Progetto Algoritmi e Strutture Dati

Alessio Muzi | Informatica Applicata | A.S. 2019/2020

Sommario:

1 - Specifica del problema

Si progetti un programma ANSI C che effettua le seguenti elaborazioni:

• acquisizione dall'utente di un valore numerico n;

• generazione casuale di n numeri interi compresi tra 0 e 3n e di n stringhe di 10 caratteri;

• caricamento in un albero binario di ricerca di tutte le coppie (numero, stringa) di dati generati;

• stampa a monitor dei dati contenuti nell'albero secondo un ordine di visita scelto dall'utente tra anticipato, simmetrico e posticipato;

• stampa a monitor dei dati contenuti nell'albero ordinati in base alla chiave numerica;

• ricerca di un elemento scelto dall'utente (sulla base della chiave numero intero).

Per quanto riguarda l'analisi teorica si deve fornire la complessità corrispondente ad ognuna delle seguenti operazioni:

- inserimento di un elemento nell'albero;
- stampa dell'albero;
- stampa ordinata;
- ricerca di un elemento.

Oltre all'analisi teorica della complessità si deve effettuare uno studio sperimentale della stessa per le operazioni di inserimento, stampa e ricerca.

2 - Analisi del problema

Input:

-Un numero intero n per la generazione casuale di chiavi numeriche e stringhe di 10 caratteri, da inserire in una struttura dati organizzata ad albero binario di ricerca.

Output:

-Stampa a monitor dei dati contenuti nell'albero secondo un ordine di visita a scelta dell'utente tra anticipato, simmetrico e posticipato,

-Stampa a monitor dei dati contenuti nell'albero ordinati in base alla chiave numerica,

-Ricerca di un elemento scelto dall'utente.

Relazioni intercorrenti utili alla soluzione del problema:

-Le chiavi numeriche e le stringhe vengono inseriti tramite l'algoritmo di inserimento per alberi binari.

-Per eseguire l'operazione di stampa vengono usati i tre algoritmi di visita per alberi binari.

-Per eseguire l'operazione di ricerca viene usato l'algoritmo di ricerca per alberi binari di ricerca.

-Il numero dei nodi dell'albero binario di ricerca è uguale a n.

-I numeri generati casualmente devono essere compresi tra 0 e 3n.

-Ogni nodo deve avere come dati una chiave numerica e una stringa di 10 caratteri.

3 - Progettazione dell’algoritmo

Scelte di progetto:

Strutture dati:

Per la risoluzione del problema assegnato, verranno usati un albero binario di ricerca, come richiesto dalla consegna, e due array, uno monodimensionale e uno bidimensionale, per memorizzare le chiavi numeriche e le stringhe di 10 caratteri generate casualmente e facilitare l’inserimento di esse nell’albero binario.

Algoritmi e tecnice di risoluzione:

La consegna del problema richiede l’eseguimento di quattro operazioni: inserimento, stampa a scelta dell’utente, stampa ordinata e ricerca.

L’operazione di inserimento viene eseguita tramite l’utilizzo dell’omonimo algoritmo, il quale nel caso ottimo, quando l’albero binario è vuoto e l’elemento da inserire è la radice, ha complessità asintotica $T(n) = O(1)$, nel caso pessimo, quando l’elemento da inserire è figlio della foglia piu’ distante, ha complessità $T(n) = O(n)$, mentre nel caso medio è dimostrabile che la complessità asintotica, è di $T(n) = O(\log n)$. Inoltre, poiché l’algoritmo di inserimento impiegato funziona correttamente solo se le chiavi principali da inserire sono distinte, bisogna controllare che gli n numeri interi generati casualmente siano diversi.

L’operazione di stampa scelta dall’utente viene eseguita tramite l’utilizzo dei tre algoritmi di visita ricorsivi (anticipato, simmetrico e posticipato), mentre per la stampa ordinata si può ricorrere all’algoritmo di visita simmetrico, poiché l’utilizzo dell’albero binario di ricerca fa’ sì che l’ordine di visita rispetti l’ordine crescente delle chiavi numeriche. La complessità asintotica dei tre algoritmi è $T(n) = O(n)$ ed è dimostrabile tramite induzione. Intuitivamente, l’algoritmo di visita dovrà attraversare n nodi e un numero $<n$ di foglie con nessun figlio, ottenendo quindi una complessità lineare con una costante moltiplicativa variabile.

