1 Hashing

L'**hashing** è una **tecnica alternativa** agli alberi binari di ricerca (capitolo 4.4) o agli array associativi, utilizzata per realizzare i dizionari.

A differenza dell'implementazione tramite alberi, nel quale si riusciva a mantenere la stessa complessità nei casi di `insert()`, `lookup()` e `remove()`, idealmente, la cosa migliore sarebbe **mantenere una complessità costante** per tutte le operazioni, avendo però una **complessità inferiore**. Questa implementazione ideale prende il nome di **tabelle di hash**.

	Array non ordinato	**Array ordinato**	**Lista**	**Alberi RB**	**~~Impl. ideale~~ Hash table**
insert()	$O(1), O(n)$	$O(n)$	$O(1), O(n)$	$O(\log n)$	$O(1)$
lookup()	$O(n)$	$O(\log n)$	$O(n)$	$O(\log n)$	$O(1)$
remove()	$O(n)$	$O(n)$	$O(n)$	$O(\log n)$	$O(1)$
foreach	$O(n)$	$O(n)$	$O(n)$	$O(n)$	~~$O(n)$~~$O(m)$

Per comprendere il funzionamento delle tabelle hash dobbiamo prendere in considerazione il concetto di **insieme universo *U***, ovvero un insieme di tutte le possibili chiavi, la cui grandezza dell'insieme varia arbitrariamente in base ai dati che si vogliono contenere.

Idea di base

Quello che si vuole fare è memorizzare tutti i dati dell'insieme ***U*** in un vettore di dimensione (m) finita $T[0...m-1]$, ed avere un meccanismo per cui, data una chiave, si trovi rapidamente la posizione in cui è memorizzata.

Le chiavi possono essere delle stringhe, degli oggetti o dei numeri, e il compito delle tabelle hash è quello di trasformarli in un indice all'interno delle tabelle hash. Per fare ciò viengono utilizzate le **funzioni hash**.

Cos'è una funzione hash?

Quello che si vuole fare è memorizzare tutti i dati dell'insieme ***U*** in un vettore di dimensione (m) finita $T[0...m-1]$, ed avere un meccanismo per cui, data una chiave, si trovi rapidamente la posizione in cui è memorizzata.