri come numeri. Entrambi i codici devono avere obbligatoriamente sei caratteri quindi si eseguono validazioni sul corretto inserimento del codice, facendo distinzione fra lettere e numeri e sulla lunghezza della stringa.
Si è reso necessario utilizzare due stringhe per l'acquisizione del modello dell'autoveicolo dato che il carattere di spaziatura presente (ad esempio: FIAT 500) presentava problemi nell'acquisizione del dato.
Infine, l'anno di immatricolazione è dato di tipo int.

Passi dell'algoritmo

1) Acquisizione del file di testo .txt e memorizzazione delle relative informazioni in una struttura dati dinamica di tipo 'albero binario'.
2) Si richiede all'utente quale fra le sei seguenti operazioni si vuole eseguire:
   - inserimento di un nuovo autoveicolo;
   - rimozione di un autoveicolo acquisito precedentemente dal file;
   - ricerca di un determinato autoveicolo attraverso l'inserimento del codice veicolo;
   - calcolo del valore più grande, secondo l'ordine lessicografico, del codice alfanumerico rappresentante l'autoveicolo;
   - calcolo del valore più piccolo del codice veicolo;
3) La sesta operazione è quella di chiusura del programma al quale conseguono due richieste:
   - richiesta di accedere ai dati relativi agli autoveicoli aggiornati attraverso la stampa a schermo;
   - richiesta di modificare il file txt utilizzato precedentemente per l'acquisizione.

A seguito di queste operazioni di stampa, il programma termina.

4. Implementazione dell'Algoritmo

Funzione main

La funzione main, come prima cosa, apre in lettura il file di testo file_auto.txt. Se il file non viene trovato o non viene aperto correttamente, viene stampato un messaggio di errore e il programma termina. Se invece si ha esito positivo, viene chiamata la funzione acquisire_file. A seguito della memorizzazione dei dati iniziali, il programma consente di scegliere fra sei operazioni digitando un numero da 0 a 5. Si esegue una validazione dell'inserimento del numero nella variabile to_do e con delle istruzioni di selezione if else si esegue un determinato blocco di istruzioni con all'interno una chiamata di funzione diversa.
Con to_do == 0 si richiede all'utente di scegliere se stampare a schermo i dati archiviati e successivamente se si vuole modificare il file di resto file_auto.txt acquisito. Per entrambe le stampe viene utilizzata la funzione stampare_dati con l'utilizzo di una variabile controllo 'stampa'. A seguito di queste due stampe, si esce dal ciclo do while e termina il programma.
Con to_do == 1 si richiama la funzione di verifica_nuovo_inserimento per l'acquisizione e validazione dei nuovi dati da inserire relativi ad un nuovo autoveicolo.
Con to_do == 2 si chiede di inserire il codice veicolo relativo all'autoveicolo che si vuole rimuovere dall'elenco e si richiama la funzione rimuovere_veicolo al quale si passa un puntatore alla radice dell'albero e una variabile contenente il dato appena acquisito.
Con to_do == 3 si chiede di inserire il codice veicolo dell'autoveicolo di cui si vuole avere tutte le informazioni relative e si chiama la funzione cercare_veicolo.
Con to_do == 4 si cerca l'autoveicolo con il codice veicolo maggiore attraverso la funzione veicolo_maggiore.
Con to_do == 5 si cerca, invece, l'autoveicolo con il codice veicolo minore con la funzione veicolo_minore.

Funzione acquisire_file

La funzione consente di acquisire dal file di testo i quattro dati relativi agli autoveicoli. All'interno della funzione è presente un ciclo do while che permette di acquisire i dati fino a quando non si raggiunge la fine del file. Per il codice veicolo e per il codice proprietario viene eseguita una validazione sui singoli caratteri della stringa, attraverso un ciclo for, e una validazione della lunghezza della stringa utilizzando la funzione strlen. A meno di errori, viene chiamata la funzione inserire_veicolo. Nel caso in cui nel file sia presente un codice veicolo non idoneo, la variabile di controllo errore (incrementata durante le precedenti validazioni) risulta diversa 0, la funzione di inserimento non viene chiamata e il codice veicolo non è aggiunto in elenco.

