Studenti: Attarantato Kevin

Matricola: 282785

Roselli Giorgia

Matricola: 282724

CORSO DI ALGORITMI E STRUTTURE DATI

Progetto per la sessione invernale 2019/2020

Programma per la gestione di CPU con memorizzazione dei dati in una struttura dati di tipo array ed una di tipo albero con relativa ricerca per CPU per entrambe le strutture.

Docente: Ing. Valerio Freschi

SPECIFICA DEL PROBLEMA

Specifica del problema:

Si supponga di dover progettare un programma per la gestione dei dati delle CPU di un datacenter. Il sistema di monitoraggio permette di rilevare, una volta all'ora per ogni CPU: la potenza richiesta (in [W]), la temperatura del chip (in [°C]), il numero di processi attivi, la quantità di memoria RAM occupata (in ` [MB]). L'intero log viene poi scritto (una volta al giorno) su un file in formato testo, secondo il seguente formato (si assumano campi separati da tabulazione o spazio):

- Tempo: un numero intero ∈ [1, 24] che rappresenta l'ora del giorno a cui si riferisce il rilevamento.
- CPU: un codice alfanumerico che risulta dalla concatenazione di tre lettere e tre numeri che identificano la singola CPU.
- Potenza: un numero reale che rappresenta il consumo di potenza (medio) nell'ora di rilevamento.
- Temperatura: un numero reale che rappresenta la temperatura (media) nell'ora di rilevamento.
- Processi: un numero reale che rappresenta il numero (medio) di processi attivi nell'ora di rilevamento.
- Memoria: un numero reale che rappresenta la quantità (media) di memoria RAM richiesta nell'ora di rilevamento.

Ad esempio:

Tempo	CPU	Potenza	Temperatura	Processi	Memoria
3	ABC020	8.36	32.70	23.7	10455.7
3	ABC176	9.24	22.43	3.70	12465.8
3	AAF184	10.11	32.14	11.20	20425.9
...	...	...	...	...	...
12	ABC020	4.60	30.43	20.1	12855.7
12	ABC176	12.24	24.55	3.70	265.8
12	AAF184	10.34	22.83	15.2	17523.8
...	...	...	...	...	...
21	ABC176	13.32	42.21	30.4	22585.5
...	...	...	...	...	...

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 albero.
2. Ricerca e restituisce i dati relativi alle rilevazioni di una data CPU nel giorno. Ad esempio: se l'utente chiede i dati relativi alla CPU ABC176, il programma deve restituire le informazioni contenute nelle righe corrispondenti:

Tempo	CPU	Potenza	Temperatura	Processi	Memoria
3	ABC176	9.24	22.43	3.70	12465.8
12	ABC176	12.24	24.55	3.70	265.8
21	ABC176	13.32	42.21	30.4	22585.5
...	...	...	...	...	...

Il programma deve inoltre prevedere una modalità che implementa le stesse funzionalità utilizzando un array al posto dell'albero. Per quanto riguarda l'analisi teorica si deve fornire la complessità dell'algoritmo di ricerca per la versione basata su albero e per quella basata su array. Oltre all'analisi teorica della complessità si deve effettuare uno studio sperimentale della stessa

per le operazioni di ricerca. Come suggerimento si può operare generando un numero N di rilevazioni casuali (dove N rappresenta il numero di CPU monitorate). L'analisi sperimentale deve quindi valutare la complessità al variare di N e confrontare l'algoritmo di ricerca che lavora su albero con il corrispondente che lavora su array.

ANALISI DEL PROBLEMA

Analisi del problema:

Il problema richiede di acquisire un file di testo contenente dati relativi a delle CPU e di inserirle in due strutture dati:

- Una struttura di tipo array;
- Una struttura di tipo albero.

Oltre all'inserimento, il programma, deve anche ricercare dato il nome di una specifica CPU tutti i dati riferiti a quest'ultima. Questo sia per la struttura dati di tipo array e di tipo albero.

