Algoritmi e Strutture Dati

Progetto per la sessione invernale 2017/2018

------- *Telemetria* -------

Sviluppatore

Luca Panariello

Matricola 289182

Specifica del problema

Si supponga di dover progettare un programma per la gestione di un sistema di telemetria di una automobile. Il sistema rileva una serie di parametri di funzionamento che vengono misurati una volta al secondo. I dati raccolti periodicamente vengono organizzati in un file di log (in formato testo) nel quale ad ogni riga sono associati i seguenti valori (si assumano campi separati da tabulazione o spazio):

- *Tempo*: un numero intero che rappresenta il tempo a cui e stata effettuata la misurazione, espresso in secondi trascorsi a partire da un istante zero in cui viene fatta iniziare la rilevazione.
- *Velocita*: un numero reale che rappresenta la velocita del veicolo (Km/h).
- *Accelerazione*: un numero reale che rappresenta l'accelerazione del veicolo (m/s^2).
- *Giri motore*: un numero intero che rappresenta il numero di giri del motore.
- *Angolo di sterzata*: un numero intero (compreso tra -360 e +360) che rappresenta l'angolo dello sterzo.

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

1. Acquisisce il file e memorizza le relative informazioni in una opportuna struttura dati.
2. Calcola per ognuno dei parametri (escluso il tempo) il valore massimo, minimo, medio e mediano.
3. Ordina il file sulla base di uno dei parametri (escluso il tempo) indicato dall'utente.
4. Ricerca e restituisce il record (ovvero le informazioni contenute in una o più righe del file) in corrispondenza del quale uno dei parametri scelto dall'utente (escluso il tempo) ha avuto un determinato valore. Ad esempio: se l'utente chiede in quali istanti la velocità e uguale a 32.5 Km/h, il programma deve restituire le informazioni contenute nella riga 2 (2, 32.5, 0.32, 950, 0.00).

Per quanto riguarda l'analisi teorica si deve fornire la complessità corrispondente ad ognuna delle seguenti operazioni: calcolo delle statistiche (massimo, minimo, media, mediana); ordinamento; ricerca. Oltre all'analisi teorica della complessità si deve effettuare uno studio sperimentale della stessa per le operazioni specificate al punto precedente. Come suggerimento si può operare generando un numero N di rilevazioni casuali. L'analisi sperimentale deve quindi valutare la complessità al variare di N.

Analisi del problema

L'input è rappresentato dall'insieme dei valori contenuti in un file F, organizzato in righe e colonne. Ogni riga è considerata come una tupla $r = \{t, v, a, g, s\}$, dove $r_n \in F$ con $n \in \mathbb{N}$, dove:

- t: tempo
- v: velocità
- a: accelerazione
- g: giri del motore
- s: angolo di sterzata
- n: numero di elementi

con $t, g \in \mathbb{N}; v \in \mathbb{R}^+; a, s \in \mathbb{R}$

Ogni colonna (t, v, a, g ed s) può essere considerata un insieme definito come:
$C = \{c_1 \dots c_n\}, n \in \mathbb{N}$

L'output è rappresentato da:

- Valore massimo definito come:
  $m, c \in C,\ \forall c \in C : m \geq c;$
- Valore minimo definito come:
  $m, c \in C,\ \forall c \in C : m \leq c;$
- Valore medio definito come
  $$M = \frac{1}{n}\sum_{i=1}^{n} c_i \ con\ i, n \in \mathbb{N}; c_i \in C$$
- Valore mediano definito come:
  $$Me = c_{\frac{n+1}{2}} \qquad \text{se } n \in \mathbb{N} \text{ è dispari}$$
  $$Me = \frac{c_{\frac{n}{2}} + c_{\frac{n}{2}+1}}{2} \qquad \text{se } n \in \mathbb{N} \text{ è pari}$$
- L'insieme dei dati ordinati sulla base di un parametro corrispondente ad una delle colonne.
- Un valore cercato dall'utente.

Le relazioni da sfruttare sono:

- riga-colonna tra i valori t e v, a, g, s.
- Relazione d'ordine totale "$<$" tra elementi dello stesso campo ma di diversi record

Progettazione dell'algoritmo

- **Rappresentazione degli input:**
  - File: una struttura dati dinamica in cui ogni elemento è una variabile strutturata contenente un record di dati
  - Colonna: una variabile di tipo carattere (nel caso dell'algoritmo di ordinamento)
  - Parametri di ricerca: una variabile strutturata, contenente il valore della colonna e quello dell'elemento cercato (nel caso dell'algoritmo di ricerca)
- **Rappresentazione degli output:**
  - Dei messaggi testuali che riportano a schermo i seguenti valori:
    - massimo e records in cui è presente;
    - minimo e records in cui è presente;
    - medio;
    - mediano;
    - records ordinati in base ad una colonna;
    - records in cui appare il valore cercato;
    - errori di inserimento degli input;
    - messaggio di congedo.
- **L'idea alla base della soluzione** è quella di considerare le righe del file come dei record aventi come chiave primaria il *tempo*, mentre ogni campo del record corrisponde ad una colonna. I dati vengono memorizzati in una lista doppiamente collegata, dove ogni elemento è una variabile strutturata, composta a sua volta da più variabili, corrispondenti per numero e tipo ai campi del record (o alle colonne).
- **Scelte di progetto:**
  - Il file in ingresso deve essere nominato "*log*", di tipo testo, con estensione txt.
  - Rappresentazione con lista:

    Si è scelto di utilizzare una lista perché
    1. il numero di dati non è sempre uguale e quindi una struttura dati dinamica è concettualmente più adatta;
    2. il consumo di memoria è più basso. Gli elementi sono rappresentati con variabili strutturate, a loro volta composte da 5 variabili semplici (2 intere e 3 reali). Una variabile semplice occupa 4 byte, invece una strutturata come quelle in questo programma, occupa $4 \cdot 5 = 20$byte (non ho volutamente calcolato le due variabili puntatore necessarie al corretto funzionamento della lista). Usando un array di variabili strutturate, ogni operazione richiederebbe la copia/spostamento di 20 byte per ciascun elemento, contro i 4 della lista (in alcuni casi 8 se è doppiamente collegata), cioè quelli della variabile puntatore che contiene l'indirizzo della locazione di memoria in cui è contenuto l'elemento successivo (risp. precedente).

    Gli elementi della lista vengono memorizzati in base alla relazione d'ordine totale "<" rispetto al tempo.

    La lista è doppiamente collegata per velocizzare l'accesso ad elementi più vicini ad un'estremità piuttosto che ad un'altra, diminuendo le iterazioni.

- Gli algoritmi d'inserimento, rimozione, calcolo della media, mediana, del massimo e del minimo sono stati generalizzati richiamandone all'interno un altro, che estrae il valore corretto rispetto alla colonna prescelta.
- Si è scelto di gestire gli algoritmi per l'acquisizione del file, dei dati da tastiera, dell'ordinamento, ricerca e visualizzazione degli output con algoritmi ad hoc, in modo da rendere il codice più leggibile.
- Si è scelto di inserire un menu in modo che l'utente possa scegliere anche più di una volta l'operazione che preferisce e quando terminare l'esecuzione.
- Affinché le funzioni accettino valori da ogni campo del record, si è scelto di dichiarare *float* tutte le variabili locali che contengono i valori numerici dei campi dei record. Il float può contenere anche l'intero, mentre è impossibile il contrario. In caso l'output richiesto sia invece di tipo intero, viene effettuato un cast della variabile.
- Acquisizione file:

  Legge e salta la prima riga perché non contiene valori utili.

  Acquisisce e memorizza i record in una variabile strutturata temporanea, 1 record per volta. Il contenuto della suddetta variabile verrà poi inserito nella lista per mezzo dell'apposita funzione di inserimento.
- Inserimento elemento:

  Scorre linearmente la lista e inserisce l'elemento in input nella giusta posizione in modo da mantenere l'ordinamento della lista. Ritorna una variabile flag per segnalare il buon (risp. cattivo) esito dell'operazione.
- Selezione dei valori:

  Poiché ogni elemento della lista è costituito da una variabile strutturata, composta a sua volta da più variabili semplici, l'algoritmo seleziona e restituisce il valore corrispondente alla colonna indicata in input.
- Ordinamento:

  Scorre linearmente la lista in input e inserisce gli elementi in una seconda lista ordinata, scorsa anch'essa linearmente per mezzo dell'algoritmo di inserimento. Contemporaneamente allo scorrimento, elimina l'ultimo elemento visitato della lista originaria. Ritorna in output il puntatore alla testa della nuova lista.

  Una variabile tiene traccia della colonna in base alla quale la lista è attualmente ordinata, in modo da evitare esecuzioni superflue dell'algoritmo di ordinamento, cioè quando la lista è già ordinata.
- Calcolo Massimo e Minimo:

  L'algoritmo del massimo e minimo è rappresentato da singole istruzioni di assegnazione, perché la lista viene preventivamente ordinata, e il minimo (risp. massimo) si trova nella testa (risp. coda). Essendo la lista doppiamente collegata, è sufficiente accedere direttamente ad un estremo. Inoltre l'ordinamento e il doppio collegamento permettono di scorrere e mostrare, con il minor numero di accessi, tutti i record in cui è presente il valore minimo (risp. massimo).

  In caso di scorrimento dei record contenenti il valore massimo (risp. minimo), l'algoritmo inizia a scorrere dalla coda (risp. testa).
- Calcolo della media:

  Scorre linearmente la lista, sommando tutti i valori della colonna prescelta, contenuti negli elementi e dividendo il valore risultante per il numero totale di elementi della lista.

