Relazione esercizio 3

Scelte implementative

Abbiamo fatto delle precise scelte implementative durante lo sviluppo dell'esercizio, qui elencate:

1. Tipo di indicizzazione: Dovendo rispettare la clausola del non poter assumere la quantità massima di associazioni ospitabili dalla mappa, abbiamo optato per l'indicizzazione concatenata.
2. Funzione di hash: Abbiamo optato per una funzione di hash il più semplice e intuitiva possibile, sfruttando il resto della divisione per la dimensione della mappa.
3. Load-factor e ridimensionamento della mappa: Per mantenere un accesso efficiente ai dati abbiamo deciso di implementare una funzione di ridimensionamento della mappa. La mappa viene ridimensionata se, a seguito di un inserimento il load-factor supera un limite (fornito dall'utente alla funzione che crea la mappa) questa viene ridimensionata il doppio della dimensione precedente. Mentre, se a seguito di una cancellazione il load-factor va al di sotto di un limite prefissato di 0.25 allora la hash-map viene ridimensionata alla metà della dimensione precedente (la funzione di ridimensionamento opera in questo modo: crea una nuova mappa, si avvale della funzione *hash_map_keys_used* per ricavare tutte le key memorizzate nella vecchia mappa, usa le funzioni *insert* e *search* per fare gli inserimenti nella nuova mappa).

Strutture utilizzate

Per implementare la Hash-Map usiamo due strutture, *HashMap* che funge da tipo opaco ed *HashEntry* che svolge la funzione di nodo per implementare la lista linkata, qui vengono memorizzati i riferimenti a "key" e "value" delle associazioni.

Funzioni di supporto

Utilizziamo due funzioni di supporto chiamate *sup_delete* e *sup_insert* al fine di modulare il codice che esegue le operazioni di cancellazione e inserimento delle entries. Più precisamente la *sup_delete* è stata creata per gestire in modo più agevole la cancellazione.

Applicazione – Hash-map vs Array

N.B. Questi dati sono stati raccolti dando in input un load_factor pari a 2 e con una dimensione iniziale della mappa pari a numero di entries del file da analizzare e tendendo tutti i file nella stessa cartella.

	Array	Hash-Map
Inserimento dati	3.250000 s	3.187500 s
Ricerca dati	3.203125s	0.71875 s

I dati raccolti supportano le ipotesi teoriche secondo le quali per quanto riguarda le operazioni di ricerca "Hash-Map" risulta più efficiente rispetto alla rivale "Array".