L’algoritmo di ricerca per alberi binari di ricerca ha la stessa complessità asintotica dell’algoritmo di inserimento (caso ottimo O(1), caso pessimo O(n), caso medio O(log n)), il che lo rende piu’ efficiente rispetto all’algoritmo di ricerca ottenuto modificando opportunamente gli algoritmi di visita anticipata, simmetrica o posticipata.

Passi dell’algoritmo:

-Inserimento del numero intero n;

-Allocazione dinamica della memoria per gli array;

-Generazione dei numeri casuali (controllando ogni volta che non ci siano doppioni);

-Generazione delle stringhe di 10 caratteri;

-Inserimento delle coppie nell’albero binario di ricerca;

-Selezione della visita da parte dell’utente;

-Stampa a monitor dei dati secondo l’ordine scelto dall’utente;

-Stampa a monitor dei dati secondo l’ordine delle chiavi numeriche;

-Selezione dell’elemento da ricercare da parte dell’utente;

-Ricerca dell’elemento selezionato;

-Stampa dell’esito della ricerca e dei rispettivi dati contenuti nel nodo.

4 - Implementazione dell'algoritmo

Il programma è strutturato su un unico file con 8 funzioni, di cui 3 si occupano dell'acquisizione e della validazione del numero n, della scelta della visita e dell'elemento da ricercare, 3 riguardano gli algoritmi di visita, 1 contiene l'algoritmo di inserimento e l'ultima contiene l'algoritmo di ricerca.

Gli array che contengono le chiavi numeriche e le stringhe sono allocate dinamicamente, per permettere loro di contenere un numero variabile di numeri e stringhe.

La scelta principale per l'implementazione dell'algoritmo risolutivo del problema riguarda la struttura dati impiegata - l'albero binario di ricerca – la quale deve essere ampliata per poter contenere due chiavi, una principale e una secondaria. Il primo campo rimano il campo `int valore`, mentre il campo aggiunto sarà di tipo `char *stringa`, per consentire l'inserimento delle stringhe nei vari nodi.

Per evitare problemi a tempo di esecuzione, i due puntatori `*radice` e `*stringhe` vengono inizializzati a `NULL`.

Per garantire la generazione di numeri casuali, viene sfruttata la funzione `rand()` della libreria `<stdlib.h>` che produce dei numeri pseudocasuali, cioè che hanno approssimativamente le caratteristiche di un reale processo casuale, come il lancio di un dado. Poiché la funzione `rand()` genera un numero compreso tra 0 e `RAND_MAX` definito nella libreria, bisogna usare l'operatore modulo `%` per ottenere un numero compreso tra 0 e 3n nel caso delle chiavi numeriche, mentre nel caso dei caratteri l'operatore modulo viene usato per ottenere numeri compresi tra 0 e 52 che, sommati al carattere `'A'`, li trasforma in lettere e caratteri.
Inoltre, la funzione `rand()` richiede un seme di generazione che renda il processo sempre diverso ad ogni utilizzo del programma. Per questo motivo il generatore viene implementato insieme alla funzione `time()` della libreria `<time.h>`, la quale trasforma il tempo del calendario in un numero che sarà necessariamente sempre diverso.

Per verificare che tutte le chiavi numeriche siano distinte, il programma controlla che ad ogni iterazione `i` del generatore di numeri casuali, l'elemento `chiavi[i]` sia diverso da tutti i precedenti elementi `chiavi[k]` memorizzati nell'array tramite un ciclo `for` .
Questo problema invece non si presenta per le stringhe, poiché esse sono chiavi secondarie e non pregiudicano il funzionamento dell'algoritmo di inserimento.