- Calcolo della mediana:
  L'algoritmo, in caso di numero di elementi...
  - dispari: scorre linearmente la lista fino al valore $\frac{n+1}{2}$ e lo restituisce in output
  - pari: scorre linearmente la lista fino al valore $\frac{n}{2}+1$ e ritorna il risultato di: $\frac{\frac{n}{2}+\left(\frac{n}{2}+1\right)}{2}$
- Visita della lista:
  Scorre linearmente la lista, mostrando a schermo il contenuto di ogni elemento e formattandolo in modo da renderne più agevole la lettura.
- Trova coda della lista:
  Scorre linearmente la lista fino all'ultimo elemento e ne memorizza il valore in una variabile.
- Conversione identificatore di colonna:
  Converte l'input da tastiera, che identifica la colonna, da lettera a numero.

- **I passi dell'algoritmo possono essere riassunti come segue:**
  1. Acquisire i dati da un file e memorizzarli in una lista ordinata;
  2. Calcolare i seguenti valori divisi per colonna: mediano, massimo, minimo, medio;
  3. Stampare a video i valori appena calcolati, con i rispettivi record, ove possibile;
  4. Mostrare a video un menu con più operazioni tra cui scegliere;
  5. Acquisire il numero della operazione;
  6. Eseguire l'algoritmo corrispondente;
     - 6.1. Acquisire da tastiera la lettera identificativa di una colonna
     - 6.2. Ordinare in base alla colonna prescelta;
     - 6.3. Stampare a video la lista ordinata;
     - 6.4. Acquisire da tastiera la lettera rappresentante la scelta dell'utente;
     - 6.5. Se è "*S*" o "*s*", ritorna al passo 6.1, altrimenti continua;
     - 6.6. Tornare al passo 4;

       Oppure
     - 6.7. Acquisire da tastiera la lettera rappresentante la scelta dell'utente;
     - 6.8. Se è "*S*" o "*s*", ritorna al passo 4, altrimenti continua;
     - 6.9. Acquisire da tastiera la lettera identificativa di una colonna;
     - 6.10. Acquisire da tastiera il valore da cercare;
     - 6.11. Stampare a video il valore cercato e relativi record, se presenti;
     - 6.12. Acquisire da tastiera la lettera rappresentante la scelta dell'utente;
     - 6.13. Se è "*S*" o "*s*", ritorna al passo 6.7, altrimenti continua;
     - 6.14. Tornare al passo 4;

       Oppure
     - 6.15. Stampare a video la lista;
     - 6.16. Tornare al passo 4;

       Oppure
     - 6.17. Stampare a video un messaggio di congedo;
  7. Terminare l'esecuzione.