Funzione verifica_nuovo_inserimento

La funzione viene chiamata quando l'utente sceglie di inserire informazioni relative ad un nuovo autoveicolo. La funzione chiede all'utente di inserire un'informazione validando che essa sia valida a seconda delle restrizioni, nel caso dei codici veicolo e proprietario per il numero di lettere maiuscole e numeri, e nel caso del modello e dell'anno di immatricolazione si esegue una validazione stretta dell'acquisizione attraverso la variabile di controllo esito_lettura. Al termine del corretto inserimento delle quattro informazioni si chiama la funzione inserire_veicolo passando ad essa le informazioni acquisite con una variabile to_add di tipo struttura definita precedentemente. Il codice veicolo viene confrontato con quelli presenti nell'albero per verificare che non ne esista già uno uguale.

Funzione inserire_veicolo

La funzione sfrutta il criterio di ordinamento per cercare il punto in cui inserire un nuovo elemento valutando:
- se l'albero è vuoto crea una radice;
- se il codice veicolo da inserire è uguale ad un codice presente nel nodo in esame rifiuta l'inserimento;
- se il codice veicolo è minore del codice in esame, si esamina il sottoalbero sinistro;
- se il codice veicolo è maggiore del codice in esame, si esamina il sottoalbero destro.

Alla funzione viene passata anche una variabile di tipo int, denominata stampa, passata con valore pari a 0 nel caso in cui la funzione sia richiamata dalla funzione di crea_albero e con valore pari ad 1 nel caso in cui la chiamata sia nella funzione di verifica_nuovo_inserimento così da poter controllare se dover stampare un messaggio di corretto inserimento di un nuovo valore ed evitarlo nel caso in cui si siano appena acquisiti i valori dal file di testo file_auto.txt.

Funzione rimuovere_veicolo

La funzione permette di rimuovere un determinato nodo contenente il codice veicolo richiesto dall'utente. Attraverso un ciclo for si confronta, a partire dalla radice, il codice contenuto nella variabile veicolo_rimosso con i codici presenti nell'albero, proseguendo con il sottoalbero sinistro in caso il valore sia minore e con il sottoalbero destro nel caso in cui il valore sia maggiore del nodo in esame.
Se si individua il nodo giusto, ovvero è presente il codice veicolo inserito dall'utente, si considerano tre casi base:
- se il nodo non ha figli, si rimuove il nodo;
- se il nodo ha un solo figlio, esso si eleva in modo da occupare la posizione del nodo da dover rimuovere;
- se il nodo ha due foglie, si individua il successore così che assuma la posizione del nodo da rimuovere.

Funzione cercare_veicolo

La funzione, partendo dalla radice, confronta con un ciclo for il codice contenuto nella variabile veicolo_ricerca con i nodi dell'albero, proseguendo con il sottoalbero sinistro in caso il valore sia minore e con il sottoalbero destro nel caso in cui il valore sia maggiore. Se il codice veicolo è presente in elenco, vengono stampati a schermo tutti i dati relativi all'autoveicolo.

Funzione veicolo_maggiore

La funzione permette di stampare i dati relativi all'autoveicolo con codice veicolo più grande secondo l'ordine lessicografico. Con un'istruzione condizionale if else se il figlio destro del nodo è pari a NULL (quindi non esiste), vengono stampati a schermo i dati relativi all'autoveicolo indirizzato dal puntatore altrimenti si richiama ricorsivamente la funzione puntando il figlio destro successivo. Questo perché nel sottoalbero destro sono presenti solo nodi con elementi di valore maggiore.

Funzione veicolo_minore

La funzione permette di stampare i dati relativi all'autoveicolo con codice veicolo più piccolo secondo l'ordine lessicografico. Con un'istruzione condizionale if else se il figlio sinistro del nodo è pari a NULL (quindi non esiste), vengono stampati a schermo i dati relativi all'autoveicolo indirizzato dal puntatore altrimenti si richiama ricorsivamente la funzione puntando il figlio sinistro successivo. Questo perché, come nel destro, nel sottoalbero sinistro sono presenti nodi con elementi di valore minore.