Infine si richiedono due analisi:

- Per quanto riguarda l'analisi teorica si deve fornire la complessità dell'algoritmo di ricerca sia per la versione basata su albero sia per la versione basata su array;
- Per quanto riguarda l'analisi sperimentale si deve effettuare uno studio sperimentale della stessa per le operazioni di ricerca.

Di seguito sono riportati gli input, output e le relazioni intercorrenti.

Analisi degli input:

- Acquisizione dei dati dal file di testo ed inserimento degli stessi nelle strutture dati di tipo albero ed array.
- Acquisizione del numero di CPU dall'utente, per la corretta generazione dinamica del file di testo.
- Acquisizione del nome della CPU da ricercare all'interno del file.

Analisi output:

- Stampa a video del file precedentemente acquisito ed inserito nella struttura desiderata dall'utente.
- Stampa a video di tutti i dati inerenti alla CPU selezionata, tramite il nome, dall'utente.

Relazioni intercorrenti fra input e output:

Per la stampa a video del file, l'utente deve digitare il numero di CPU che vuole inserire, per permettere al programma di generare il file di testo, inoltre l'acquisizione del nome della CPU, unito al numero totale delle CPU, permette di ritrovare quella scelta e di stamparne i relativi dati.

PROGETTAZIONE DELL'ALGORITMO

Operazioni svolte dal programma

Il programma deve essere in grado di svolgere le seguenti funzioni:

- Acquisire i dati relativi alle CPU ed inserirli nella struttura dati array e nella struttura dati albero;
- In base al nome della CPU desiderata stamparla a schermo con i relativi dati.

Strutture dati utilizzate

Le strutture dati utilizzate in questo programma, sono state come da specifica, la struttura dati di tipo array e la struttura dati di tipo albero binario di ricerca.

Per l'implementazione della struttura dati di tipo array si è pensato di far selezionare all'utente il numero di CPU totali cosi da riuscire ad allocare la giusta memoria e non dargli una grandezza fissa. Inoltre l'array si è volutamente fatto non ordinato così da avere una complessità, nel caso della ricerca del nome della CPU, pari a

$$O(n)$$

sia nel caso pessimo che nel caso ottimo, questo dovuto al fatto che l'algoritmo non trova solo una CPU, cercata dall'utente, all'interno dell'array ma al più 'n' CPU. Questo perché si è pensato, soprattutto ai fini dell'analisi, di voler mettere in risalto ancor di più le differenze dei due algoritmi richiesti, dato che la struttura dati di tipo albero, cosi come si è pensato di implementarla, ordina i nodi in base al nome della CPU, e quindi avrà una complessità migliore di quella per gli array.
Per quanto riguarda la complessità quando parliamo della stampa dei valori presenti nell'array sarà ancora

$$O(n)$$

in quanto è necessario passare tutti i valori per poterli stampare.

Per l'implementazione della struttura dati di tipo albero, invece, si è optato per alberi binari di ricerca in modo tale da guadagnare in termini di complessità del tempo di esecuzione. Questo perché ad ogni chiamata ricorsiva scendiamo sempre di un livello fino a trovare l'elemento desiderato oppure fino a terminare l'albero.
Come detto in precedenza, si è voluto ordinare i nodi in base all'ordine alfabetico della stringa del nome della CPU. Questo per poter riuscire ad avere una complessità che sia uguale all'altezza stessa dell'albero ovvero O(h) dove h è l'altezza dell'albero. Detto questo e sapendo che $h = \log_2 n$, otterremo una complessità pari a

$$O(\log_2(n))$$

dove ‘n’ sono il numero di nodi dell’albero, in quanto ad ogni chiamata ricorsiva scendiamo di un livello alla ricerca dell’elemento da cercare. Se così non fosse stato avremo riottenuto la complessità dell’algoritmo usante la struttura dati di tipo array.
Anche qui, per quanto riguarda la complessità quando parliamo della stampa dei valori sarà

$$O(n)$$

poiché tutti i nodi vengono visitati una sola volta.

