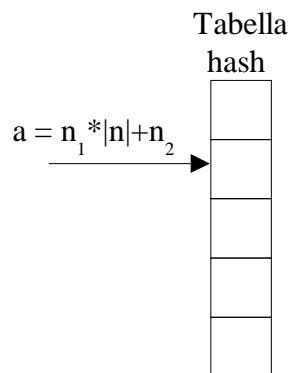


## **APPENDICE G : STRUTTURE DATI**

### **Hash Table**

Per la memorizzare dei dati utilizzati (i disagi e gli N) serve una struttura dinamica che consenta di poter accedere e modificare rapidamente ogni elemento della struttura. La struttura da noi utilizzata è la hash table come descritta in [Sedgewick]: questa consente, con una opportunamente inizializzazione dei parametri (dimensione e fattore di carico), di ricercare un elemento con uno, due accessi alla tabella. Nel nostro caso la chiave è rappresentata dal valore ID dell'arco, otteniamo così il valore cercato.

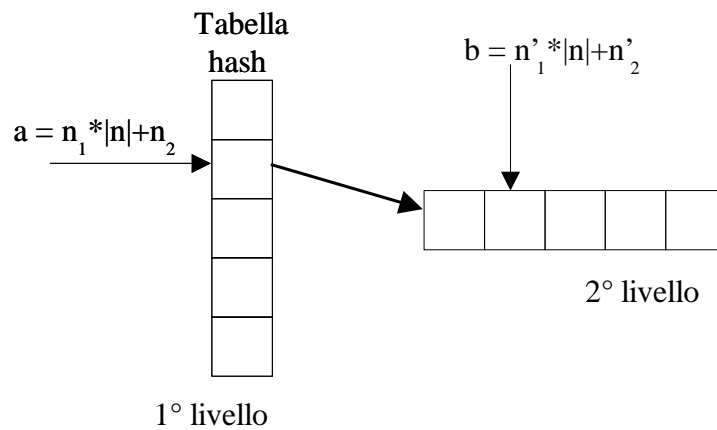


La hash table è una struttura dati statica, abbiamo perciò dovuto renderla dinamica aggiungendo il ridimensionamento nel caso in cui il fattore di carico superi la soglia del 75%. Questa operazione è molto costosa dato che richiede il reinserimento di tutti gli elementi nella nuova tabella. Per contro il numero di collisioni rimane pressochè costante, anche perchè la tabella ha sempre una dimensioni pari ad un numero primo.

### **Hash Table a due livelli**

Per memorizzare i valori di dipendenza e i disagi dei cammini comuni ( $r_{ij}$ ,  $F_{cij} + DC_{ij}$ ) servirebbe una matrice quadrata, ma questa comporta una occupazione quadratica di memoria, in generale troppo onerosa, inoltre sarebbe sparsa: la maggior parte dei dati è infatti zero. Per ovviare a questo inconveniente abbiamo deciso di utilizzare una hash table dove ogni elemento è a sua volta una tabella hash: chiamiamo questa struttura hash table a due livelli. Ogni ricerca richiede due accessi ai due livelli delle hash table.

Noi utilizziamo due archi come chiavi d'accesso: il primo livello è accessibile utilizzando il primo arco (prima chiave), il valore ricavato è un puntatore ad una seconda tabella hash accessibile attraverso il secondo arco (seconda chiave) ottenendo così il valore cercato.



### Heap

Per la scelta dell'arco da chiudere, nell'algoritmo di scelta degli archi da chiudere, abbiamo bisogno di una struttura dinamica che contenga una sequenza di record completamente ordinata con la possibilità di aggiungere un nuovo elemento e togliere quello con chiave maggiore. La struttura appena descritta è un Heap descritta in [3 ppg 149].