5 - Testing del programma

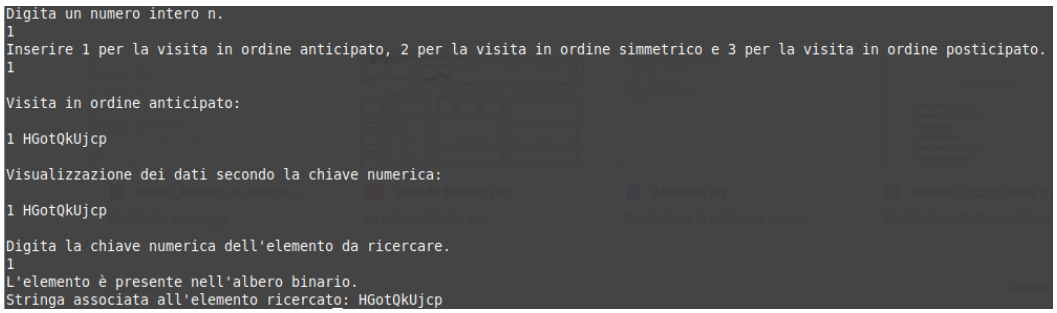
I casi significativi per testare correttamente il programma sono:

1) Albero con un solo nodo (radice)
2) Albero con piu' di un nodo
3) Albero in cui è presente l'elemento ricercato
4) Albero in cui non è presente l'elemento ricercato

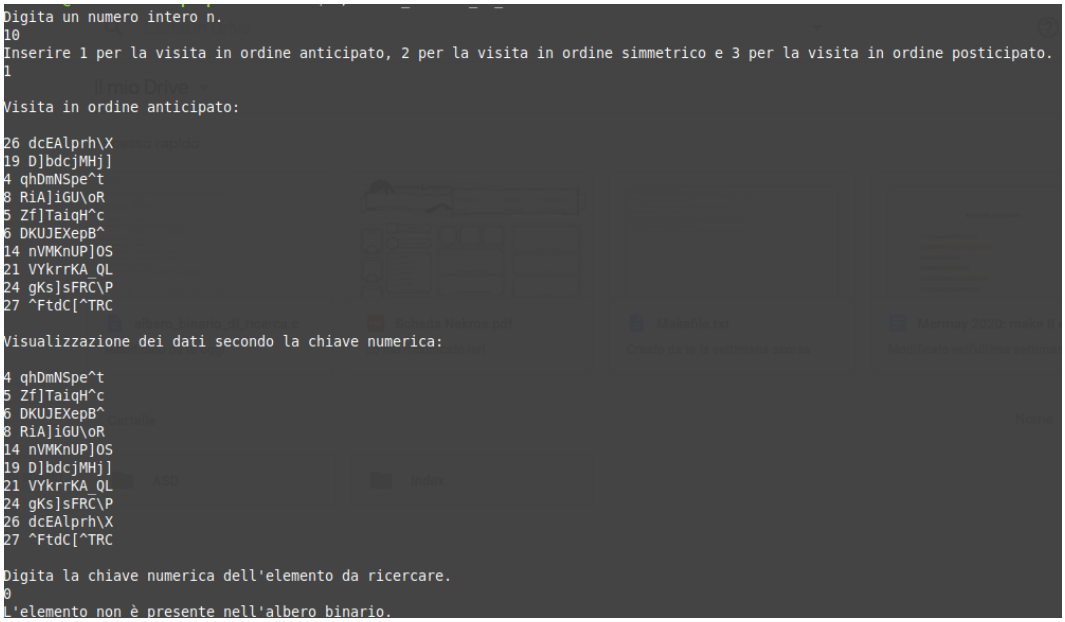
Inoltre bisogna testare i tre algoritmi di visita, secondo l'ordine anticipato, simmetrico e posticipato.

Per testare i casi in cui l'albero abbia piu' di un nodo viene usata un numero n arbitrario pari a 10.

Caso 1: Albero con un solo nodo (radice):



Caso 2: Albero con piu' di un nodo (ordine anticipato):



Caso 3: Albero in cui è presente l'elemento ricercato (ordine simmetrico):

```
Digita un numero intero n.
10
Inserire 1 per la visita in ordine anticipato, 2 per la visita in ordine simmetrico e 3 per la visita in ordine posticipato.
2

Visita in ordine simmetrico:

10 qVkaRXbnfV
11 `sokATmYsH
12 jEcg]lQ\th
13 _ArteKXEUC
15 KEAbIqLKgI
23 ^tLRKrZ[VX
24 pJ`NeUdNqm
25 T[^XSVcKhU
26 OM]FE`EmmL
28 OV_JNCIqEZ

Visualizzazione dei dati secondo la chiave numerica:

10 qVkaRXbnfV
11 `sokATmYsH
12 jEcg]lQ\th
13 _ArteKXEUC
15 KEAbIqLKgI
23 ^tLRKrZ[VX
24 pJ`NeUdNqm
25 T[^XSVcKhU
26 OM]FE`EmmL
28 OV_JNCIqEZ

Digita la chiave numerica dell'elemento da ricercare.
15
L'elemento è presente nell'albero binario.
Stringa associata all'elemento ricercato: KEAbIqLKgI
```