Implementazione dell'algoritmo

```
/*****************************************************************/
/* programma per la gestione della telemetria */
/*****************************************************************/

/*****************************/
/* inclusione delle librerie */
/*****************************/

#include <stdio.h>
#include <stdlib.h>

/***********************************/
/* definizione delle costanti */
/***********************************/

#define LUNG_RIGA 70

/***********************************/
/* definizione dei tipi */
/***********************************/

typedef struct elem_lista
    {
    int tempo,
        giri;
    float velocita,
          accelerazione,
          sterzata;
    struct elem_lista *succ_p, *prec_p;
    } elem_lista_t;

typedef struct elem_ricerca
    {
    int colonna;
    float cercato;
    } elem_ricerca_t;

/***********************************/
/* dichiarazione delle funzioni */
/***********************************/

int acquisisciFile(elem_lista_t**);
int inserisciElemento(elem_lista_t**,
                      int,
                      elem_lista_t*);
float selezionaValori(elem_lista_t*,
                      int);
void visitaLista(elem_lista_t*);
elem_lista_t* ordinaLista(elem_lista_t*,
                          int);
elem_lista_t* codaLista(elem_lista_t*);
void mostraRisultati(elem_lista_t*,
                     elem_lista_t*,
                     int,
                     int);
float medianoLista(elem_lista_t*,
                   int,
                   int);
float mediaLista(elem_lista_t*,
                 int,
                 int);
void mostraRecord(elem_lista_t*,
                  int,
                  int,
                  float,
                  int);
int menuLista();
int gestioneOrdinamento(elem_lista_t**,
                        elem_lista_t**,
                        int,
                        int);
int datiOrdinamento();
int convertiColonna(char);
void gestioneRicerca(elem_lista_t*);
void datiRicerca(elem_ricerca_t*);
```

```
elem_lista_t* cercaElemento(elem_lista_t*,
                            int,
                            float);

/**********************************/
/* definizione delle funzioni */
/**********************************/

/** FUNZIONE MAIN **/
int main(void)
{
    /* dichiarazione variabili */
    elem_lista_t *testa_p = NULL,       /* lavoro: testa della lista */
                 *coda_p = NULL;        /* lavoro: coda della lista */
    int ordine_corr = 1,                /* lavoro: tiene traccia della colonna ordinata */
        lunghezza_lista = 0,            /* lavoro: numero di elementi nella lista */
        colonna,                        /* lavoro: contatore delle colonne */
        scelta;                         /* lavoro: scelta dell'utente tra le opzioni del menu */

    /* acquisisce i dati dal file e ne restituisce il numero */
    lunghezza_lista = acquisisciFile(&testa_p);

    if (lunghezza_lista > 0)
        {
         printf("\n*** VISUALIZZA I DATI ***\n\n");

         /* visualizza i dati a schermo */
         visitaLista(testa_p);

         for(colonna = 2; colonna <= 5; colonna++)
             {
               switch(colonna)
                {
                case 2:
                    printf("\n ----- VALORI VELOCITA' ----- \n");
                    break;
                case 3:
                    printf("\n ----- VALORI ACCELERAZIONE ----- \n");
                    break;
                case 4:
                    printf("\n ----- VALORI GIRI ----- \n");
                    break;
                case 5:
                    printf("\n ----- VALORI ANGOLO DI STERZATA ----- \n");
                    break;
                }

               /* ordina la lista in base alla colonna selezionata */
               testa_p = ordinaLista(testa_p,
                                     colonna);

               ordine_corr = colonna;

               /* memorizza la nuova coda */
               coda_p = codaLista(testa_p);

               /* visualizza a schermo tutti i valori calcolati divisi per colonna */
               mostraRisultati(testa_p,
                               coda_p,
                               colonna,
                               lunghezza_lista);
             }

        do {
            /* mostra a video il menu con le possibili operazioni */
            scelta = menuLista();

            switch(scelta)
                 {
                 case 1:
                    printf("\n\n*** ORDINA I DATI ***\n");
                    /* acquisizione dei dati e ordinamento */
                    ordine_corr = gestioneOrdinamento(&testa_p,
                                                      &coda_p,
                                                      lunghezza_lista,
                                                      ordine_corr);
                    break;
                 case 2:
```

```
                    printf("\n\n*** CERCA UN VALORE ***\n");
                    /* acquisizione dei dati e ricerca */
                    gestioneRicerca(testa_p);
                    break;
                case 3:
                    visitaLista(testa_p);
                    break;
                case 4:
                    printf("\nBuona giornata\n");
                }

            }while(scelta != 4);
        }

    return 0;
}

/** ACQUISISCI DATI DA FILE **/
/* acquisisce i dati da file, li inserisce in una
   struttura dati, poi in una lista ordinata
   e infine restituisce la lunghezza della lista */
int acquisisciFile(elem_lista_t **testa_p)  /* input: puntatore alla testa della lista */
{
    FILE *file_input;           /* lavoro: file con i dati di telemetria */
    elem_lista_t valori_input;  /* lavoro: contiene i dati di una riga */
    int lunghezza = 0;          /* output: totale elementi inseriti nella lista */
    char riga[LUNG_RIGA];       /* lavoro: contiene i caratteri di una riga */

    /* apre il file in lettura e assegna un puntatore */
    file_input = fopen("log.txt",
                       "r");

    /* in caso di errore mostra un messaggio*/
    if (!file_input)
        printf("Errore apertura file\n");
    else
        {/* legge (e salta) la prima riga del file */
        if (fgets(riga,
                  LUNG_RIGA,
                  file_input) != NULL)
            {
             do
                {/* memorizza i dati di una riga del file in "valori_input" */
                if (fscanf(file_input,
                           "%d%f%f%d%f",
                           &valori_input.tempo,
                           &valori_input.velocita,
                           &valori_input.accelerazione,
                           &valori_input.giri,
                           &valori_input.sterzata) > 0)
                    { /* inserisce i dati in un nuovo elemento della lista */
                     if (inserisciElemento(&(*testa_p),
                                           1,
                                           &valori_input) != 1)
                        lunghezza = 0;
                     else /* se non ci sono stati errori, incrementa il contatore */
                        lunghezza++;
                    }
                /* ripete finchè non è stata raggiunta la fine del file o in caso di errore */
                }while (!feof(file_input) && lunghezza != 0);
            }
        /* chiude il file */
        fclose(file_input);
        }

    return(lunghezza);
}

/** INSERISCI ELEMENTO **/
/* inserisce l'elemento nella giusta
   posizione nella lista */
int inserisciElemento(elem_lista_t **testa_p,  /* input: testa della lista */
                      int colonna,             /* input: valore numerico della colonna */
                      elem_lista_t *input_p)   /* input: struttura dei valori da inserire */
{
    int inserito;           /* output: posizione di inserimento */
    elem_lista_t *corr_p,   /* lavoro: tiene traccia della posizione durante lo scorrimento */
                 *prec_p,   /* lavoro: elemento precedente */
```

```
                *nuovo_p; /* lavoro: elemento nuovo */

    /* scorre la lista finché l'elemento corrente è nullo
       oppure il valore in lista è minore di quello di
       input (relazione d'ordine "<") */
    for (corr_p = prec_p = *testa_p;
         ((corr_p != NULL) &&
          (selezionaValori(corr_p,colonna) < selezionaValori(input_p,colonna)));
         prec_p = corr_p,corr_p = corr_p->succ_p);

    /* se l'elemento è nullo o se esiste già un record con lo stesso valore di tempo */
    if ((corr_p != NULL) && (corr_p->tempo == input_p->tempo))
        inserito = 0;
    else
        {
        inserito = 1;
        nuovo_p = (elem_lista_t *)malloc(sizeof(elem_lista_t));
        nuovo_p->tempo = input_p->tempo;
        nuovo_p->velocita = input_p->velocita;
        nuovo_p->accelerazione = input_p->accelerazione;
        nuovo_p->giri = input_p->giri;
        nuovo_p->sterzata = input_p->sterzata;
        nuovo_p->succ_p = corr_p;

        if (corr_p == *testa_p)
            {
            nuovo_p->prec_p = NULL;
            *testa_p = nuovo_p;
            }
        else
            {
            prec_p->succ_p = nuovo_p;
            nuovo_p->prec_p = prec_p;
            }
        }

    return(inserito);
}

/** SELEZIONA VALORI **/
/* seleziona il giusto valore estratto
   dalla variabile strutturata in input,
   in base alla colonna indicata */
float selezionaValori(elem_lista_t *elem_p, /* input: puntatore ad una variabile strutturata*/
                      int colonna)          /* input: identificatore numerico della colonna */
{
    float valore; /* output: valore della colonna desiderata */

    valore = 0.0;

     switch(colonna)
        {
        case 1:
            valore = elem_p->tempo;
            break;
        case 2:
            valore = elem_p->velocita;
            break;
        case 3:
            valore = elem_p->accelerazione;
            break;
        case 4:
            valore = elem_p->giri;
            break;
        case 5:
            valore = elem_p->sterzata;
            break;
        }

    return(valore);
}

/** VISITA LISTA **/
/* scorre la lista partendo da un'estremità
   e stampando a video tutti i record*/
void visitaLista(elem_lista_t *testa_p) /* input: puntatore alla testa della lista */
{
    elem_lista_t *elem_p;
```

```
    printf("Tempo(A)\tVelocita(B)\tAccel.(C)\tGiri(D)\t\tAngolo di sterzata(E)\n");

    for (elem_p = testa_p;
         elem_p != NULL;
         elem_p = elem_p->succ_p)
         printf("%d\t\t%.2f\t\t%.2f\t\t%d\t\t%.2f\n",
                elem_p->tempo,
                elem_p->velocita,
                elem_p->accelerazione,
                elem_p->giri,
                elem_p->sterzata);
}

/** ORDINA LISTA  **/
/* ordina la lista spostandoli ordinatamente
   dalla lista primaria in una di appoggio */
elem_lista_t* ordinaLista(elem_lista_t *testa_p,     /* input: testa della lista */
                          int colonna)               /* input: valore numerico di colonna */
{
    elem_lista_t *elem_p,
                 *lista2;

    lista2 = NULL;

    for (elem_p = testa_p;
         elem_p != NULL;
         elem_p = elem_p->succ_p)
         {
          /* inserisce l'elemento nella lista d'appoggio */
          inserisciElemento(&lista2,
                            colonna,
                            elem_p);

          /* dealloca la memoria */
          free(elem_p);
         }

    return(lista2);
}

/** TROVA CODA DELLA LISTA **/
/* scorre linearmente tutti gli elementi
   della lista fino alla fine e memorizza
   il valore dell'ultimo elemento */
elem_lista_t* codaLista(elem_lista_t *testa_p) /* input: puntatore alla testa della lista */
{
    elem_lista_t *elem_p;   /* output: elemento della lista */

    for (elem_p = testa_p;
         elem_p != NULL && elem_p->succ_p != NULL;
         elem_p = elem_p->succ_p);

    return(elem_p);
}

/** MOSTRA RISULTATI **/
/* mostra a video i valori massimi, minimi,
   medi, mediani con i rispettivi record */
void mostraRisultati(elem_lista_t *testa_p, /* input: puntatore alla testa della lista */
                     elem_lista_t *coda_p,  /* input: puntatore alla coda della lista */
                     int colonna,           /* input: valore numerico di colonna */
                     int lunghezza_lista)   /* input: numero di elementi nella lista */
{
  float massimo,    /* lavoro: valore massimo della colonna */
        minimo,     /* lavoro: valore minimo della colonna */
        media,      /* lavoro: valore medio della colonna */
        mediano;    /* lavoro: valore mediano della colonna */

  /* seleziona il valore massimo della colonna indicata */
  massimo = selezionaValori(coda_p,
                            colonna);

  /* seleziona il valore minimo della colonna indicata */
  minimo = selezionaValori(testa_p,
                           colonna);

  /* restituisce la media dei valori della colonna indicata */
```

```
  media = mediaLista(testa_p,
                     lunghezza_lista,
                     colonna);

  /* restituisce la mediana dei valori della colonna indicata */
  mediano = medianoLista(testa_p,
                         lunghezza_lista,
                         colonna);

  /* ----- MEDIANO E MASSIMO ----- */
  if (colonna != 4)
      {
       printf("\nMediano: %.2f\n",mediano);
       printf("\nMassimo: %.2f\n",massimo);
      }
  else
      {
       printf("\nMediano: %d\n",(int)mediano);
       printf("\nMassimo: %d\n",(int)massimo);
      }

  /* restituisce tutti i record in cui è presente il valore massimo */
  mostraRecord(coda_p,
               colonna,
               1,
               massimo,
               1);

  /* ----- MEDIO E MINIMO ----- */
  if (colonna != 4)
      {
       printf("\nMedio: %.2f\n",media);
       printf("\nMinimo: %.2f\n",minimo);
      }
  else
      {
       printf("\nMedio: %d\n",(int)media);
       printf("\nMinimo: %d\n",(int)minimo);
      }

  /* restituisce tutti i record in cui è presente il valore minimo */
  mostraRecord(testa_p,
               colonna,
               1,
               minimo,
               0);
}

/** CALCOLA MEDIA **/
float mediaLista(elem_lista_t *testa_p, /* input: puntatore alla testa della lista */
                 int lunghezza_lista,   /* input: numero di elementi attualmente in lista */
                 int colonna)           /* input: valore numerico della colonna */
{
    elem_lista_t *elem_p;   /* lavoro: puntatore all'elemento di appoggio */
    float media;            /* output: memorizza il risultato del calcolo*/

    media = 0.0;

    for (elem_p = testa_p;
         elem_p != NULL;
         elem_p = elem_p->succ_p)
         media += selezionaValori(elem_p,colonna);

    media = media / ((float)lunghezza_lista);

    return(media);
}

/** CALCOLA MEDIANO **/
float medianoLista(elem_lista_t *testa_p,   /* input: puntatore alla testa della lista */
                   int lunghezza_lista,     /* input: numero di elementi della lista */
                   int colonna)             /* input: valore numerico della colonna */
{
    elem_lista_t *elem_p;   /* lavoro: puntatore all'elemento di appoggio */
    int contatore = 0,      /* lavoro: conta gli elementi */
        termina = 0;        /* lavoro: segnala che la mediana è stata trovata */
    float valore1 = 0.0,    /* lavoro: 1° valore in caso di lista pari */
          valore2 = 0.0,    /* lavoro: 2° valore in caso di lista pari */
```

```
          mediana = 0.0;     /* output: memorizza il risultato del calcolo */

        for (elem_p = testa_p;
             elem_p != NULL && termina == 0;
             elem_p = elem_p->succ_p)
             {
              contatore++;
              if ((lunghezza_lista % 2) != 0)
                  {/* In caso la lista contenga un numero dispari di elementi */
                   if (contatore == (lunghezza_lista + 1) / 2)
                       {
                        mediana = selezionaValori(elem_p,
                                                  colonna);
                        termina = 1;
                       }
                  }
              else
                  {/* In caso la lista contenga un numero pari di elementi */
                   if (contatore == lunghezza_lista / 2)
                       {
                       valore1 = selezionaValori(elem_p,
                                                 colonna);

                       valore2 = selezionaValori(elem_p->succ_p,
                                                 colonna);

                       mediana = (valore1 + valore2) / 2;

                       termina = 1;
                       }
                  }
             }

    return(mediana);
}

/** MOSTRA RECORD **/
/* cerca e restituisce tutti i record
   in cui si trova l'elemento */
void mostraRecord(elem_lista_t *estremo,/* input: puntatore ad una delle estremità  */
                  int colonna,          /* input: valore numerico di colonna */
                  int maxmin,           /* input: indica se si sta mostrando solo i valori max/min*/
                  float cercato,        /* input: valore cercato */
                  int inverti)          /* input: flag per capire in quale direzione scorrere */
{
    elem_lista_t *elem_p = estremo; /* lavoro: puntatore all'elemento di appoggio */
    int contatore = 0,              /* lavoro: conta i record trovati */
        termina = 0;                /* lavoro: flag per capire se uscire dal ciclo */

    while(elem_p != NULL && termina == 0)
        {
        if (selezionaValori(elem_p,colonna) == cercato)
            {
             contatore++;
             /* stampa a video i record dove è presente il valore cercato */
             printf("%d(%d,%.2f,%.2f,%d,%.2f)\n",
                    contatore,
                    elem_p->tempo,
                    elem_p->velocita,
                    elem_p->accelerazione,
                    elem_p->giri,
                    elem_p->sterzata);
            }
        else if (maxmin)
                 termina = 1;

        elem_p = (inverti) ? elem_p->prec_p : elem_p->succ_p;
        }
}

/** MENU **/
int menuLista()
{
    int scelta,         /* output: scelta dell'utente */
        esito_lettura;  /* lavoro: validazione input */

    printf("\n\n\t***MENU OPERAZIONI***\n");
```

```
    do {
        esito_lettura = scelta = 0;

        printf("\nInserire il numero dell'operazione desiderata:\n");
        printf("\t1 - Ordinare la lista\n");
        printf("\t2 - Cercare un elemento\n");
        printf("\t3 - Visualizzare la lista\n");
        printf("\t4 - Terminare l'esecuzione\n");
        esito_lettura = scanf("%d",
                              &scelta);

        while (getchar() != '\n');

       }while(esito_lettura != 1 && (scelta < 1 || scelta > 4));

    return(scelta);
}

/** GESTISCE GLI ALGORITMI PER L'ORDINAMENTO **/
int gestioneOrdinamento(elem_lista_t **testa_p, /* input: testa della lista */
                        elem_lista_t **coda_p,  /* input: coda della lista */
                        int lunghezza_lista,    /* input: lunghezza della lista */
                        int ordine_corr)        /* input/output: ordinamento corrente */
{
    int esito_lettura,  /* lavoro: validazione */
        colonna;        /* lavoro: valore numerico della colonna da ordinare */
    char scelta;        /* lavoro: scelta dell'utente */

    /* acquisisce i parametri dall'utente */
     do {
         colonna = datiOrdinamento();

         if (colonna != ordine_corr && lunghezza_lista > 1)
             {
               /* ordina la lista in base alla colonna desiderata */
               *testa_p = ordinaLista(*testa_p,
                                      colonna);

               ordine_corr = colonna;

               /* memorizza la nuova coda */
               *coda_p = codaLista(*testa_p);
             }

        /* visualizza i dati appena ordinati */
        visitaLista(*testa_p);

        do {
            while (getchar() != '\n');
            printf("\nSi desidera effettuare un altro ordinamento? [S/n]\n");
            esito_lettura = scanf("%c",
                                  &scelta);

           } while(esito_lettura != 1);

       while (getchar() != '\n');

     } while(scelta == 'S' || scelta == 's');

    return(ordine_corr);
}

/** ACQUISIZIONE PARAMETRI DI ORDINAMENTO **/
/* richiede, memorizza e restituisce
   i dati di input da tastiera */
int datiOrdinamento()
{
    char lettera_colonna;   /* lavoro: valore letterale di colonna */
    int esito_lettura,      /* lavoro: variabile per la validazione  */
        colonna;            /* output: valore numerico di colonna */

    do
        {
         printf("\nInserire la colonna in base alla quale effettuare l'ordinamento: B, C, D, E\n");
         esito_lettura = scanf("%1c",
                               &lettera_colonna);
```

```
         /* converte il valore letterale della colonna in numerico */
         colonna = convertiColonna(lettera_colonna);

         if (esito_lettura != 1 || colonna == 0)
            {
             while (getchar() != '\n');
             printf("Errore: input non accettabile\n");
             printf("\nPremere Invio per continuare...\n");
             while (getchar() != '\n');
            }

        }while (esito_lettura != 1 || colonna == 0);

    return(colonna);
}

/** CONVERTI COLONNA **/
/* converte l'identificatore della colonna
   da lettera a numero */
int convertiColonna(char lettera_colonna)   /* input: identificatore letterale della colonna */
{
    int colonna;    /* output: identificatore numerico della colonna */

     switch(lettera_colonna)
        {
        case 'B':
        case 'b':
            colonna = 2;
            break;
        case 'C':
        case 'c':
            colonna = 3;
            break;
        case 'D':
        case 'd':
            colonna = 4;
            break;
        case 'E':
        case 'e':
            colonna = 5;
            break;
        default:
            colonna = 0;
            break;
        }

    return(colonna);
}

/** GESTISCE GLI ALGORITMI DI RICERCA **/
void gestioneRicerca(elem_lista_t *testa_p) /* input: testa della lista */
{
    elem_ricerca_t param_ricerca;   /* lavoro: parametri di ricerca */
    int esito_lettura;              /* lavoro: validazione */
    char scelta;                    /* lavoro: scelta dell'utente */

    do {
         esito_lettura = 0;
         /* acquisisce i parametri dall'utente */
         datiRicerca(&param_ricerca);

         /* restituisce l'elemento cercato, se presente */
         if (cercaElemento(testa_p,
                           param_ricerca.colonna,
                           param_ricerca.cercato) != NULL)
             /* mostra tutti i record in cui appare l'elemento cercato */
             mostraRecord(testa_p,
                          param_ricerca.colonna,
                          0,                        /* flag maxmin */
                          param_ricerca.cercato,
                          0);                       /* flag inverti */
         else
            printf("Elemento non trovato\n");

         do {
             printf("\nSi desidera effettuare un'altra ricerca? [S/n]\n");
             esito_lettura = scanf("%c",
                                   &scelta);
```

```
            } while(esito_lettura != 1);

            while (getchar() != '\n');

        } while (scelta == 'S' || scelta == 's');
}

/** ACQUISIZIONE PARAMETRI DI RICERCA **/
/* richiede, memorizza e restituisce
   i dati di input da tastiera */
void datiRicerca(elem_ricerca_t *param_ricerca) /* input/output: contiene i dati per la ricerca */
{
    int esito_lettura;      /* lavoro: variabile per la validazione */
    char lettera_colonna;   /* lavoro: valore letterale della colonna */

    do
        {
         printf("\nInserire la colonna nella quale cercare il valore: B, C, D, E\n");
         esito_lettura = scanf("%1s",
                               &lettera_colonna);

         /* converte il valore letterale della colonna in numerico */
         param_ricerca->colonna = convertiColonna(lettera_colonna);

         if (esito_lettura < 1 || param_ricerca->colonna == 0)
             {
              printf("Errore: input non accetabile\n");
              printf("\n\nPremere Invio per continuare...\n");
              while (getchar() != '\n');
             }

         while (getchar() != '\n');

        }while (esito_lettura < 1 || param_ricerca->colonna == 0);

    do
        {
         printf("\nInserire il valore da cercare: \n");
         esito_lettura = scanf("%f",
                               &param_ricerca->cercato);

         if (esito_lettura < 1)
             {
              printf("Errore: input non accetabile\n");
              printf("\nPremere Invio per continuare...\n");
              while (getchar() != '\n');
             }

         while (getchar() != '\n');

        }while (esito_lettura < 1);
}

/** CERCA ELEMENTO **/
/* cerca e restituisce tutti i record
   in cui si trova l'elemento */
elem_lista_t* cercaElemento(elem_lista_t *testa_p,     /* input: puntatore ad una delle estremitÃ  */
                            int colonna,               /* input: valore numerico di colonna */
                            float cercato)             /* input: valore cercato */
{
    elem_lista_t *elem_p;   /* output: puntatore all'elemento di appoggio */

    for(elem_p = testa_p;
        elem_p != NULL && selezionaValori(elem_p,colonna) != cercato;
        elem_p = elem_p->succ_p);

    return(elem_p);
}
```

```
#*****************************************
# MakeFile Telemetria
#*****************************************

telemetria: telemetria.c makefile
	gcc -ansi -Wall -O telemetria.c -o telemetria

clean:
	rm -f telemetria.o
```

