ALGORITMI E STRUTTURE DATI

Autore: GIORGIO D'ANGELO

1) Hashtable di disjoint set

Si supponga di dover memorizzare un insieme di coppie <key,value> contenute in un file. key è numero intero e value è una stringa alfanumerica di lunghezza arbitraria. Inizialmente le singole coppie sono memorizzate in insiemi disgiunti i quali sono memorizzati in una hash table. Si progetti ed implementi una struttura dati, composta da un hash table in cui ogni cella punta ad un insieme disgiunto, che preveda i seguenti metodi: MAKE_SET(), FIND_SET(), UNION(). Dotare il programma di un menu da cui sia possibile richiamare le suddette operazioni.

1.1) Descrizione del problema

Mi viene chiesto di creare un hashtable di cui ogni cella punta all'insieme disgiunto e implementare le funzione make-set(), find-set(), union().

Una tabella di hash è un struttura dati molto efficiente per implementare i dizionari. E' molto importante perché l'inserimento di una chiave all'interno della tabella hash segue dei criteri ben precisi e da qui nascono le funzioni di hashing.

La funzione di hash deve essere sempre deterministica in quanto applicando la funzione hash ad una stessa chiave mi deve restituire la stessa posizione e la posizione che mi restituisce deve essere equiprobabile cioè nessuna posizione deve essere favorita rispetto ad un'altra in modo tale da non incorrere a collisioni.

Come funzione hash ho usato il metodo della divisione, consiste in questo: Quando sto inserendo la chiave ad esempio 50 farò chiave modulo con il size del mio hashtable e mi ritornerà il resto della divisione e quello sarò l'indice dove inserirò la chiave.

Questo è molto importante per la ricerca perché ricercare un chiave significa calcolarne il modulo quindi so esattamente in quale insieme disgiunto si trova e quindi devo cercare all'interno di quell'insieme se la chiave esiste o meno.

Mi viene chiesto di implementare le seguenti funzioni:

La make-set() consiste nel creare un insieme disgiunto con un unico elemento che gli passiamo alla funzione e sarà anche il rappresentate dell'insieme.

La find-set() dato un insieme disgiunto, si passa alla find-set() la chiave che si vuole cercare ,se esiste tale chiave all'interno dell'insieme si restituisce il rappresentante.

La union() di due insiemi disgiunti ,vengono passate alla union in input due chiavi dove controlleremo se la find-set della prima chiave è diversa dalla find-set della seconda chiave ossia se i tuoi rappresentati degli insiemi sono diversi,

se sono diversi significa che le chiavi non appartengono allo stesso insieme disgiunto e quindi possiamo unire gli insiemi.