Caso 4: Albero in cui non è presente l'elemento ricercato (ordine posticipato):

```
Digita un numero intero n.
10
Inserire 1 per la visita in ordine anticipato, 2 per la visita in ordine simmetrico e 3 per la visita in ordine posticipato.
3

Visita in ordine posticipato:

1 sMQL]alVqg
3 \pOEBZXCGj
6 smN\HlrGrf
10 QqGKgRXDLs
8 \dN\TBg^VX
23 i`nipeHNtj
29 cWQHoSNdfR
26 MJPSBYPllS
16 eAkKJ_mJrO
5 jiVopYIhLK

Visualizzazione dei dati secondo la chiave numerica:

1 sMQL]alVqg
3 \pOEBZXCGj
5 jiVopYIhLK
6 smN\HlrGrf
8 \dN\TBg^VX
10 QqGKgRXDLs
16 eAkKJ_mJrO
23 i`nipeHNtj
26 MJPSBYPllS
29 cWQHoSNdfR

Digita la chiave numerica dell'elemento da ricercare.
7
L'elemento non è presente nell'albero binario.
```

6 - Valutazione della complessità del programma

Le operazioni di cui viene richiesta la complessità asintotica sono quelle di inserimento, stampa e ricerca. Gli esperimenti sono stati effettuati modificando opportunamente il programma con l'inserimento di contatori dei passi base nelle apposite funzioni e ripetendo 10 volte per ogni input di n l'esecuzione del programma. Il risultato finale è stato ottenuto facendo la media dei 10 risultati intermedi (arrotondata alla cifra delle unità).

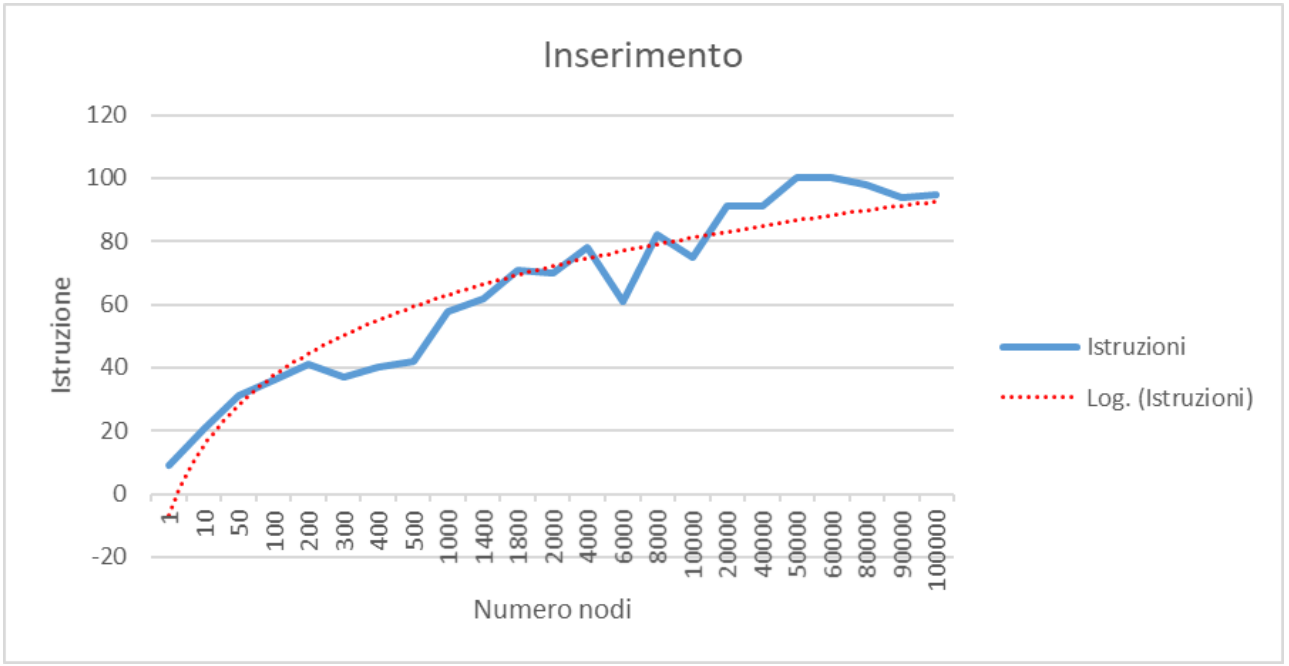
Nel caso dell'inserimento vengono calcolati i passi base richiesti per l'inserimento dell'ultimo elemento.