Testing del programma

*** VISUALIZZA I DATI ***

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
1	45.06	0.28	1086	244.94
2	44.32	-0.16	1035	296.39
3	40.91	-0.02	921	-160.00
4	47.95	0.45	1059	-97.36
5	40.37	-0.36	926	97.71
6	45.20	-0.36	868	-348.26
7	28.17	-0.37	1030	-247.19
8	46.06	0.01	1035	359.23
9	38.35	0.14	1058	81.10
10	44.40	-0.21	1056	-4.62
11	47.34	-0.10	1073	19.26
12	48.02	0.31	837	-156.01
13	27.61	0.03	1070	-309.78
14	34.05	0.39	880	-221.61
15	27.05	-0.04	970	-313.80
16	46.35	0.40	1025	-188.44
17	44.13	-0.12	805	-168.00
18	26.46	0.03	1057	9.03
19	43.16	0.43	845	-44.90
20	34.17	0.14	964	-155.31
21	47.06	-0.06	908	135.26
22	47.39	-0.27	884	237.02
23	39.92	0.46	999	-107.74
24	48.14	-0.06	968	113.26
25	47.82	0.18	986	-73.13

----- VALORI VELOCITA' -----

Mediano: 44.32

Massimo: 48.14
1(24,48.14,-0.06,968,113.26)

Medio: 41.18

Minimo: 26.46
1(18,26.46,0.03,1057,9.03)

----- VALORI ACCELERAZIONE -----

Mediano: 0.01

Massimo: 0.46
1(23,39.92,0.46,999,-107.74)

Medio: 0.04

Minimo: -0.37
1(7,28.17,-0.37,1030,-247.19)

----- VALORI GIRI -----

Mediano: 986

Massimo: 1086
1(1,45.06,0.28,1086,244.94)

Medio: 973

Minimo: 805
1(17,44.13,-0.12,805,-168.00)

----- VALORI ANGOLO DI STERZATA -----

Mediano: -73.13

Massimo: 359.23
1(8,46.06,0.01,1035,359.23)

Medio: -40.12

Minimo: -348.26
1(6,45.20,-0.36,868,-348.26)

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
1

*** ORDINA I DATI ***

Inserire la colonna in base alla quale effettuare l'ordinamento: B, C, D, E
B

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
18	26.46	0.03	1057	9.03
15	27.05	-0.04	970	-313.80
13	27.61	0.03	1070	-309.78
7	28.17	-0.37	1030	-247.19
14	34.05	0.39	880	-221.61
20	34.17	0.14	964	-155.31
9	38.35	0.14	1058	81.10
23	39.92	0.46	999	-107.74
5	40.37	-0.36	926	97.71
3	40.91	-0.02	921	-160.00
19	43.16	0.43	845	-44.90
17	44.13	-0.12	805	-168.00
2	44.32	-0.16	1035	296.39
10	44.40	-0.21	1056	-4.62
1	45.06	0.28	1086	244.94
6	45.20	-0.36	868	-348.26
8	46.06	0.01	1035	359.23
16	46.35	0.40	1025	-188.44
21	47.06	-0.06	908	135.26
11	47.34	-0.10	1073	19.26
22	47.39	-0.27	884	237.02
25	47.82	0.18	986	-73.13

```
4          47.95        0.45         1059         -97.36
12          48.02        0.31         837         -156.01
24          48.14        -0.06        968          113.26

Vuoi effettuare un altro ordinamento? [S/n]
n


      ***MENU OPERAZIONI***

Inserire il numero dell'operazione desiderata:
      1 - Ordinare la lista
      2 - Cercare un elemento
      3 - Visualizzare la lista
      4 - Terminare l'esecuzione
2


*** CERCA UN VALORE ***

Inserire la colonna nella quale cercare il valore: B, C, D, E
E

Inserire il valore da cercare:
81.10
1(9,38.35,0.14,1058,81.10)

Si desidera effettuare un'altra ricerca? [S/n]
n


      ***MENU OPERAZIONI***

Inserire il numero dell'operazione desiderata:
      1 - Ordinare la lista
      2 - Cercare un elemento
      3 - Visualizzare la lista
      4 - Terminare l'esecuzione
4

Buona giornata
```