Passi principali dell’algoritmo

Acquisizione del numero totale delle CPU

La funzione per acquisire il numero totale di CPU stampa un messaggio dove viene espressamente chiesto di inserire un numero di CPU arbitrario strettamente maggiore di zero in quanto, nella progettazione di questa funzione e del programma intero, si è scelto di non poter costruire un file di testo vuoto ma deve contenere almeno una CPU.
Assunto il numero di CPU si va ad usare questo dato per assegnare memoria per la creazione della struttura array.

Creazione casuale dei dati

Questa funzione sfrutta il dato precedentemente acquisito per andare a generare i sei dati in maniera casuale in un file di testo chiamato “lista_cpu.txt”.
Il primo dato quello relativo al tempo sarà un intero compreso tra 1 e 24 estremi compresi, il secondo dato relativo al nome della CPU viene costruito come segue:

1. Viene generata una stringa, di lunghezza tre, casuale di sole lettere maiuscole;
2. Viene generata una stringa, di lunghezza tre, casuale di soli numeri;
3. Viene usata una funzione apposita per le stringhe per concatenare il risultato della prima stringa con la seconda, così da andare a creare il nome alfanumerico relativo alla CPU.

Dal terzo dato al quinto dato, rispettivamente relativi alla potenza, temperatura e processi sono tutti generati casualmente da numeri reali aventi massimo tre valori della parte intera e massimo due valori della parte decimale.

Per quanto riguarda il sesto ed ultimo dato quest’ultimo viene creato senza tagliare la parte intera quindi avendo massimo otto valori per la parte intera e due per la parte decimale.

Messaggio per stampare a schermo il menù di scelta

In questo punto tramite la funzione “messaggio” viene mostrato all’utente a schermo il menu di scelta il quale potrà scegliere se stampare a schermo i dati tramite l’array e

ricercare poi la CPU desiderata tramite quest'ultimo oppure eseguire gli stessi passi ma tramite l'utilizzo della struttura di tipo albero.

Nel caso di scelta della struttura di tipo array

Acquisizione dati dal file

Attraverso il dato relativo al numero totali della CPU presenti nel file di testo, questa funzione va a caricare nell'array i dati così come sono stati scritti nel file di testo.

Stampa dati

Vengono stampati i dati contenuti nel file di testo salvati nell'array.

Ricerca dell'elemento

Questa funzione scorre tutti gli elementi nell'array stampando solo quelli che rispettano determinate caratteristiche ovvero essere uguali alla stringa inserita dall'utente tramite un'apposita funzione per le stringhe.
Generando le stringhe riferite al nome della CPU in modo casuale, è molto improbabile che si generino due uguale o ancor meno probabile tre uguali. Nonostante questo però sappiamo che se dovesse succedere verrebbero comunque stampate tutte, quindi anche eventuali doppioni.

Nel caso di scelta della struttura di tipo albero

Acquisizione dati dal file

Finché il file ha dati da leggere va ad aggiungere nodi all'albero, caricando i dati presenti su quella specifica riga nel nodo.

Aggiungi nodo

Questa funzione va a creare la radice nel caso non fosse stato ancora creato nulla e successivamente e ricorsivamente va a controllare la nuova stringa da inserire. Nel caso la stringa contenesse lettere più "piccole" alfabeticamente andrebbe ad inserire la parola nel nodo di sinistra altrimenti nel nodo di destra.

Stampa dati

Vengono stampati i dati contenuti nel file di testo salvati nell'albero nel passo precedente.

Ricerca dell'elemento

Quest'ultima funzione per la ricerca del valore all'interno dell'albero va a controllare il contenuto dei singoli nodi. Se la stringa desiderata non fosse nel nodo che si sta valutando a seconda della sua grandezza (intesa come ordine alfabetico) si scenderebbe nel nodo destro o sinistro senza dover mai passare necessariamente tutti i nodi per trovare la CPU richiesta.