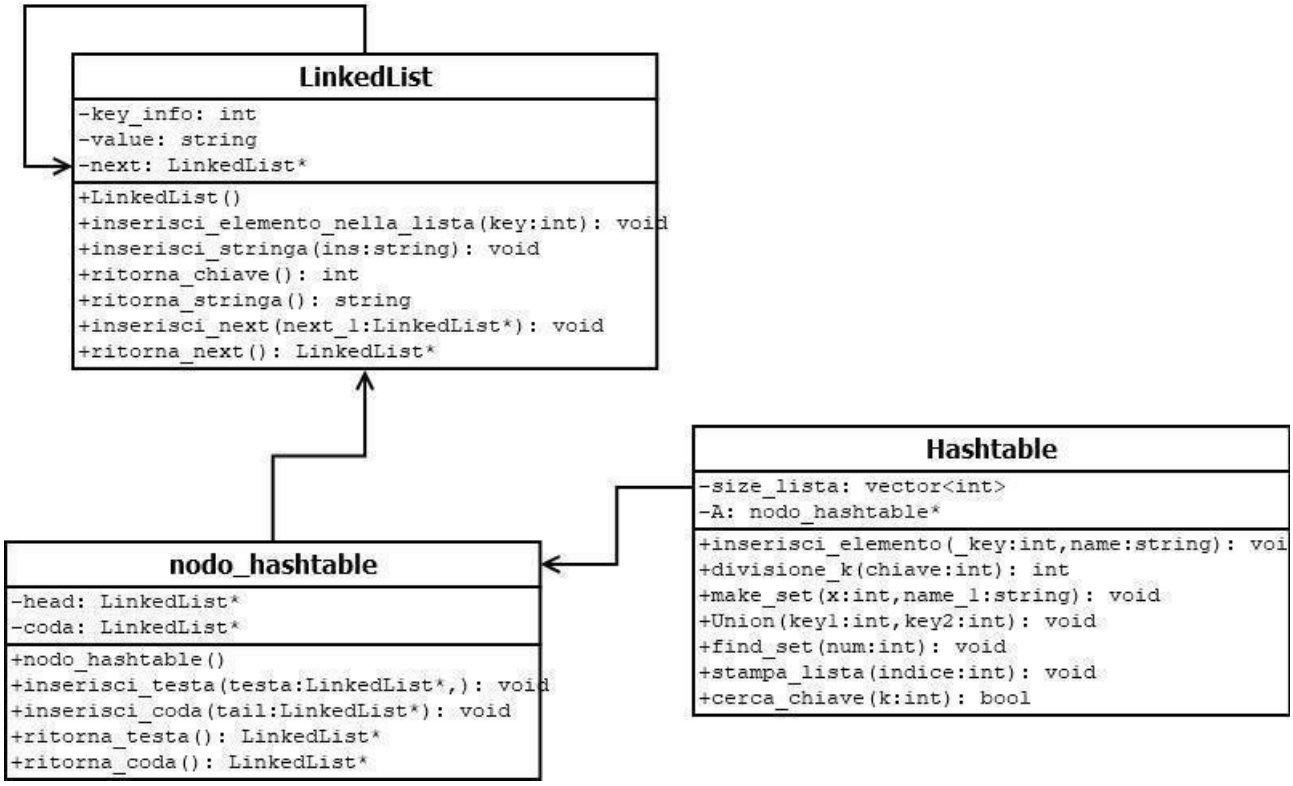
1.2) Descrizione strutture dati utilizzate

Come strutture dati ho utilizzato per la costruzione dell'hashtable, una lista che rappresenta l'insieme disgiunto e un vettore dove ogni cella punterà ad una lista quindi un insieme disgiunto.

1.3) Formato dati in input/output

Leggo un file input in cui ogni rigo è rappresentato da un intero e una stringa.

1.4) UML



1.5) Descrizione algoritmo

L'algoritmo funziona in questo modo:
Leggo da un file in input le chiave e le stringhe, per ogni chiave userò la funzione hash (metodo della divisione) controllerò se la cella corrispondente al modulo della funzione hash è già occupata o meno indipendentemente se è occupata o meno mi creò un oggetto LinkedList che contiene la chiave la stringa e un puntantore.
Se la cella non è occupata allora mi creerò l'oggetto hashtable che praticamente è come se fosse un nodo sentinella che punta alla testa e alla coda della lista e inserirò l'indirizzo dell'oggetto all'interno della cella. Se la cella è già occupata allora mi dichiaro una variabile temporanea in cui mi salvo il valore della testa della lista corrente dal nodo sentinella e scorrerò la lista fino a trovare il null e inserirò l'elemento (quindi è

un inserimento in coda) e ogni volta aggiornerò la nuova coda, per ogni lista che creo avrò un vector che mi conterrà gli elementi presenti nella lista, questo mi servirà per la union.
Il primo elemento inserito nella cella sarà il rappresentante dell'insieme. La union funziona in questo modo, praticamente avremo un menu dove potremmo inserire le due chiave che vogliamo inserire, effettuerò un controllo per vedere se le chiavi che sto cercando esistono e se le chiavi che sto cercando di inserire non siano uguali, quindi con l'informazione che ho nel vector che tiene conto del size delle liste, unirò la lista con il size minore a quella con il size maggiore in modo da diminuire le operazioni di inserimento.
Quindi semplicemente mi scorrerò la lista con size minore e la inserirò nella coda della lista con il size maggiore.
Con la union nasce un problema una volta che ho inserito la lista 2 nella lista 1, se sto cercando ad esempio una chiave e quest'ultima si troverebbe nella lista 2 non la troverò perché ho inserito la lista 2 nella lista 1 e ciò comporterebbe un errore dunque farò puntatare la testa della lista 2 alla testa della lista 1.
La find-set() vuole che dato una chiave restituisco il suo rappresentante.

