Hashing

Terminologia

- **Hashing**: Il processo con cui si mappa una chiave ad una posizione in una tabella.
- **Hash function** la funzione che da una chiave determina una posizione. E' indicata con **h**.
- **Hash table** e' un array che contiene i record. E' indicato con ***T***.
- La hash table ha *M* **slot**, che vanno da 0 a *M-1*.

Obiettivi

- L'hashing e' appropriato per insiemi che non contengono elementi duplicati.
- Tuttavia, i possibili valori che una chiave può assumere sono normalmente molto maggiori degli slot disponibili in una tabella.
- Data una funzione **h** e due chiavi $\mathbf{k_1}$ e $\mathbf{k_2}$, se $\mathbf{h(k_1) = h(k_2)}$ allora si ha una **collisione**.
- **Perfect Hashing**: una funzione di hash che non genera mai collisioni. Molto costosa da realizzare

La gestione delle collisioni

- Durante un inserimento, quando si verifica una collisione, si cerca uno slot libero in una posizione differente.
- Lo slot libero deve essere calcolato con un algoritmo riutilizzabile durante la fase di ricerca.

Probe Function

- Con una “Probe Function” si genera un offset
- Si controlla lo slot presente alla posiz. iniz + offset
- Se lo slot e’ occupato si ripete
  - Per questo la funzione che genera l’offset si chiama “probe function”
- Offset = P(K, i)
  - K: chiave di cui deve essere calcolato lo hash
  - i: n° di tentativo. Serve per differenziare l’offset prodotto ad ogni tentativo
- Data una chiave K, l’insieme degli offset{P(K,0), P(K,1), … P(K,n)} si chiama Probe Function di K

Esempi di Probe Functions

- Linear probing: P(K, i) = i
  - Problema: gli slot tendono a formare degli agglomerati che crescono nel tempo (primary clustering) → Prima di trovare uno slot libero devono essere effettuati molti tentativi.
- Quadratic probing: $P(K, i) = i^2$
  - Risolve il problema del "primary clustering", ma se l'hash di due chiavi diverse da come risultato lo stesso slot, le due chiavi avranno sempre la stessa probe sequence (secondary clustering)
- Double hashing: P(K, i) = i * H2(K)
  - Rendendo il risultato funzione di K, si risolve il problema del secondary clustering

Problema

Data una tabella hash di lunghezza m=13, si supponga di dover inserire (in ordine) le chiavi: 2, 8, 31, 20, 19, 18, 53, 27. Si utilizzi un double hashing con H1 e H2 definite come segue:

H1(k) = k mod 13

H2(k) = (Rev(k+1) mod 11)

Rev(k) = … inverte le cifre decimali di k, per esempio Rev(37) = 73, Rev(7) = 7

Calcolo di h(k)

H1(2)=2 mod 13=2

H1(8)=8 mod 13=8

H1(31)=31 mod 13= 5

H1(20)=20 mod 13= 7

H1(19)=19 mod 13= 6

H1(18)=18 mod 13= 5

H1(53)=53 mod 13= 1

H1(27)=27 mod 13=1

0	1	2	3	4	5	6	7	8	9	10	11	12
	53	***2***	***<u>27</u>***		***31***	***19***	***20***	***8***			***<u>18</u>***	

H1(18)=5 // lo slot 5 è occupato.

P(18, 1) = 1 * H2(18) = 3 // lo slot 8 e' occupato

P(18, 2) = 2 * H2(18) = 6 // slot 11 aggiudicato

H1(27)=1 // lo slot 1 è occupato

P(27, 1) = 1 * H2(27) = 5 // lo slot 6 e' occupato

P(27, 2) = 2* H2(27) =10 // lo slot 11 e' occupato

P(27, 3) = 3* H2(27) =15, (16 mod 13) = 3 // OK

h(2)=2
h(8)=8
h(31)=5
h(20)=7
h(19)=6
h(18)=5
h(53)=1
h(27)=1