Infine, possiamo fare lo stesso discorso che abbiamo fatto per la ricerca dell'elemento con la struttura array ovvero sapendo la bassa possibilità che si generino due stringhe riferite al nome della CPU uguali sappiamo che anche se succedesse queste verrebbero gestite dall'algoritmo, in quanto le nuove stringhe più piccole (alfabeticamente) verrebbero inserite nei nodi di sinistra mentre quelle più grandi nei nodi di destra.

IMPLEMENTAZIONE DELL'ALGORITMO

Implementazione dell'algoritmo:

Di seguito è riportato il Makefile il cui codice prevede la compilazione del file sorgente secondo lo standard ANSI C e segnalazione di warning:

```
gestione_cpu: gestione_cpu.c Makefile
     gcc -ansi -Wall -O gestione_cpu.c -o gestione_cpu
pulisci:
     rm -f gestione_cpu.o
```

Per rappresentare i dati del file di testo in codice C si è scelto di usare una struttura denominata data_t dove all'interno sono presenti dei campi corrispondenti alle informazioni dei dati relativi ad una CPU. La struttura è definita come segue:

```
typedef struct dati
{
    int    tempo;
    char   nome_cpu[6];
    double potenza;
    double temperatura;
    double processi;
    double memoria;
} data_t;
```

Per rappresentare l'albero in codice C invece si è usato una struttura denominata nodo_t dove all'interno sono presenti i campi di una struttura di tipo albero. La struttura è definita come segue:

```
typedef struct nodo
{
    data_t valore;
    struct nodo *figlio_sx;
    struct nodo *figlio_dx;
} nodo_t;
```

Nella main l'utente, dopo aver indicato quante CPU vuole inserire, ha la possibilità di scegliere dal menu 3 diverse opzioni:

1. Stampare i dati presenti nel file di testo e ricercare una CPU salvata nella struttura di tipo array;

2. Stampare i dati presenti nel file di testo e ricercare una CPU salvata nella struttura di tipo albero;
3. Chiudere il programma;

Nel caso l'utente dovesse scegliere o di usare la struttura array o di usare la struttura albero dovrà inserire il nome della CPU da cercare.

Nel caso l'utente scegliesse l'opzione per chiudere il programma, quest'ultimo verrebbe subito chiuso senza che si abbia la necessità di premere altro.

Si è deciso inoltre di organizzare il programma in funzioni permettendo un notevole risparmio e riuso del codice, con vantaggi piuttosto evidenti oltre che per permettere al codice di essere organizzato meglio, più leggibile e ordinato.

Per quanto riguarda l'acquisizione del file di testo, questo avviene dopo essere stato generato casualmente e dopo aver scelto quale struttura dati usare. Nonostante questo, renderà difficile il fatto che compaiano due CPU uguali, si è scelto comunque di gestire il caso all'interno dei relativi algoritmi di ricerca per motivi di robustezza del codice e completezza personale.

Per quanto riguarda, invece, l'algoritmo di ricerca per alberi si è deciso di usare alberi binari di ricerca così da ottenere una complessità di O(log2 n), dove n sono il numero di nodi dell'albero, per evidenziare, ancor di più, la differenza dall'algoritmo utilizzante la struttura di tipo array.