*** VISUALIZZA I DATI ***

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
1	45.31	-0.31	967	-359.10
2	47.45	-0.15	924	61.21
3	46.54	-0.33	964	232.44
4	25.87	-0.20	827	151.56
5	29.56	-0.35	1042	-294.19
6	25.61	-0.38	1004	351.74
7	39.49	0.03	882	-353.58
8	41.74	-0.33	895	73.27
9	40.39	-0.44	938	-35.43
10	32.90	0.02	999	203.99
11	48.18	0.46	911	270.70
12	31.27	-0.04	964	28.33
13	46.17	0.28	1068	260.81
14	35.11	0.11	857	357.69
15	26.08	0.34	941	-168.33
16	26.88	0.18	835	-89.38
17	32.19	-0.22	906	-353.67
18	43.30	0.34	948	63.29
19	36.98	0.24	929	-10.84
20	28.15	0.24	1001	-30.27
21	40.42	0.24	823	71.31
22	31.02	-0.35	940	52.13
23	49.75	0.02	1008	-53.89
24	41.60	-0.33	1023	181.12
25	37.87	0.20	1082	-5.83
26	47.68	-0.36	1015	-253.80
27	27.22	-0.07	830	138.88
28	46.99	-0.35	986	335.96
29	29.86	-0.31	972	231.61
30	37.85	-0.34	1073	228.38
31	39.43	-0.22	890	167.05
32	37.14	0.22	967	131.21
33	26.36	0.33	852	-271.43
34	28.06	-0.07	924	12.25
35	33.98	0.05	991	323.52
36	33.26	0.35	1087	-20.36
37	32.80	0.48	1009	-31.61
38	44.15	-0.30	1088	172.22
39	47.31	0.00	930	244.40
40	26.74	0.07	991	-340.22
41	40.86	0.34	859	22.56
42	28.52	0.34	983	113.48
43	48.56	-0.19	850	-280.85
44	43.46	-0.36	1020	-154.02
45	43.81	0.10	899	240.93
46	26.99	-0.50	1034	-178.04
47	28.33	-0.29	838	220.49
48	36.64	-0.39	967	38.31
49	27.31	0.04	1083	181.60
50	32.61	0.20	817	-45.56

----- VALORI VELOCITA' -----

Mediano: 36.81

Massimo: 49.75
1(23,49.75,0.02,1008,-53.89)

Medio: 36.52

Minimo: 25.61
1(6,25.61,-0.38,1004,351.74)

----- VALORI ACCELERAZIONE -----

Mediano: -0.02

Massimo: 0.48
1(37,32.80,0.48,1009,-31.61)

Medio: -0.04

Minimo: -0.50
1(46,26.99,-0.50,1034,-178.04)

----- VALORI GIRI -----

Mediano: 964

Massimo: 1088
1(38,44.15,-0.30,1088,172.22)

Medio: 952

Minimo: 817
1(50,32.61,0.20,817,-45.56)

----- VALORI ANGOLO DI STERZATA -----

Mediano: 56.67

Massimo: 357.69
1(14,35.11,0.11,857,357.69)

Medio: 36.64

Minimo: -359.10
1(1,45.31,-0.31,967,-359.10)

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento

```
    3 - Visualizzare la lista
    4 - Terminare l'esecuzione
2


*** CERCA UN VALORE ***

Inserire la colonna nella quale cercare il valore: B, C, D, E
B

Inserire il valore da cercare:
41.60
1(24,41.60,-0.33,1023,181.12)

Si desidera effettuare un'altra ricerca? [S/n]
n


    ***MENU OPERAZIONI***

Inserire il numero dell'operazione desiderata:
    1 - Ordinare la lista
    2 - Cercare un elemento
    3 - Visualizzare la lista
    4 - Terminare l'esecuzione
1


*** ORDINA I DATI ***

Inserire la colonna in base alla quale effettuare l'ordinamento: B, C, D, E
k
Errore: input non accettabile

Premere Invio per continuare...


Inserire la colonna in base alla quale effettuare l'ordinamento: B, C, D, E
d
Tempo(A)      Velocita(B)    Accel.(C)     Giri(D)       Angolo di sterzata(E)
50            32.61          0.20          817           -45.56
21            40.42          0.24          823           71.31
4             25.87          -0.20         827           151.56
27            27.22          -0.07         830           138.88
16            26.88          0.18          835           -89.38
47            28.33          -0.29         838           220.49
43            48.56          -0.19         850           -280.85
33            26.36          0.33          852           -271.43
14            35.11          0.11          857           357.69
41            40.86          0.34          859           22.56
7             39.49          0.03          882           -353.58
31            39.43          -0.22         890           167.05
8             41.74          -0.33         895           73.27
45            43.81          0.10          899           240.93
```

17	32.19	-0.22	906	-353.67
11	48.18	0.46	911	270.70
2	47.45	-0.15	924	61.21
34	28.06	-0.07	924	12.25
19	36.98	0.24	929	-10.84
39	47.31	0.00	930	244.40
9	40.39	-0.44	938	-35.43
22	31.02	-0.35	940	52.13
15	26.08	0.34	941	-168.33
18	43.30	0.34	948	63.29
3	46.54	-0.33	964	232.44
12	31.27	-0.04	964	28.33
32	37.14	0.22	967	131.21
48	36.64	-0.39	967	38.31
1	45.31	-0.31	967	-359.10
29	29.86	-0.31	972	231.61
42	28.52	0.34	983	113.48
28	46.99	-0.35	986	335.96
35	33.98	0.05	991	323.52
40	26.74	0.07	991	-340.22
10	32.90	0.02	999	203.99
20	28.15	0.24	1001	-30.27
6	25.61	-0.38	1004	351.74
23	49.75	0.02	1008	-53.89
37	32.80	0.48	1009	-31.61
26	47.68	-0.36	1015	-253.80
44	43.46	-0.36	1020	-154.02
24	41.60	-0.33	1023	181.12
46	26.99	-0.50	1034	-178.04
5	29.56	-0.35	1042	-294.19
13	46.17	0.28	1068	260.81
30	37.85	-0.34	1073	228.38
25	37.87	0.20	1082	-5.83
49	27.31	0.04	1083	181.60
36	33.26	0.35	1087	-20.36
38	44.15	-0.30	1088	172.22

Vuoi effettuare un altro ordinamento? [S/n]
n

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
4

Buona giornata

*** VISUALIZZA I DATI ***

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
1	45.31	-0.31	967	-359.10
2	47.45	-0.15	924	61.21
3	46.54	-0.33	964	232.44
4	25.87	-0.20	827	151.56
5	29.56	-0.35	1042	-294.19
6	25.61	-0.38	1004	351.74
7	39.49	0.03	882	-353.58
8	41.74	-0.33	895	73.27
9	40.39	-0.44	938	-35.43
10	32.90	0.02	999	203.99
11	48.18	0.46	911	270.70
12	31.27	-0.04	964	28.33
13	46.17	0.28	1068	260.81
14	35.11	0.11	857	357.69
15	26.08	0.34	941	-168.33
16	26.88	0.18	835	-89.38
17	32.19	-0.22	906	-353.67
18	43.30	0.34	948	63.29
19	36.98	0.24	929	-10.84
20	28.15	0.24	1001	-30.27
21	40.42	0.24	823	71.31
22	31.02	-0.35	940	52.13
23	49.75	0.02	1008	-53.89
24	41.60	-0.33	1023	181.12
25	37.87	0.20	1082	-5.83
26	47.68	-0.36	1015	-253.80
27	27.22	-0.07	830	138.88
28	46.99	-0.35	986	335.96
29	29.86	-0.31	972	231.61
30	37.85	-0.34	1073	228.38
31	39.43	-0.22	890	167.05
32	37.14	0.22	967	131.21
33	26.36	0.33	852	-271.43
34	28.06	-0.07	924	12.25
35	33.98	0.05	991	323.52
36	33.26	0.35	1087	-20.36
37	32.80	0.48	1009	-31.61
38	44.15	-0.30	1088	172.22
39	47.31	0.00	930	244.40
40	26.74	0.07	991	-340.22
41	40.86	0.34	859	22.56
42	28.52	0.34	983	113.48
43	48.56	-0.19	850	-280.85
44	43.46	-0.36	1020	-154.02
45	43.81	0.10	899	240.93
46	26.99	-0.50	1034	-178.04
47	28.33	-0.29	838	220.49
48	36.64	-0.39	967	38.31
49	27.31	0.04	1083	181.60
50	32.61	0.20	817	-45.56
51	38.55	0.08	916	-45.58