Quindi inserisco la chiave, con il metodo della divisione capirò in quale insieme disgiunto si trova, scorrerò la lista se trovo la chiave restituirò la testa della lista ossia dove si trova il rappresentante.

La make-set() vuole creare una lista con un solo elemento.

Quindi inserisco la chiave e la stringa, controllerò con la funzione hash della chiave se nell'indice restituito se il vettore con quell'indice è già occupato da un altro insieme, se quella cella è vuota creerò il nuovo insieme disgiunto , se invece è occupata la make-set fallisce ma effettuerà un controllo e ci consiglierà la prima cella libera dove poter effettuare una make-set

1.5) Complessità

La complessità è data dalle operazioni che effettuiamo nell'hashtable.
Per popolare l'hashtable avremo un complessità data dalla funzione hash, nel caso migliore l'inserimento è O(1) se la cella è vuota, nel caso peggiore è O(n) in quanto la cella è già occupata e dobbiamo scorrere la lista e inserire il nuovo elemento in coda.
L'operazione find-set avrà come complessità O(1) nel momento in cui il find-set della chiave è la chiave stessa ,ossia la chiave che sto cercando è anche il rappresentante dell'insieme, mentre negli altri casi avrà complessità O(n) perché prima devo cercare se una determinata chiave esiste nell'insieme e dopo restituisco il rappresentante che è sempre l'operazione meno costosa in quando è la prima chiave della lista.
La make-set avrà complessità O(1) poiché vuole creare un insieme con un solo rappresentante quindi si tratterà di fare un inserimento con complessità O(1).
Infine abbiamo la union, per migliorare la complessità abbiamo un vector in cui abbiamo i size delle varie liste, quando voglio unire due liste verifico quale sia la lista più piccola e quest'ultima verrà inserita nella coda della lista più grande, l'inserimento in questo caso è O(n).
Poichè inserirò N elementi e l'inserimento nel caso peggiore ha complessità O(n), la complessità finale è $O(n^2)$.