TESTING DEL PROGRAMMA

Testing del programma:

Qui di seguito viene riportata la schermata da terminale del file di testo generato con 5 dati puramente casuali e visualizzati scegliendo prima la struttura dati array e poi la struttura dati ad albero:

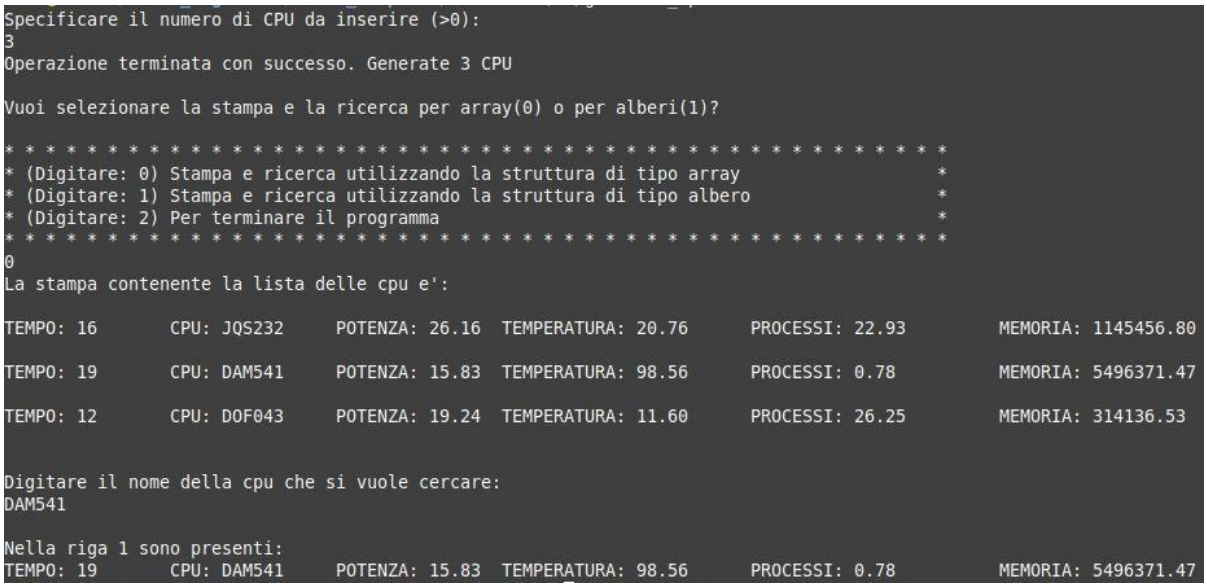
1.

```
Specificare il numero di CPU da inserire (>0):
5
Operazione terminata con successo. Generate 5 CPU

Vuoi selezionare la stampa e la ricerca per array(0) o per alberi(1)?

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                          *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                         *
* (Digitare: 2) Per terminare il programma                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
0
La stampa contenente la lista delle cpu e':

TEMPO: 13       CPU: EFD284     POTENZA: 6.08   TEMPERATURA: 73.20      PROCESSI: 42.77         MEMORIA: 20712525.01

TEMPO: 16       CPU: DKE711     POTENZA: 0.54   TEMPERATURA: 72.63      PROCESSI: 46.54         MEMORIA: 11595172.35

TEMPO: 13       CPU: SGZ254     POTENZA: 18.20  TEMPERATURA: 105.33     PROCESSI: 17.36         MEMORIA: 1627150.81

TEMPO: 2        CPU: BGV038     POTENZA: 14.92  TEMPERATURA: 11.42      PROCESSI: 47.38         MEMORIA: 10865426.83

TEMPO: 20       CPU: GPO066     POTENZA: 1.58   TEMPERATURA: 98.25      PROCESSI: 23.90         MEMORIA: 1444112.15
```

2.