Nel caso della stampa viene presa come esempio la visita secondo la chiave numerica (ovvero l'ordine di visita simmetrico) poiché i tre algoritmi agiscono in modo equiparabile ed hanno complessità teorica equivalente.

Nel caso della ricerca viene generato un ulteriore numero casuale compreso tra 0 e 3n il quale diventa l'elemento da ricercare.

Inserimento

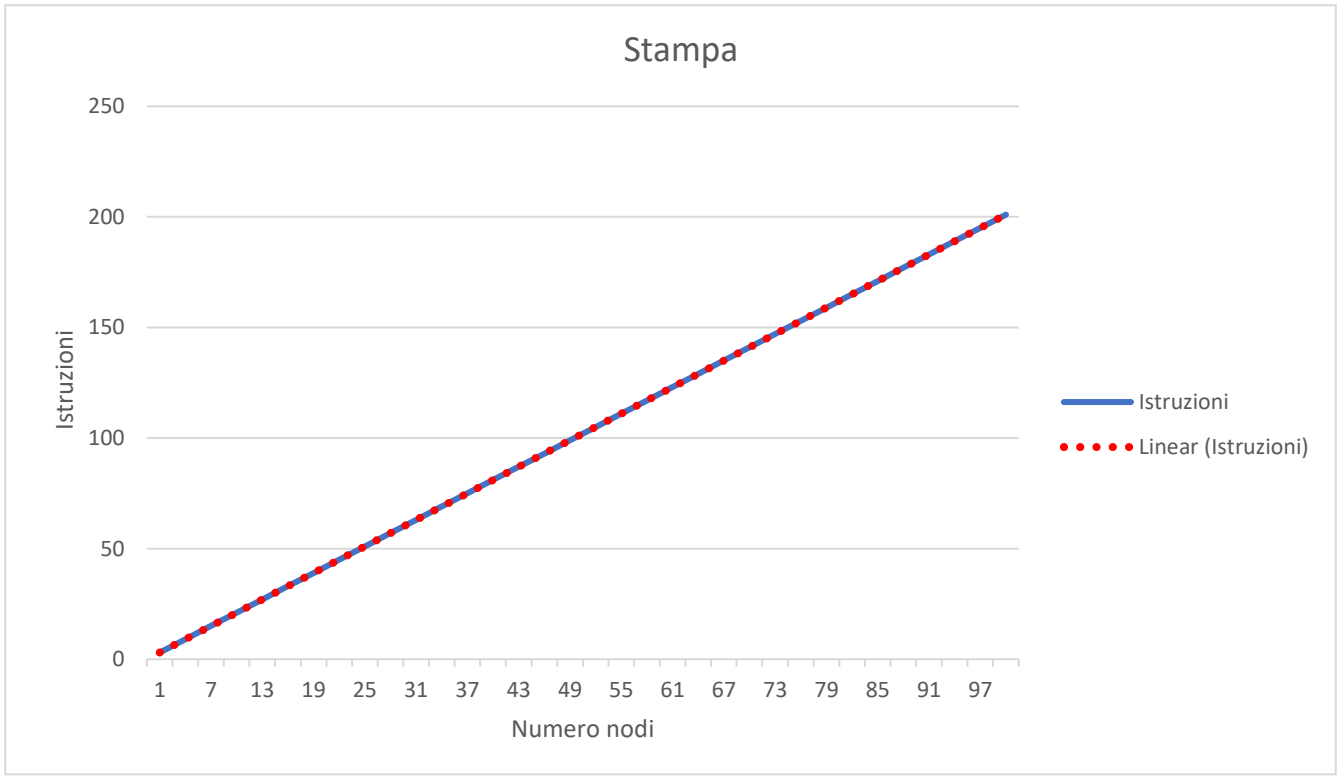
n	Istruzioni
1	9
10	21
50	31
100	36
200	41
300	37
400	40
500	42
1000	58
1400	62
1800	71
2000	70
4000	78
6000	61
8000	82
10000	75
20000	91
40000	91
50000	100
60000	100
80000	98
90000	94
100000	95



