Università degli Studi di Torino
Corso di Laurea Magistrale in Informatica
Algoritmi e Strutture dati

Relazione esercitazione 3

Alessandro Clocchiatti
Matricola 909105

Consegna
Si implementi una libreria per la struttura dati Hash Map, tenendo conto delle seguenti indicazioni:

- Una Hash Map rappresenta un insieme di associazioni del tipo `<K,V>`, dove `K` è una chiave e `V` è il valore ad essa associato;
- In una Hash Map, non possono esservi chiavi ripetute;
- L'implementazione sfrutta un meccanismo di hashing;
- L'implementazione deve offrire le seguenti operazioni:
  - Creazione di una Hash Map vuota;
  - Distruzione di una Hash Map (con conseguente deallocazione della memoria associata);
  - Verifica se una Hash Map è vuota;
  - Recupero del numero di associazioni presenti in una Hash Map;
  - Verifica se la chiave specificata è presente in una Hash Map;
  - Inserimento in una Hash Map di un'associazione di tipo `<K,V>`;
  - Recupero da una Hash Map dell'eventuale valore, associato alla chiave specificata;
  - Cancellazione da una Hash Map dell'eventuale associazione con una chiave specificata;
  - Recupero dell'insieme delle chiavi presenti in una Hash Map;
- Il codice che implementa la Hash Map deve essere generico (nel senso che deve consentire di inserire associazioni `<K,V>` di cui non è noto a tempo di compilazione né il tipo della chiave `K`, né quello del valore `V`) e non deve assumere alcuna cardinalità massima per l'insieme di associazioni che possono essere ospitate nella Hash Map.

Unit Testing
Implementare gli unit-test degli algoritmi secondo le indicazioni suggerite nel documento Unit Testing.

Uso delle funzioni implementate
Il file *hashes.txt* contiene circa sei milioni di coppie di interi. Il primo elemento di ogni coppia rappresenta una chiave, il secondo elemento rappresenta un valore. Ogni coppia è scritta su una linea. Gli elementi della coppia sono separati da una virgola.
Si esegua quanto segue e si scriva una breve relazione riguardante i risultati ottenuti:

- si carichi il contenuto del file in un oggetto di tipo Hash Map; si misurino i tempi di caricamento;
- si carichi il contenuto del file in un array allocato staticamente (*hashes.txt* contiene 6321078 coppie di interi) ordinato per chiave (la decisione a proposito di quando e come effettuare l'ordinamento è lasciata allo studente e deve essere presa seguendo un criterio di economia dei tempi di esecuzione); si misurino i tempi di caricamento;
- si estraggano a caso 10000000 numeri interi tra 0 e 10000000 e li si memorizzi in un array keys;
- si misuri il tempo necessario per recuperare i valori associati alle chiavi contenute nell'array keys usando la Hash Map;
- si misuri il tempo necessario a recuperare i valori associati alle chiavi contenute nell'array keys usando una ricerca binaria sull'array ordinato;
- si controlli che il numero di chiavi reperite con successo sia identico nei due casi presi in considerazione.

Nota: Il numero esatto di chiavi che riuscirete a reperire dipende dai valori che avete generato casualmente. Dato però che il dataset contiene $6\,321\,078 \simeq (1 - 1/e) \cdot 10^7$ chiavi distinte estratte nell'intervallo $[0 \dots 10^7)$, un semplice ragionamento probabilistico suggerisce che il numero di chiavi che dovreste riuscire a reperire facendo N tentativi è circa $(1 - 1/e) \cdot N$ e quindi, nello 7 specifico dell'esercizio proposto, tale numero è di nuovo $(1 - 1/e) \cdot 10^7$.

Nota 2: Non è necessario che entrambe le prove (il test con array e quello con la Hash Map) siano eseguite durante la stessa esecuzione del programma. Nel caso si proceda eseguendo i due test in due esecuzioni separate, ci si accerti che il seme usato per la generazione dei valori contenuti nell'array keys sia lo stesso in entrambi i casi.

---

1. Risultati

In questo esercizio si è scelto di implementare una tavola di Hash con concatenamento, in cui la funzione hash viene richiesta all'utente. Inoltre è prevista un'operazione di re-hashing nel caso in cui la dimensione dell'hash map (`size`, numero di associazioni inserite) è superiore o uguale al 75% della capacità totale.
Inoltre sono state introdotte due funzioni che permettono di creare una hash map vuota, una in cui la capacità è fissa e una in cui la capacità viene decisa dall'utente.

Il caricamento dei dati e la ricerca delle chiavi nelle due strutture dati hanno impiegato i seguenti tempi:

	Hash map	**Array**
Loading data	$6.14\ sec$	$2.37\ sec$
Retrieve entries	$3.16\ sec$	$19.91\ sec$

Il caricamento dei dati risulta essere più lento nel caso dell'hash map. Questo può dipendere da due fattori: scelta della funzione hash e capacità della hash map. Nel caso in cui la funzione hash non garantisca una distribuzione uniforme dei valori oppure nel caso in cui la capacità dell'hash map non sia adeguata rispetto al numero di associazioni inserite, il numero di collisioni aumenta, aumentando il tempo di caricamento. Inoltre, la presenza dell'operazione di re-hashing determina un rallentamento dell'esecuzione. In particolare, minore è la capacità iniziale, maggiore è il numero di operazioni di ridimensionamento necessarie, aumentando il tempo di caricamento.

L'ordinamento dell'array è stato fatto attraverso l'algoritmo Quick-sort, utilizzando la libreria sviluppata nella prima esercitazione. La complessità dell'ordinamento è $O(n \log n)$ e richiede 4,83 secondi.

In una tabella hash con concatenamento la complessità di ricerca di una chiave è in media $O(1 + \alpha)$ dove α è il numero di elementi medio in una lista

$$\alpha = \frac{N}{m}\ dove$$

N: $numero\ di\ elementi$
m: $capacità\ della\ tabella\ hash$

Nel caso in cui la capacità dell'hash map (m) sia proporzionale al numero degli elementi (N), la complessità di ricerca è $O(1)$.
La ricerca delle chiavi nell'array, siccome quest'ultimo è ordinato in base alle chiavi, viene fatta tramite una ricerca binaria (dicotomica), con complessità $O(\log n)$.
Nonostante il numero di chiavi reperite siano le stesse, la ricerca nell'hash map necessita di un tempo decisamente minore rispetto alla ricerca nell'array statico.