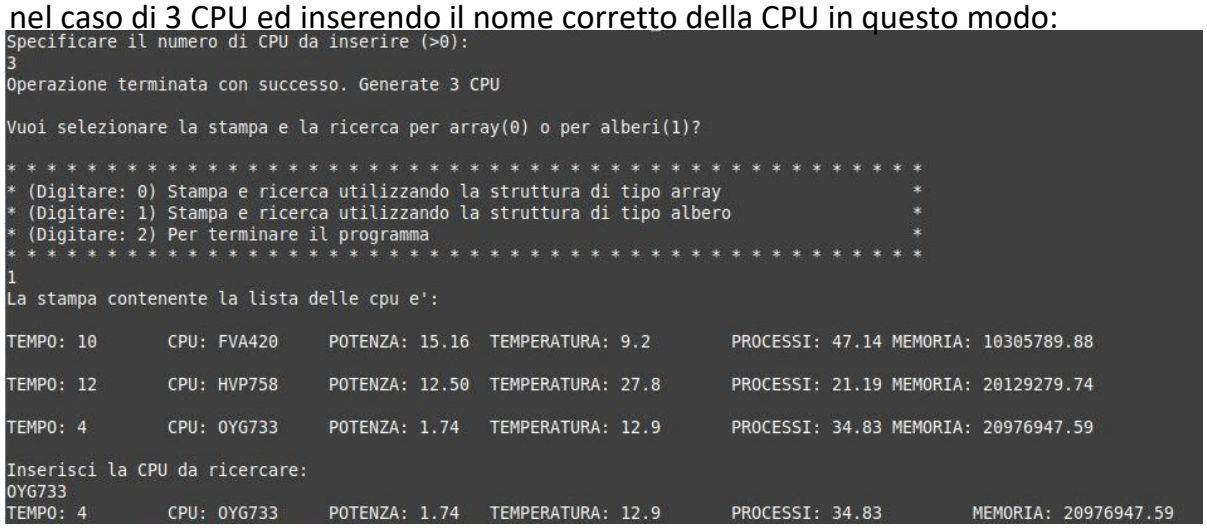
```
Specificare il numero di CPU da inserire (>0):
5
Operazione terminata con successo. Generate 5 CPU

Vuoi selezionare la stampa e la ricerca per array(0) o per alberi(1)?

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                          *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                         *
* (Digitare: 2) Per terminare il programma                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1
La stampa contenente la lista delle cpu e':

TEMPO: 4        CPU: SKH568     POTENZA: 14.96  TEMPERATURA: 19.3       PROCESSI: 44.03 MEMORIA: 18983556.89

TEMPO: 24       CPU: UEA657     POTENZA: 20.54  TEMPERATURA: 43.5       PROCESSI: 46.85 MEMORIA: 12130193.16

TEMPO: 8        CPU: VAK728     POTENZA: 13.70  TEMPERATURA: 16.5       PROCESSI: 36.02 MEMORIA: 18347022.45

TEMPO: 19       CPU: XIR710     POTENZA: 16.23  TEMPERATURA: 90.6       PROCESSI: 10.24 MEMORIA: 1566814.22

TEMPO: 9        CPU: ZDR850     POTENZA: 15.57  TEMPERATURA: 11.6       PROCESSI: 42.57 MEMORIA: 17087459.94
```

- La schermata quando si seleziona la struttura dati array, si presenta, nel caso di 3 CPU ed inserendo il nome corretto della CPU in questo modo:



- In caso di input errato per la ricerca di una CPU ci verrà segnalato che l'elemento non è presente:

- Mentre la schermata quando si seleziona la struttura dati di tipo albero, si presenta, nel caso di 3 CPU ed inserendo il nome corretto della CPU in questo modo:



- In caso di input errato per la ricerca di una CPU ci verrà segnalato che l'elemento non è presente:

➢ La schermata in caso di input errato nella selezione nel menù, si presenta in questo modo:

```
Vuoi selezionare la stampa e la ricerca per array o per alberi?

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                     *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                    *
* (Digitare: 2) Per terminare il programma                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
3
Input di selezione errato!

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                     *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                    *
* (Digitare: 2) Per terminare il programma                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
a
Input di selezione errato!

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                     *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                    *
* (Digitare: 2) Per terminare il programma                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

➢ Quando si inserisce, invece, 2 nel menu iniziale, il programma termina:

```
Vuoi selezionare la stampa e la ricerca per array(0) o per alberi(1)?

