CHIAVE = [illegible] "MA" h(CHIAVE) = 6	CHIAVE = [illegible] "CO" h(CHIAVE) = 0	CHIAVE = [illegible] "GRA" h(CHIAVE) = 5	CHIAVE = [illegible] "PE" h(CHIAVE) = 0	CHIAVE = [illegible] "LE" h(CHIAVE) = 5	CHIAVE = [illegible] "RE" h(CHIAVE) = 2
CHIAVE = [illegible] "CA" h(CHIAVE) = 6	CHIAVE = [illegible] "GIU" h(CHIAVE) = 3				

D1.1

D1.2.1 Le due principali buone proprietà di una funzione di hash sono: uniformità della distribuzione e unicità del risultato.

D1.2.2 Come si può notare anche dall'esempio del D1.1 la funzione di hash h gode di uniformità, infatti l'operatore modulo di 7 permette di ottenere risultati che rispettano l'uniformità semplice per cui ogni chiave ha la stessa possibilità di essere associata ad ogni bucket. Se per esempio avessimo avuto mod 5 la funzione avrebbe restituito risultati non uniformi poiché uguali a numeri da 0 a 4 lasciando vuoti gli ultimi due bucket.

D1.2.3 Per soddisfare l'unicità dobbiamo avvicinarci il più possibile all'iniettività e avere meno collisioni possibili. Sempre per l'uniformità semplice la funzione soddisfa l'unicità nei limiti della tabella poiché se avessimo un numero molto elevato di bucket e di conseguenza modulo di un numero elevato, molto difficilmente due input diversi potrebbero dare lo stesso risultato.

D1.3.1 Data la chiave di un elemento, l'algoritmo di inserimento deve prima di tutto verificare se la chiave è duplicata, che significa che è già stata inserita nella struttura. In caso di duplicazione si può semplicemente non agire oppure segnalare il fallimento dell'inserimento, in ogni caso la struttura non viene modificata.
Se la chiave è "nuova" allora l'algoritmo calcola la funzione di hash a lei applicata (matematicamente h(k) con h funzione e k chiave) ed il risultato sarà l'indice a cui accedere per l'inserimento; trovato il bucket da associare all'elemento scorriamo la lista ad esso associata fino al fondo e inseriamo l'elemento.

D1.3.2 L'inserimento ha complessità $\Theta(n)$ nel caso peggiore per cui tutti gli n elementi sono associati allo stesso indice e quindi per inserirne uno nuovo dobbiamo scorrere una lista di n elementi quando controlliamo se la chiave è duplicata.

D1.3.3 L'inserimento ha complessità $\Theta(n/m)$ quando la distribuzione degli elementi è uniforme e quindi la lunghezza della lista da scorrere per l'inserimento è lunga n/m.

D1.3.4 L'inserimento ha complessità $\Theta(1)$ nel caso migliore, ossia quando l'elemento è duplicato ed è il primo della lista associata al bucket oppure tale lista è vuota e quindi l'inserimento è diretto.

In tutti i casi consideriamo che la funzione di hash abbia complessità costante per semplicità.