$$T(n) = O(\log(n))$$

n	Istruzioni
1	3
4	9
7	15
10	21
13	27
16	33
19	39
22	45
25	51
28	57
31	63
34	69
37	75
40	81
43	87
46	93
49	99
52	105
55	111
58	117
61	123
64	129
67	135
70	141
73	147
76	153
79	159
82	165
85	171
88	177
91	183
94	189
97	195
100	201

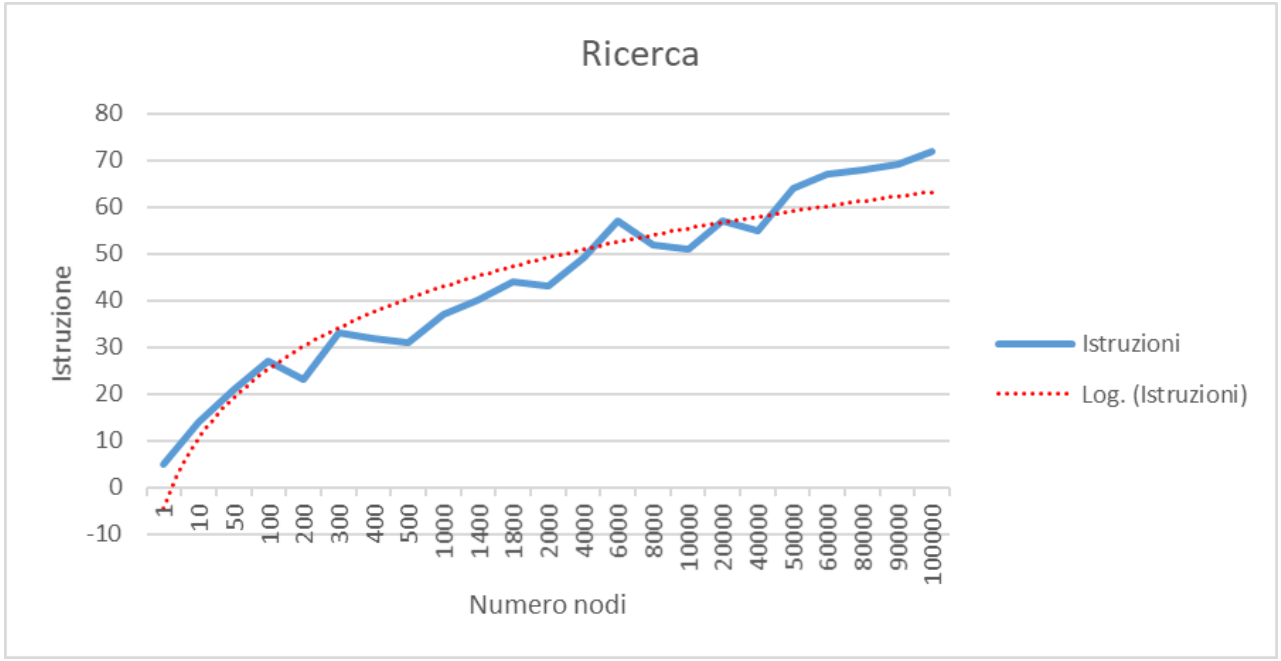
Stampa



$$T(n) = O(n)$$

Ricerca

n	Istruzioni
1	5
10	14
50	21
100	27
200	23
300	33
400	32
500	31
1000	37
1400	40
1800	44
2000	43
4000	49
6000	57
8000	52
10000	51
20000	57
40000	55
50000	64
60000	67
80000	68
90000	69
100000	72



$$T(n) = O(\log(n))$$