Funzione stampare_dati

La funzione è un algoritmo di visita in ordine anticipato in cui si elabora il valore contenuto in un nodo quindi si applica lo stesso algoritmo prima al sottoalbero sinistro e poi al sottoalbero destro. Nella funzione è presente una variabile di controllo 'stampa' che permette, tramite un'istruzione if else di stampare a schermo i dati presenti nel nodo o se stampare su file attraverso un fprintf. Questa variabile mi permette di sfruttare la stessa funzione per le operazioni conclusive di stampa quando il programma termina.

5. Testing del Programma

1) Compilazione e avvio del programma con errore nell'acquisizione del file.

```
supe@supe-VirtualBox:~/Scrivania/algoritmi$ make
gcc -ansi -Wall -O anagrafe_autoveicoli.c -o anagrafe_autoveicoli
supe@supe-VirtualBox:~/Scrivania/algoritmi$ ./anagrafe_autoveicoli
Avvio programma.

Il file non e' stato acquisito correttamente. Il programma e' terminato.
supe@supe-VirtualBox:~/Scrivania/algoritmi$
```

2) Acquisizione corretta del file di testo e selezione delle operazioni da poter eseguire.

```
supe@supe-VirtualBox:~/Scrivania/algoritmi$ ./anagrafe_autoveicoli
Avvio programma.

Il file e' stato acquisito correttamente.

Selezionare l'operazione da svolgere:
0)Chiudi programma              1)Inserire un veicolo
2)Rimuovere un veicolo          3)Cercare un veicolo
4)Calcolare il valore maggiore  5)Calcolare il valore minore
```

3) Inserimento valido di un nuovo autoveicolo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma              1)Inserire un veicolo
2)Rimuovere un veicolo          3)Cercare un veicolo
4)Calcolare il valore maggiore  5)Calcolare il valore minore
1

Inserire il codice del nuovo veicolo: AMRA93

Inserire il codice proprietario del nuovo veicolo: DSE114

Inserire il modello del veicolo: (max 20 caratteri): opel mokka

Inserire l'anno di immatricolazione del nuovo veicolo: 2000

Il veicolo e' stato inserito con successo.
```

4) Inserimento non valido di un autoveicolo già presente in elenco.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma              1)Inserire un veicolo
2)Rimuovere un veicolo          3)Cercare un veicolo
4)Calcolare il valore maggiore  5)Calcolare il valore minore
1

Inserire il codice del nuovo veicolo: DDSJ16

Inserire il codice proprietario del nuovo veicolo: DZY228

Inserire il modello del veicolo: (max 20 caratteri): fiat Bravo

Inserire l'anno di immatricolazione del nuovo veicolo: 2017

Il veicolo non e' stato inserito perche' e' già presente in elenco.
```

5) Corretta rimozione di un autoveicolo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
2

Inserire il codice veicolo da rimuovere: DDSJ16

Il veicolo e' stato rimosso con successo.
```

6) Rimozione non valida di un autoveicolo perché non presente in elenco.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
2

Inserire il codice veicolo da rimuovere: DDSJ15

Il codice veicolo non e' stato trovato.
```

7) Corretta ricerca delle informazioni relative ad un dato codice veicolo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
3

Inserire il codice veicolo da cercare: MZKP03

Veicolo richiesto:
Veicolo         Proprietario    Modello         Anno
MZKP03          ACM603          ford S-Max      1996
```

8) Ricerca non valida dato un codice veicolo perché non presente in elenco.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
3

Inserire il codice veicolo da cercare: OKON73

Il veicolo richiesto non e' stato trovato.
```

9) Calcolo del valore più grande, secondo l'ordine lessicografico, del codice alfanumerico rappresentante il veicolo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
4

Il veicolo con il valore maggiore e':
Veicolo         Proprietario    Modello          Anno
YQSK66          ESP541          volkswagen polo 2017
```

10) Calcolo del valore più piccolo, secondo l'ordine lessicografico, del codice alfanumerico rappresentante il veicolo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
5

Il veicolo con il valore minore e':
Veicolo         Proprietario    Modello          Anno
DDSJ16          DZY228          fiat Bravo       2017
```