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* (Digitare: 0) Stampa e ricerca utilizzando la struttura di tipo array                     *
* (Digitare: 1) Stampa e ricerca utilizzando la struttura di tipo albero                    *
* (Digitare: 2) Per terminare il programma                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
2
Programma terminato...
```

ANALISI TEORICHE SULLA COMPLESSITÀ DEGLI ALGORITMI

Analisi teorica sulla complessità degli algoritmi di ricerca per array e alberi

Riguardo l'analisi teorica dell'algoritmo di ricerca per array il calcolo si divide in due casi:

- Caso in cui non sia presente l'elemento cercato;
- Caso in cui sia presente l'elemento cercato.

Nel caso l'elemento cercato non sia presente avremo un ciclo for che esegue 'n' iterazioni (dove n sono gli elementi presente nel file di testo) e per ogni iterazione esegue tre istruzioni quindi 3*n + 3 istruzioni totali. Otterremmo quindi una complessità pari a O(n).

Nel caso, invece, l'elemento cercato fosse presente invece il ciclo for eseguirebbe 'n' iterazioni e per ogni iterazione eseguirebbe 4 istruzioni (quella in piu' è l'assegnamento della variabile 'trovato') così da ottenere 4*n + 3 istruzioni totali e riottenere una complessità di O(n).

Quindi in entrambi i casi la complessità rimarrebbe invariata.

Dal momento che l'algoritmo trova tutte le CPU con lo stesso nome, e non una sola, non è possibile parlare di caso pessimo e caso ottimo in quanto in qualsiasi caso l'algoritmo eseguirà sempre 'n' istruzioni.

Per quanto riguarda l'analisi teorica dell'algoritmo di ricerca per alberi binari di ricerca anche questo si divide in due casi:

- Caso l'elemento cercato non sia stato trovato;
- Caso l'elemento cercato sia stato trovato.

In entrambi i casi otterremo la stessa complessità. Quindi avremo un'istruzione if annidata con la presenza di una funzione ricorsiva e, quindi, quello che otterremo sarà T(h - 1) + 1 ovvero O(h) dove sappiamo che h = $\log_2(n)$) (in quanto l'albero binario di ricerca un particolare tipo di albero binario), questo in virtù del fatto che ad ogni chiamata ricorsiva scenderemo di un livello tante volte quanto è h e l'albero non degenererà mai in una lista e quindi ottenere una complessità pari a O(n).

Studio sperimentale sulla complessità degli algoritmi di ricerca per array e alberi