52	49.09	0.00	949	92.65
53	33.73	-0.31	803	140.95
54	49.95	-0.04	1009	-231.59
55	36.22	-0.41	885	-289.80
56	32.61	0.39	887	310.69
57	30.45	-0.09	800	-196.34
58	36.93	-0.05	1096	92.21
59	45.81	0.35	898	70.44
60	34.69	0.07	1022	89.84
61	47.67	0.06	979	-227.33
62	42.61	0.10	991	-185.14
63	27.47	-0.01	911	60.92
64	42.43	0.12	1055	173.34
65	46.76	0.08	886	219.26
66	26.56	0.23	842	296.32
67	39.45	-0.18	998	120.80
68	46.79	-0.39	1099	-139.81
69	33.51	-0.35	889	252.88
70	45.59	0.14	910	-302.87
71	32.82	-0.09	1088	32.47
72	33.41	-0.35	1023	-24.80
73	49.00	0.33	885	171.36
74	48.63	-0.20	857	268.81
75	38.35	0.28	854	-268.39

----- VALORI VELOCITA' -----

Mediano: 37.14

Massimo: 49.95
1(54,49.95,-0.04,1009,-231.59)

Medio: 37.53

Minimo: 25.61
1(6,25.61,-0.38,1004,351.74)

----- VALORI ACCELERAZIONE -----

Mediano: 0.00

Massimo: 0.48
1(37,32.80,0.48,1009,-31.61)

Medio: -0.03

Minimo: -0.50
1(46,26.99,-0.50,1034,-178.04)

----- VALORI GIRI -----

Mediano: 948

Massimo: 1099

1(68,46.79,-0.39,1099,-139.81)

Medio: 948

Minimo: 800
1(57,30.45,-0.09,800,-196.34)

----- VALORI ANGOLO DI STERZATA -----

Mediano: 61.21

Massimo: 357.69
1(14,35.11,0.11,857,357.69)

Medio: 30.84

Minimo: -359.10
1(1,45.31,-0.31,967,-359.10)

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
1

*** ORDINA I DATI ***

Inserire la colonna in base alla quale effettuare l'ordinamento: B, C, D, E
e
Tempo(A) Velocita(B) Accel.(C) Giri(D) Angolo di sterzata(E)
1 45.31 -0.31 967 -359.10
17 32.19 -0.22 906 -353.67
7 39.49 0.03 882 -353.58
40 26.74 0.07 991 -340.22
70 45.59 0.14 910 -302.87
5 29.56 -0.35 1042 -294.19
55 36.22 -0.41 885 -289.80
43 48.56 -0.19 850 -280.85
33 26.36 0.33 852 -271.43
75 38.35 0.28 854 -268.39
26 47.68 -0.36 1015 -253.80
54 49.95 -0.04 1009 -231.59
61 47.67 0.06 979 -227.33
57 30.45 -0.09 800 -196.34
62 42.61 0.10 991 -185.14
46 26.99 -0.50 1034 -178.04
15 26.08 0.34 941 -168.33
44 43.46 -0.36 1020 -154.02
68 46.79 -0.39 1099 -139.81

16	26.88	0.18	835	-89.38
23	49.75	0.02	1008	-53.89
51	38.55	0.08	916	-45.58
50	32.61	0.20	817	-45.56
9	40.39	-0.44	938	-35.43
37	32.80	0.48	1009	-31.61
20	28.15	0.24	1001	-30.27
72	33.41	-0.35	1023	-24.80
36	33.26	0.35	1087	-20.36
19	36.98	0.24	929	-10.84
25	37.87	0.20	1082	-5.83
34	28.06	-0.07	924	12.25
41	40.86	0.34	859	22.56
12	31.27	-0.04	964	28.33
71	32.82	-0.09	1088	32.47
48	36.64	-0.39	967	38.31
22	31.02	-0.35	940	52.13
63	27.47	-0.01	911	60.92
2	47.45	-0.15	924	61.21
18	43.30	0.34	948	63.29
59	45.81	0.35	898	70.44
21	40.42	0.24	823	71.31
8	41.74	-0.33	895	73.27
60	34.69	0.07	1022	89.84
58	36.93	-0.05	1096	92.21
52	49.09	0.00	949	92.65
42	28.52	0.34	983	113.48
67	39.45	-0.18	998	120.80
32	37.14	0.22	967	131.21
27	27.22	-0.07	830	138.88
53	33.73	-0.31	803	140.95
4	25.87	-0.20	827	151.56
31	39.43	-0.22	890	167.05
73	49.00	0.33	885	171.36
38	44.15	-0.30	1088	172.22
64	42.43	0.12	1055	173.34
24	41.60	-0.33	1023	181.12
49	27.31	0.04	1083	181.60
10	32.90	0.02	999	203.99
65	46.76	0.08	886	219.26
47	28.33	-0.29	838	220.49
30	37.85	-0.34	1073	228.38
29	29.86	-0.31	972	231.61
3	46.54	-0.33	964	232.44
45	43.81	0.10	899	240.93
39	47.31	0.00	930	244.40
69	33.51	-0.35	889	252.88
13	46.17	0.28	1068	260.81
74	48.63	-0.20	857	268.81
11	48.18	0.46	911	270.70
66	26.56	0.23	842	296.32
56	32.61	0.39	887	310.69
35	33.98	0.05	991	323.52
28	46.99	-0.35	986	335.96

6 25.61 -0.38 1004 351.74
14 35.11 0.11 857 357.69

Vuoi effettuare un altro ordinamento? [S/n]
n

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
3

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
1	45.31	-0.31	967	-359.10
17	32.19	-0.22	906	-353.67
7	39.49	0.03	882	-353.58
40	26.74	0.07	991	-340.22
70	45.59	0.14	910	-302.87
5	29.56	-0.35	1042	-294.19
55	36.22	-0.41	885	-289.80
43	48.56	-0.19	850	-280.85
33	26.36	0.33	852	-271.43
75	38.35	0.28	854	-268.39
26	47.68	-0.36	1015	-253.80
54	49.95	-0.04	1009	-231.59
61	47.67	0.06	979	-227.33
57	30.45	-0.09	800	-196.34
62	42.61	0.10	991	-185.14
46	26.99	-0.50	1034	-178.04
15	26.08	0.34	941	-168.33
44	43.46	-0.36	1020	-154.02
68	46.79	-0.39	1099	-139.81
16	26.88	0.18	835	-89.38
23	49.75	0.02	1008	-53.89
51	38.55	0.08	916	-45.58
50	32.61	0.20	817	-45.56
9	40.39	-0.44	938	-35.43
37	32.80	0.48	1009	-31.61
20	28.15	0.24	1001	-30.27
72	33.41	-0.35	1023	-24.80
36	33.26	0.35	1087	-20.36
19	36.98	0.24	929	-10.84
25	37.87	0.20	1082	-5.83
34	28.06	-0.07	924	12.25
41	40.86	0.34	859	22.56
12	31.27	-0.04	964	28.33
71	32.82	-0.09	1088	32.47
48	36.64	-0.39	967	38.31
22	31.02	-0.35	940	52.13
63	27.47	-0.01	911	60.92
2	47.45	-0.15	924	61.21

18	43.30	0.34	948	63.29
59	45.81	0.35	898	70.44
21	40.42	0.24	823	71.31
8	41.74	-0.33	895	73.27
60	34.69	0.07	1022	89.84
58	36.93	-0.05	1096	92.21
52	49.09	0.00	949	92.65
42	28.52	0.34	983	113.48
67	39.45	-0.18	998	120.80
32	37.14	0.22	967	131.21
27	27.22	-0.07	830	138.88
53	33.73	-0.31	803	140.95
4	25.87	-0.20	827	151.56
31	39.43	-0.22	890	167.05
73	49.00	0.33	885	171.36
38	44.15	-0.30	1088	172.22
64	42.43	0.12	1055	173.34
24	41.60	-0.33	1023	181.12
49	27.31	0.04	1083	181.60
10	32.90	0.02	999	203.99
65	46.76	0.08	886	219.26
47	28.33	-0.29	838	220.49
30	37.85	-0.34	1073	228.38
29	29.86	-0.31	972	231.61
3	46.54	-0.33	964	232.44
45	43.81	0.10	899	240.93
39	47.31	0.00	930	244.40
69	33.51	-0.35	889	252.88
13	46.17	0.28	1068	260.81
74	48.63	-0.20	857	268.81
11	48.18	0.46	911	270.70
66	26.56	0.23	842	296.32
56	32.61	0.39	887	310.69
35	33.98	0.05	991	323.52
28	46.99	-0.35	986	335.96
6	25.61	-0.38	1004	351.74
14	35.11	0.11	857	357.69

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
4

Buona giornata

*** VISUALIZZA I DATI ***

Tempo(A)	Velocita(B)	Accel.(C)	Giri(D)	Angolo di sterzata(E)
1	45.31	-0.31	967	-359.10
2	47.45	-0.15	924	61.21
3	46.54	-0.33	964	232.44
4	25.87	-0.20	827	151.56
5	29.56	-0.35	1042	-294.19
6	25.61	-0.38	1004	351.74
7	39.49	0.03	882	-353.58
8	41.74	-0.33	895	73.27
9	40.39	-0.44	938	-35.43
10	32.90	0.02	999	203.99
11	48.18	0.46	911	270.70
12	31.27	-0.04	964	28.33
13	46.17	0.28	1068	260.81
14	35.11	0.11	857	357.69
15	26.08	0.34	941	-168.33
16	26.88	0.18	835	-89.38
17	32.19	-0.22	906	-353.67
18	43.30	0.34	948	63.29
19	36.98	0.24	929	-10.84
20	28.15	0.24	1001	-30.27
21	40.42	0.24	823	71.31
22	31.02	-0.35	940	52.13
23	49.75	0.02	1008	-53.89
24	41.60	-0.33	1023	181.12
25	37.87	0.20	1082	-5.83
26	47.68	-0.36	1015	-253.80
27	27.22	-0.07	830	138.88
28	46.99	-0.35	986	335.96
29	29.86	-0.31	972	231.61
30	37.85	-0.34	1073	228.38
31	39.43	-0.22	890	167.05
32	37.14	0.22	967	131.21
33	26.36	0.33	852	-271.43
34	28.06	-0.07	924	12.25
35	33.98	0.05	991	323.52
36	33.26	0.35	1087	-20.36
37	32.80	0.48	1009	-31.61
38	44.15	-0.30	1088	172.22
39	47.31	0.00	930	244.40
40	26.74	0.07	991	-340.22
41	40.86	0.34	859	22.56
42	28.52	0.34	983	113.48
43	48.56	-0.19	850	-280.85
44	43.46	-0.36	1020	-154.02
45	43.81	0.10	899	240.93
46	26.99	-0.50	1034	-178.04
47	28.33	-0.29	838	220.49
48	36.64	-0.39	967	38.31
49	27.31	0.04	1083	181.60
50	32.61	0.20	817	-45.56
51	38.55	0.08	916	-45.58