11) Richiesta di stampa a schermo e aggiornamento del file di testo.

```
Selezionare l'operazione da svolgere:
0)Chiudi programma                1)Inserire un veicolo
2)Rimuovere un veicolo            3)Cercare un veicolo
4)Calcolare il valore maggiore    5)Calcolare il valore minore
0

Premere 's' per accedere ai dati relativi agli autoveicoli salvati.
s
Veicolo         Proprietario    Modello          Anno
FVQS07          RBV167          hyundai I20      2005
DDSJ16          DZY228          fiat Bravo       2017
YQSK66          ESP541          volkswagen polo 2017
OKQN73          XOK105          ford Focus       2013
MZKP03          ACM603          ford S-Max       1996

Premere 's' per modificare i dati all'interno del file di testo 'file_auto.txt'.
s

Il file e' stato modificato con successo.

Termine programma.
```

6. Valutazione della complessità del programma

Analisi teoria

Calcolo della complessità dell'algoritmo di inserimento della funzione **inserire_veicolo**

Caso ottimo

1 + 1 +(1 + 1) + (1 + 1) + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1.

- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni pari a 0;
- istruzione di selezione if;
- istruzione di selezione if;
- funzione printf() di stampa
- allocazione dinamica della memoria per la nuova radice;
- assegnamento del nuovo valore al puntatore nuovo_p;
- inizializzazione pari a NULL dei due figli del nuovo elemento;
- istruzione di selezione if;
- assegnamento del valore del nuovo elemento alla radice.

Nel caso ottimo non sono eseguite iterazioni all'interno del ciclo for dato che il valore è inserito in un albero vuoto, e quindi nella radice, e l'algoritmo non dipende dall'altezza dell'albero. L'algoritmo ha una complessità costante $T(n) = O(1)$.

Caso pessimo

1 + 1 + n(1 + 1) +n(1 + 1) + 1 + 1 + 1 + 1 + 1 + 1 + 1.

- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni proporzionale all'altezza dell'albero;
- istruzione di selezione if;
- funzione printf() di stampa;
- allocazione dinamica della memoria per il nuovo nodo;
- assegnamento del nuovo valore al puntatore nuovo_p;
- inizializzazione pari a NULL dei due figli del nuovo elemento;
- istruzione di selezione if;
- il puntatore destro o sinistro del nodo padre punta al nuovo nodo;

Nel caso pessimo il numero di iterazioni è proporzionale all'altezza dell'albero poiché l'altezza di un albero è massima quando esso degenera in una lista, quindi il codice veicolo deve essere inserito alla fine dell'elenco come foglia. L'algoritmo ha una complessità asintotica lineare $T(n) = O(n)$.

Calcolo della complessità dell'algoritmo di rimozione della funzione **rimuovere_veicolo**

Caso ottimo

1 + 1 + (1 + 1) + (1 + 1) + 1 + 1.

- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni pari a 0;
- istruzione di selezione if;
- funzione printf() di stampa;

Nel caso ottimo l'algoritmo di rimozione è applicato ad un albero vuoto, la complessità è costante $T(n) = O(1)$.

Caso pessimo

1 + 1 + n(1 + 1) + n(1 + 1) + 1 + 1 + 1 + 1 + 1 + 1 + 1 + n + n(1 + 1) + 1 + 1 + 1.

- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni proporzionale all'altezza dell'albero;
- istruzione di selezione if;
- funzione printf() di stampa;
- due istruzioni di selezione if;
- assegnamento sost_p = nodo_p;
- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili;
- assegnamento;
- istruzione di selezione if;

- assegnamento.
In questo caso, si presuppone che l'albero sia degenerato in una lista e il veicolo da rimuovere sia l'ultimo elemento, ovvero la distanza fra la radice e la foglia da rimuovere è pari all'altezza h dell'albero. L'algoritmo ha complessità lineare T(n) = O(n).