Per quanto riguarda l'analisi della complessità sperimentale, si è pensato di utilizzare come metodo di misurazione il tempo di esecuzione degli algoritmi mediante il tempo di clock per entrambe le funzioni di ricerca. Tuttavia, essendo l'algoritmo di ricerca mediante la struttura dati ad albero con complessità O ($\log_2(n)$) troppo veloce in esecuzione anche con l'inserimento di grandi quantità di dati, abbiamo deciso di non far partire da 0 come primo valore l'asse delle ordinate ma direttamente dal valore più piccolo che abbiamo trovato, attraverso il tempo di clock. Quindi abbiamo allargato di molto il grafico per mettere in risalto il più possibile l'andamento del nostro algoritmo.

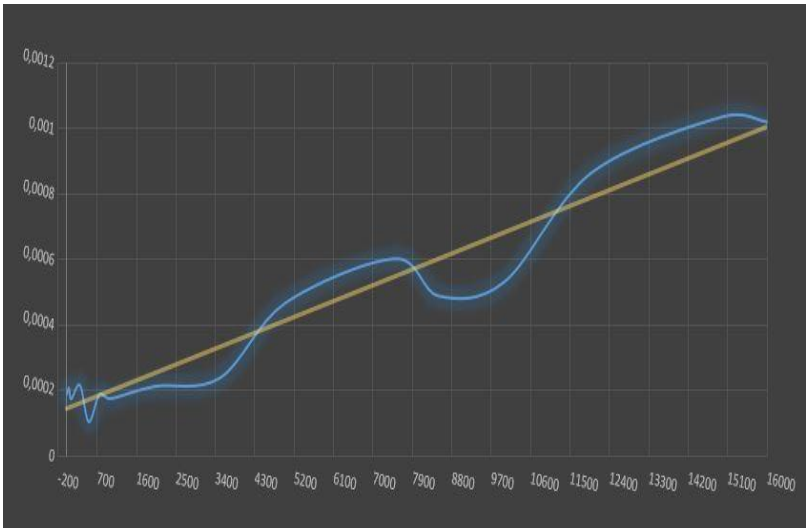
Per quanto riguarda i dati da acquisire, contenuti all'interno del file, si è deciso di operare richiedendo all'utente all'inizio del programma di inserire il numero desiderato, e generandoli quindi in maniera casuale, pur sempre rispettando le restrizioni poste nel programma principale.

Per quanto riguarda la generazione dei dati, si è deciso di aumentare il numero linearmente, crescendo attraverso un valore costante.

Nei grafici seguenti, sono evidenziate in blu le operazioni eseguite dagli algoritmi al crescere di n, e in arancione la trend-line della rispettiva complessità prospettata.

Ricerca con array

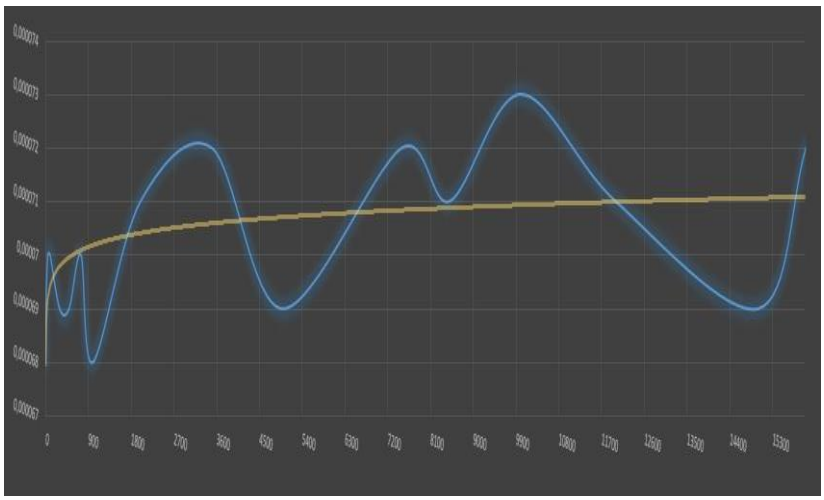
INPUT	TEMPO
1	0.000188
10	0.000192
50	0.000206
100	0.000173
300	0.000215
500	0.000103
750	0.000188
1000	0.000174
2000	0.000211
3500	0.000237
5000	0.000467
7500	0.000602
8500	0.000488
10000	0.000531
12000	0.000866
15000	0.001037
16000	0.001021



T(n) = O(n)

Ricerca con alberi

INPUT	TEMPO
1	0.000068
10	0.000069
50	0.000070
100	0.000070
300	0.000069
500	0.000069
750	0.000070
1000	0.000068
2000	0.000071
3500	0.000072
5000	0.000069
7500	0.000072
8500	0.000071
10000	0.000073
12000	0.000071
15000	0.000069
16000	0.000072



T(n) = O(log(n))

Conclusioni

Analizzando l'ultima analisi, facendo presente che lo scostamento tra i valori è nell'ordine di 10^(-6), e che quindi il grafico risulta "zoomato", si può notare come l'andamento segua il tracciato logaritmico, per quanto riguarda le operazioni di ricerca mediante la struttura dati di tipo albero.

Per quanto riguarda invece la funzione di ricerca mediante la struttura dati di tipo array il grafico riporta chiaramente un andamento di tipo lineare, come ci si aspettava, corrispondendo quasi con precisione alla trend-line con il medesimo scostamento tra i valori sempre nell'ordine di 10^(-6).