52	49.09	0.00	949	92.65
53	33.73	-0.31	803	140.95
54	49.95	-0.04	1009	-231.59
55	36.22	-0.41	885	-289.80
56	32.61	0.39	887	310.69
57	30.45	-0.09	800	-196.34
58	36.93	-0.05	1096	92.21
59	45.81	0.35	898	70.44
60	34.69	0.07	1022	89.84
61	47.67	0.06	979	-227.33
62	42.61	0.10	991	-185.14
63	27.47	-0.01	911	60.92
64	42.43	0.12	1055	173.34
65	46.76	0.08	886	219.26
66	26.56	0.23	842	296.32
67	39.45	-0.18	998	120.80
68	46.79	-0.39	1099	-139.81
69	33.51	-0.35	889	252.88
70	45.59	0.14	910	-302.87
71	32.82	-0.09	1088	32.47
72	33.41	-0.35	1023	-24.80
73	49.00	0.33	885	171.36
74	48.63	-0.20	857	268.81
75	38.35	0.28	854	-268.39
76	46.95	-0.43	929	78.94
77	37.87	-0.40	855	110.78
78	39.00	0.42	1012	-196.50
79	34.79	-0.01	953	117.22
80	34.87	0.01	996	-2.30
81	35.18	0.11	837	135.48
82	40.77	-0.40	895	-355.76
83	37.67	0.25	1005	261.54
84	34.25	0.05	1034	-86.32
85	34.67	-0.32	1071	328.12
86	40.49	0.45	855	-265.23
87	41.16	-0.44	808	-339.96
88	40.12	0.36	956	-265.23
89	28.60	-0.49	882	159.58
90	27.12	-0.28	922	149.69
91	33.33	-0.16	850	-238.38
92	45.27	0.30	859	-95.35
93	47.56	0.30	1024	19.11
94	31.72	-0.10	1049	-255.74
95	35.98	-0.43	922	-261.69
96	31.03	0.05	805	52.81
97	28.80	0.31	825	-135.83
98	44.85	0.40	824	-155.34
99	46.49	-0.14	859	175.53
100	31.61	-0.27	897	-195.49

----- VALORI VELOCITA' -----

Mediano: 37.06

Massimo: 49.95
1(54,49.95,-0.04,1009,-231.59)

Medio: 37.45

Minimo: 25.61
1(6,25.61,-0.38,1004,351.74)

----- VALORI ACCELERAZIONE -----

Mediano: -0.00

Massimo: 0.48
1(37,32.80,0.48,1009,-31.61)

Medio: -0.03

Minimo: -0.50
1(46,26.99,-0.50,1034,-178.04)

----- VALORI GIRI -----

Mediano: 934

Massimo: 1099
1(68,46.79,-0.39,1099,-139.81)

Medio: 940

Minimo: 800
1(57,30.45,-0.09,800,-196.34)

----- VALORI ANGOLO DI STERZATA -----

Mediano: 45.22

Massimo: 357.69
1(14,35.11,0.11,857,357.69)

Medio: 10.53

Minimo: -359.10
1(1,45.31,-0.31,967,-359.10)

MENU OPERAZIONI

Inserire il numero dell'operazione desiderata:
1 - Ordinare la lista
2 - Cercare un elemento
3 - Visualizzare la lista
4 - Terminare l'esecuzione
2

```
*** CERCA UN VALORE ***

Inserire la colonna nella quale cercare il valore: B, C, D, E
d

Inserire il valore da cercare:
859
1(92,45.27,0.30,859,-95.35)
2(41,40.86,0.34,859,22.56)
3(99,46.49,-0.14,859,175.53)

Si desidera effettuare un'altra ricerca? [S/n]
n


      ***MENU OPERAZIONI***

Inserire il numero dell'operazione desiderata:
      1 - Ordinare la lista
      2 - Cercare un elemento
      3 - Visualizzare la lista
      4 - Terminare l'esecuzione
4

Buona giornata
```

Valutazione della complessità del programma – Analisi teorica

Massimo	
Passi	**Migliore/ Pessimo**
`massimo = selezionaValori(coda_p,colonna)`	$1 + d = \boldsymbol{O(1)}$

Minimo	
Passi	**Migliore/ Pessimo**
`minimo = selezionaValori(testa_p,colonna)`	$1 + d = \boldsymbol{O(1)}$

Media	
Passi	**Migliore/ Pessimo**
`media = 0.0;`	1
`for (elem_p = testa_p;`	1
`elem_p != NULL;`	n
`elem_p = elem_p->succ_p)`	n
`media += selezionaValori(elem_p,colonna);`	$(1 + d)n$
`media = media / ((float)lunghezza_lista);`	1
Caso unico:	$T(n) = 1 + 1 + n \cdot (1 + 1 + 1 + d) + 1 + 1 = n(3 + d) + 4 = \boldsymbol{O(n)}$

Mediana		
Passi	**Migliore**	**Pessimo**
`contatore = termina = 0;`	2	
`valore1 = valore2 = mediana = 0.0;`	3	
`for (elem_p = testa_p;`	1	
`elem_p != NULL && termina == 0.0;`	$(n+1)/2$	$(n/2)$
`elem_p = elem_p->succ_p) {`	$(n+1)/2$	$(n/2)$
`contatore++;`	$(n+1)/2$	$(n/2)$
`if ((lunghezza_lista % 2) != 0) {`	$(n+1)/2$	$(n/2)$
`if (contatore == (lunghezza_lista+1)/2) {`	$(n+1)/2$	0
`mediana = selezionaValori(elem_p,colonna);`	$1 + d$	0
`termina = 1 }`	1	0
`else { if (contatore == lunghezza_lista / 2) {`	0	$(n/2)$
`valore1 = selezionaValori(elem_p,colonna);`	0	$1 + d$
`valore2 = selezionaValori(elem_p->succ_p,colonna);`	0	$1 + d$
`mediana = (valore1 + valore2) / 2;`	0	1
`termina = 1 } }`	0	1
Dispari:	$T(n) = 5 + 1 + \frac{n+1}{2} \cdot (1 + 1 + 1 + 1 + 1) + 1 + d + 1 = \frac{5n + 2d + 21}{2} = \boldsymbol{O\left(\frac{n}{2}\right)}$	
Pari:	$T(n) = 5 + 1 + \frac{n}{2} \cdot (1 + 1 + 1 + 1 + 1) + 1 + 2 \cdot (1 + d) + 1 + 1 = \frac{5n + 4d + 22}{2} = \boldsymbol{O\left(\frac{n}{2}\right)}$	

Ricerca		
Passi	**Migliore**	**Pessimo**

`for (elem_p = testa_p;`
`elem_p != NULL &&`
`selezionaValori(elem_p,colonna) != cercato`
`elem_p = elem_p->succ_p)`
	1
	$1+d$
	0
		1
		$(1+d)n$
		n

Migliore (elemento in posto 1):	$T(n) = 1+1+d = 2+d = \boldsymbol{O(1)}$	
Pessimo (elemento assente):	$T(n) = 1+n\cdot(1+d+1)+1 = n(2+d)+2 = \boldsymbol{O(n)}$	

Ordinamento		
Passi	**Migliore**	**Pessimo**
`*lista2 = NULL;`	1	
`for (elem_p = *testa_p;`	1	
`elem_p != NULL;`	n	
`elem_p = elem_p->succ_p)`	n	
`inserisciElemento(&lista2,colonna,elem_p);`	1	n
`free(elem_p) }`	1	
Migliore (lista ordinata al contrario):	$T(n) = 1+1+n\cdot(1+1+d+1)+1 = n(3+d)+3 = \boldsymbol{O(n)}$	
Pessimo (lista già ordinata):	$T(n) = 1+1+n\cdot(1+1+n+1)+1 = n^2+3n+3 = \boldsymbol{O(n^2)}$	

Selezione Valori		
Passi	**Migliore**	**Pessimo**
`valore = 0.0;`	1	1
`switch(colonna){`	-	
`case 1:`	1	1
`valore = elem_p->tempo;` `break;`	1 1	0
`case 2:`	0	1
`valore = elem_p->velocita;` `break;`	0	0
`case 3:`	0	1
`valore = elem_p->accelerazione;` `break;`	0	0
`case 4:`	0	1
`valore = elem_p->giri;` `break;`	0	0
`case 5:`	0	1
`valore = elem_p->sterzata;` `break; }`	0	1 1
Migliore (scelta 1):	$T(n) = 1+1+1+1 = 4 = \boldsymbol{O(1)}$	
Pessimo (scelta 5):	$T(n) = 1+1+1+1+1+1+1+1 = 8 = \boldsymbol{O(1)}$	