Calcolo della complessità dell'algoritmo di ricerca della funzione **cercare_veicolo**
Caso ottimo
1 + (1 + 1) + 1 + 1;
- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni pari a 0;
- istruzione di selezione if;
- funzione printf() di stampa;
Nel caso ottimo, la complessità asintotica è T(n) = O(1).
Caso pessimo
1 + n(1+ 1) + n + (1 + 1 + 1).
- istruzione di ripetizione for: espressione di inizializzazione, condizione di continuazione e aggiornamento delle variabili. Numero di iterazioni proporzionale all'altezza dell'albero;
- tre funzioni printf() di stampa.
In questo caso si presuppone che l'albero sia degenerato in una lista e il veicolo di cui stampare tutti i dati sia l'ultimo codice veicolo dell'elenco, quindi l'algoritmo deve scorrere tutto l'albero prima di arrivare al nodo giusto. Di conseguenza, la complessità è T(n) = O(n).

Calcolo della complessità degli algoritmi di ricerca delle funzioni **veicolo_maggiore** e **veicolo_minore.**
Le funzioni di ricerca del valore maggiore e minore hanno stessa complessità asintotica.
Caso ottimo
1 + 1 + 1.
- istruzione di selezione if;
- due funzioni printf() di stampa.
Complessità asintotica T(n) = O(1).
Caso pessimo
n(1+ 1 + 1).
Complessità asintotica T(n) = O(n).

Nel caso medio la complessità degli algoritmi precedentemente elencati si assume che le n chiavi presenti nell'albero binario di ricerca siano state inserite casualmente e che la distribuzione delle n chiavi sia bilanciata, così da rendere l'albero stesso bilanciato. In questo caso, si può dimostrare che gli algoritmi precedentemente elencati hanno una complessità asintotica pari a T(n) = O(log(n)).

Analisi sperimentale

Si esegue l'analisi sperimentale calcolando i passi che l'algoritmo compie su un albero con nodi via via crescenti.

Per effettuare i test si genera un file di testo con l'esecuzione del file eseguibile file_auto, del codice sorgente file_auto.c. Il programma consente di inserire in input il numero di autoveicoli desiderati e stampa su un file di testo, chiamato file_auto.txt: n codici veicolo e codici proprietario creati casualmente attraverso la funzione srand, il modello di auto è selezionato casualmente fra 25 tipi di modelli e, infine, l'anno di immatricolazione anch'esso casuale dal 1990 in poi. A seguire si compila ed esegue il programma v_sperimentale, ovvero una copia dell'originale programma anagrafe_autoveicoli, contenente una variabile contatore 'passi' che misura il numero di istruzioni base eseguite a tempo di esecuzione dalle varie funzioni di inserimento, rimozione e ricerca.

Esempio di un file di testo file_auto.txt con 10 autoveicoli:

```
Veicolo     Proprietario    Modello         Anno

RYHS62      KJJ456      Ford Fiesta         2004
THSL77      ENP381      Opel Corsa          1998
LJTF08      XMG285      Fiat Bravo          1999
ULZN54      YOU120      Fiat Bravo          2002
FLSS80      XFO035      Fiat Ottimo         1991
LMYF51      TSQ035      Fiat Punto          2018
XOYE00      WAF645      Ford Laser          2001
QUUC02      JIT456      Fiat Doblo'         2003
XWVW82      AOP337      Ford Focus          1997
QLYH82      LQM335      Opel Diplomat       2011
```

Test 1

Numero veicoli	Inserimento	Rimozione	Ricerca	V. maggiore	V. minore
10	31	18	19	3	6
100	56	36	27	8	7
1000	91	53	51	9	6
10000	116	88	75	8	13