1.6) Test/risultati

Test con il file input.txt

25,ciao
26,lala

35,

qu

a

90,

cia

s

20,giorgi

50,francesco

2,salvatore

10,ferdinando

21,giovanni

47,paolo

94,luca

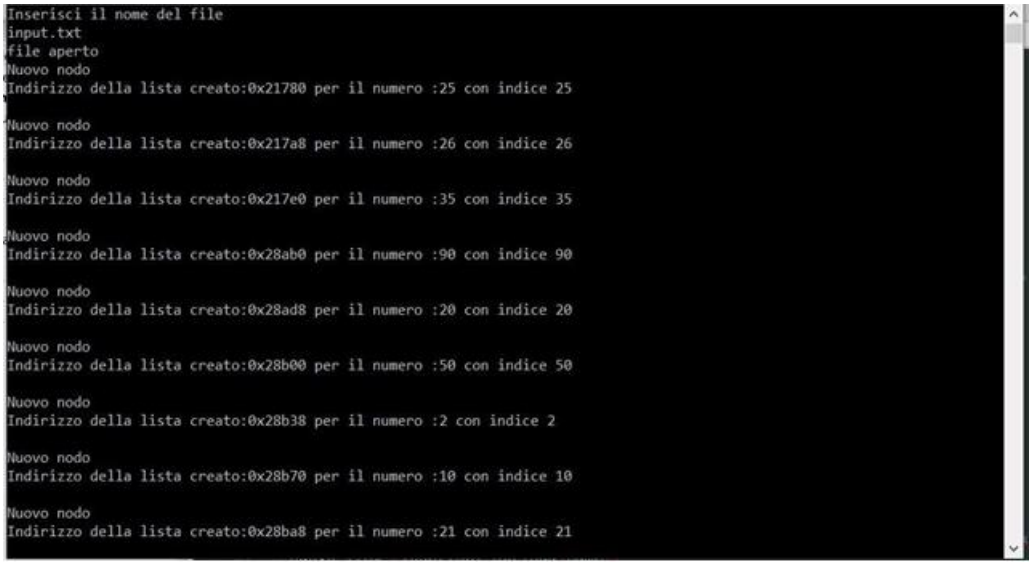
703,francesco

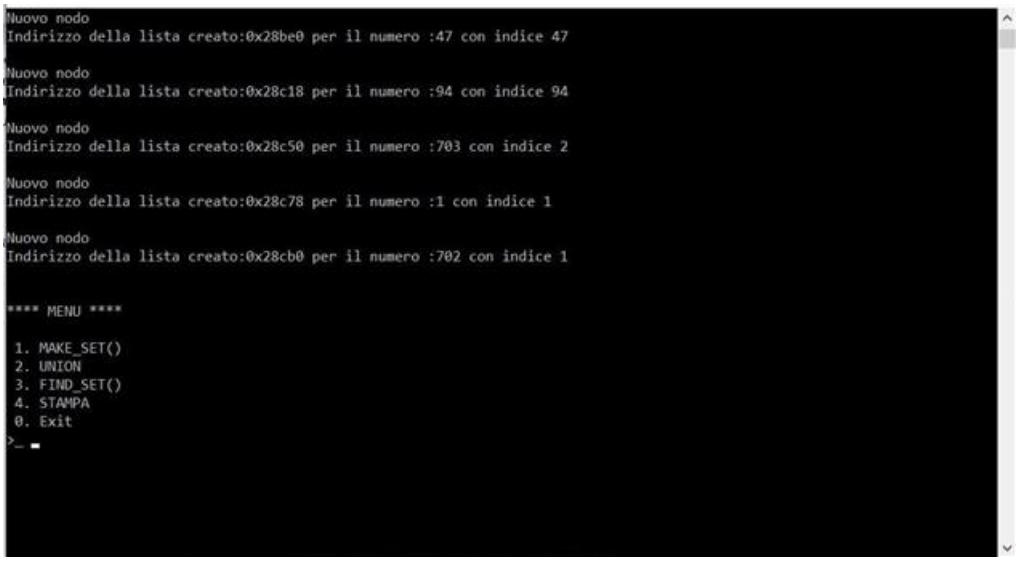
1,paola

702,daniela

Nel primo passo inseriremo il nome del file in questo caso input.txt

Andremo a vedere gli inserimenti all'interno dell'hashtable

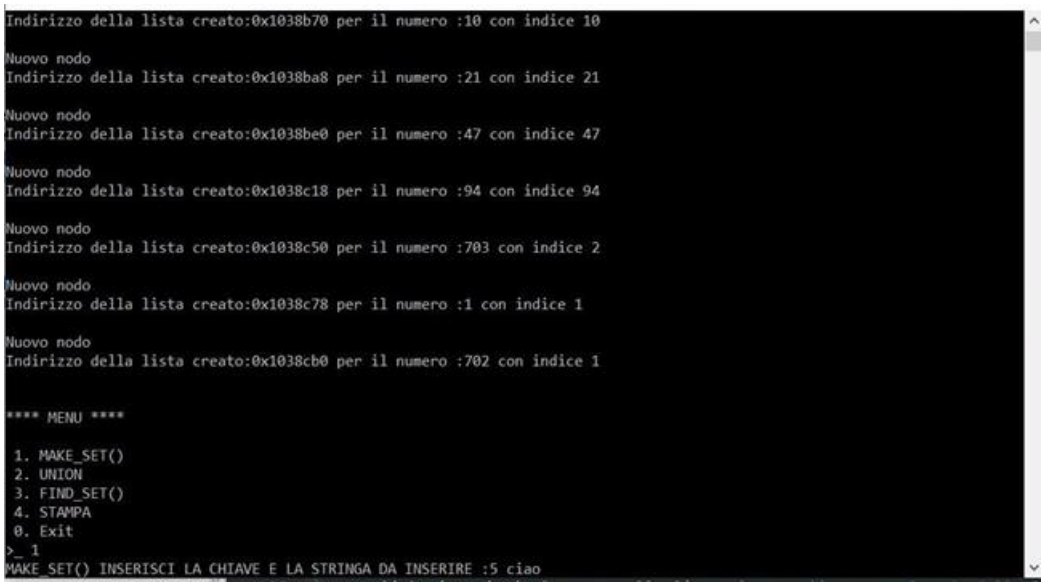




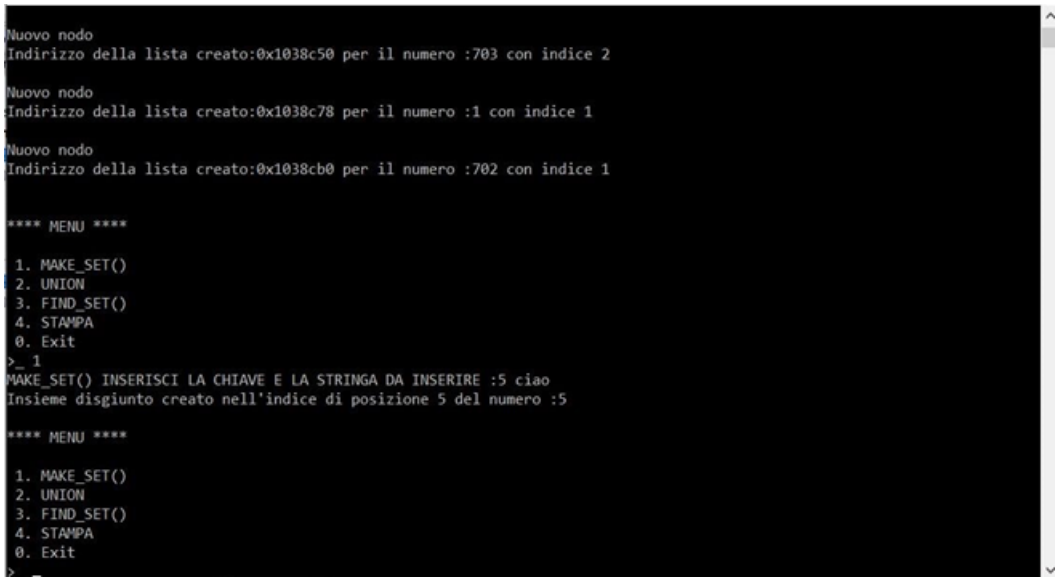
MAKE-SET

Adesso scegliamo quale opzione eseguire, sceglierò la make-set e voglio inserire la chiave

5 e stringa ciao



Ecco l’inserimento



UNION

Sceglierò di fare la union inserendo due chiavi, 5 e 1

```
 4. STAMPA
 0. Exit
>_ 2

 UNION
INSERISCI LE CHIAVI DA UNIRE :5 1
Union riuscita

**** MENU ****

 1. MAKE_SET()
 2. UNION
 3. FIND_SET()
 4. STAMPA
 0. Exit
>_ 4
STAMPA L'INSIEME DISGIUNTO INSERENDO IL NUMERO :
5