Inserimento Elemento		
Operazioni	**Migliore**	**Pessimo**
`for (corr_p = prec_p = *testa_p;`	1	
`(corr_p != NULL &&`	$1+d+d$	$(1+2d)n$
`(selezionaValori(corr_p,colonna) <`		
`selezionaValori(input_p,colonna);`		
`prec_p = corr_p,corr_p = corr_p->succ_p);`	0	1
`if (corr_p != NULL && (corr_p->tempo == input_p->tempo)){`	1	
`inserito = 0; }`	1	0
`else {`		
`inserito = 1;`	0	n
`nuovo_p = (elem_lista_t *)malloc(sizeof(elem_lista_t));`	0	n
`nuovo_p->tempo = input_p->tempo;`	0	n
`nuovo_p->velocita = input_p->velocita;`	0	n
`nuovo_p->accelerazione = input_p->accelerazione;`	0	n
`nuovo_p->giri = input_p->giri;`	0	n
`nuovo_p->sterzata = input_p->sterzata;`	0	n
`nuovo_p->succ_p = corr_p; }`	0	n
`if (corr_p == *testa_p) {`	0	n
`prec_p->succ_p = nuovo_p;`	0	n
`nuovo_p->prec_p = prec_p; }`	0	n
`else {`		
`prec_p->succ_p = nuovo_p;`	0	n
`nuovo_p->prec_p = prec_p; }`	0	n
Migliore (elemento già in posto 1):	$T(n) = 1+1+d+d+1+1 = 4+2d = \boldsymbol{O(1)}$	
Pessimo (elemento inserito in posto n):	$T(n) = 1+n\cdot(1+2d+1)+1+1+1+1+1+1+1+1+1+1+1+1+1 = n(2+2d)+14 = \boldsymbol{O(n)}$	

Valutazione della complessità del programma – Analisi sperimentale

Per valutare la complessità di ogni algoritmo è stata creata una copia del codice sorgente, rinominato *telemetriaMod.c*

Tra le righe di codice dei diversi algoritmi è stata inserita una variabile (*n*) di tipo intero che conta il numero di passi base.

Nel caso della funzione inserisciElemento(), l'output è stato modificato rendendolo pari al flag {0,1} sommato al valore della variabile intera che conta i passi base.

Nel caso della funzione selezionaValori(), il numero di passi base è pari al valore della variabile "*colonna*" sommato al costo delle istruzioni che vengono eseguite in qualsiasi caso (cioè 3).

La complessità dell'algoritmo in esame è indicata con $T(n)$

La complessità dell'algoritmo selezionaValori() è indicata con $V(n) = c$, con c costante

La complessità dell'algoritmo inserisciElemento() è indicata con $I(n) = n(2 + 2V(n)) + 14$

Il numero totale di passi base, riportato di seguito in grassetto, è frutto del conteggio della suddetta variabile. Le rispettive formule sono state scritte per chiarezza e per una controverifica dei dati sperimentali.

Costo algoritmi costanti

I seguenti algoritmi hanno complessità costante a prescindere dal numero di elementi di cui è composta la lista. Il costo dipende solo dalla colonna.

Seleziona valori in base alla colonna:

Colonna	Costo
Velocità	selezionaValori() → $V(n) = \mathbf{5}$
Accelerazione	selezionaValori() → $V(n) = \mathbf{6}$
Giri del motore	selezionaValori() → $V(n) = \mathbf{7}$
Angolo di sterzata	selezionaValori() → $V(n) = \mathbf{8}$

Massimo e Minimo:

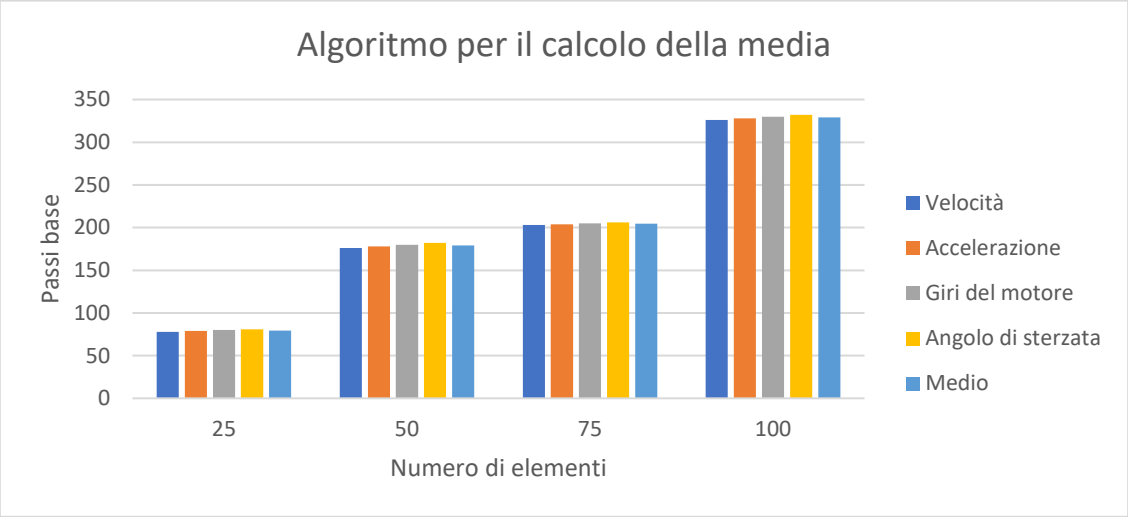
Colonna	Costo
Velocità	massimo, minimo → $T(n) = 1 + V(n) = \mathbf{6}$
Accelerazione	massimo, minimo → $T(n) = 1 + V(n) = \mathbf{7}$
Giri del motore	massimo, minimo → $T(n) = 1 + V(n) = \mathbf{8}$
Angolo di sterzata	massimo, minimo → $T(n) = 1 + V(n) = \mathbf{9}$

Costo algoritmi

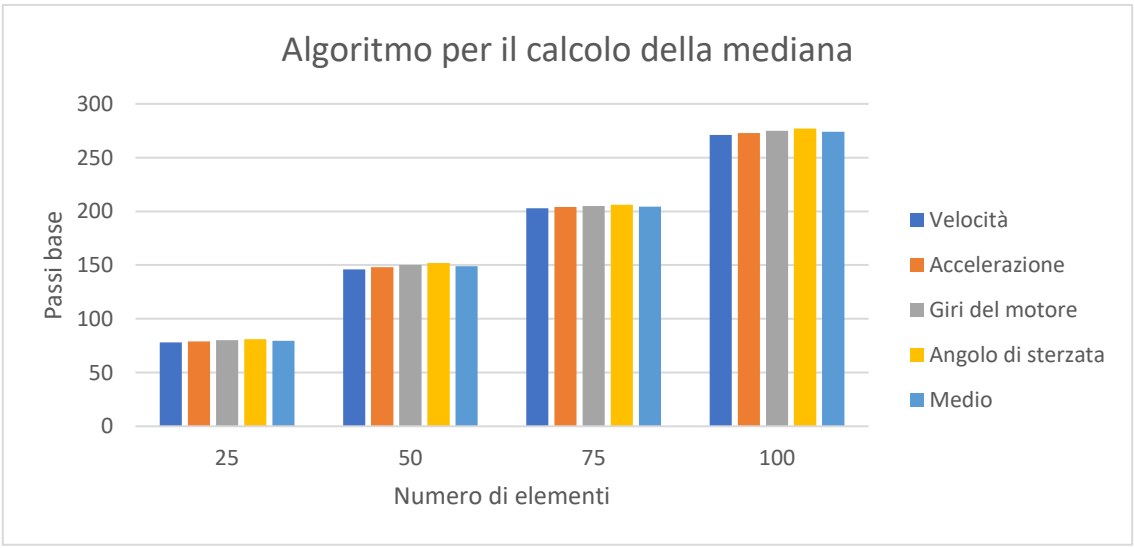
Ricerca:

25 elementi	**50 Elementi**
Ordinamento: colonna 2 (Velocità) **Test 1** Colonna: 3 (Accelerazione) Elemento cercato: 0,14 Costo: **42** **Test 2** Colonna: 4 (Giri del motore) Elemento cercato: 1035 Costo: **110** **Test 3** Colonna: 5 (Angolo di sterzata) Elemento cercato: 81,10 Costo: **62**	Ordinamento: colonna 4 (Giri del motore) **Test 1** Colonna: 2 (Velocità) Elemento cercato: 41,60 Costo: **289** **Test 2** Colonna: 3 (Accelerazione) Elemento cercato: 0,34 Costo: **74** **Test 3** Colonna: 4 (Giri del motore) Elemento cercato: 838 Costo: **47**
75 Elementi	**100 Elementi**
Ordinamento: colonna 5 (Angolo di sterzata) **Test 1** Colonna: 3 (Accelerazione) Elemento cercato: 0,33 Costo: **66** **Test 2** Colonna: 4 (Giri del motore) Elemento cercato: 1068 Costo: **596** **Test 3** Colonna: 5 (Angolo di sterzata) Elemento cercato: -359,10 Costo: $2 + V(n) =$ **10** (Caso ottimo)	Ordinamento: colonna 3 (Accelerazione) **Test 1** Colonna: 2 (Velocità) Elemento cercato: 32.82 Costo: **289** **Test 2** Colonna: 3 (Accelerazione) Elemento cercato: 0,48 Costo: **802** **Test 3** Colonna: 4 (Giri del motore) Elemento cercato: 1001 Costo: **704**

Costo algoritmo per il calcolo della media					
Nr. Elementi	**Velocità**	**Accelerazione**	**Giri del motore**	**Angolo di sterzata**	**Medio**
25	204	229	254	279	241.5
50	404	454	504	554	479
75	604	679	754	829	716.5
100	804	904	1004	1104	954



Costo algoritmo per il calcolo della mediana					
Nr. Elementi	**Velocità**	**Accelerazione**	**Giri del motore**	**Angolo di sterzata**	**Medio**
25	78	79	80	81	79.5
50	146	148	150	152	149
75	203	204	205	206	204.5
100	271	273	275	277	274



Costo algoritmo di ordinamento					
Nr. Elementi	**Velocità**	**Accelerazione**	**Giri del motore**	**Angolo di sterzata**	**Medio**
25	2332	2550	3046	3610	2884.5
50	7711	8871	9529	12091	9550.5
75	19260	20146	20724	25694	21456
100	32345	36895	38765	47899	38976