Test 2

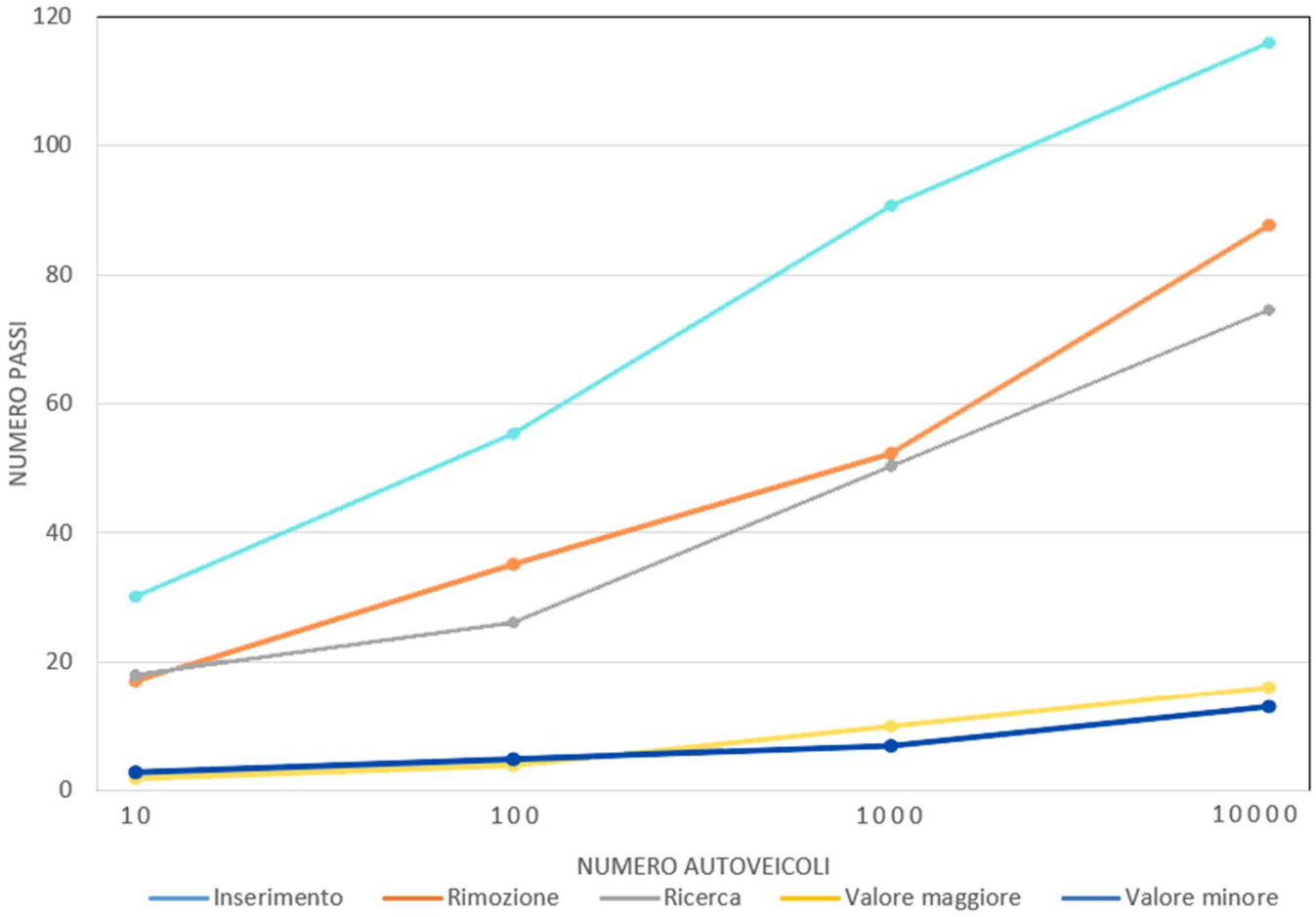
Numero veicoli	Inserimento	Rimozione	Ricerca	V. maggiore	V. minore
10	29	23	18	4	5
100	46	63	31	5	8
1000	101	58	52	10	8
10000	126	128	83	13	13

Test 3

Numero veicoli	Inserimento	Rimozione	Ricerca	V. maggiore	V. minore
10	36	23	11	4	5
100	51	28	51	6	8
1000	86	73	64	11	7
10000	121	74	73	12	12

Dai dati si analizza come la funzione di inserimento sia quella con complessità asintotica maggiore, questo perché la struttura dati utilizzata è un albero binario di ricerca che richiede di cercare il nodo corretto (secondo l'ordine di grandezza) per poter inserire un elemento al suo interno.

Di seguito, il grafico delle complessità asintotiche delle funzioni, secondo una media del numero di passi calcolato nei test. In ascissa il numero di autoveicoli registrati, mentre in ordinata il numero di passi base eseguiti dalle funzioni.



Si può notare dal grafico come le funzioni tendino ad avere una crescita logaritmica dovuta al numero di autoveicoli utilizzati per fare i test (10, 100, …) dato che risulti più probabile l'uscita di un caso medio, in particolare per la funzione di ricerca.