Valori della lista con  Chiave :1 Stringa :paola
Valori della lista con  Chiave :702 Stringa :daniela
Valori della lista con  Chiave :5 Stringa :ciao

**** MENU ****

 1. MAKE_SET()
 2. UNION
 3. FIND_SET()
 4. STAMPA
 0. Exit
>
```

La union è andata a buon fine e sono state unite le due liste poiché il size della lista di 1 è maggiore rispetto a quello della lista di 5 in quanto contiene un solo elemento ,la lista 5 è stata unita in coda alla lista 1

Find-set: voglio il rappresentante di 5

```
 4. STAMPA
 0. Exit
>_ 2

 UNION
INSERISCI LE CHIAVI DA UNIRE :5 1
Union riuscita

**** MENU ****

 1. MAKE_SET()
 2. UNION
 3. FIND_SET()
 4. STAMPA
 0. Exit
>_ 4
STAMPA L'INSIEME DISGIUNTO INSERENDO IL NUMERO :
5
Valori della lista con  Chiave :1 Stringa :paola
Valori della lista con  Chiave :702 Stringa :daniela
Valori della lista con  Chiave :5 Stringa :ciao

**** MENU ****

 1. MAKE_SET()
 2. UNION
 3. FIND_SET()
 4. STAMPA
 0. Exit
>
```