TABELLE HASH

Angelo Di Iorio
Università di Bologna

Esercizio 1

- Implementare una classe Java per gestire una tabella Hash in cui sia le chiavi che i valori sono stringhe.
- Metodi (dizionario):
  - `HashTable(Integer capacity)`
  - `String lookup(String key)`
  - `void insert(String key, String value)`
  - `void remove(String key)`

- Aggiungere un metodo `print()` utility per stampare il contenuto della tabella

  - **Hashing: modular hashing (divisione, regola di Horner)**
  - **Collisioni: memorizzazione esterna (liste di trabocco)**

Hashing modulare – regola di Horner

- Un polinomio di grado q può essere valutato con q prodotti e q addizioni
- Operazione di modulo ad ogni iterazione per ridurre 'overflow'

$$p(x) = a_q x^q + a_{q-1} x^{q-1} + \cdots + a_1 x + a_0 = x(\cdots(x((a_q x) + a_{q-1})\cdots) + a_1) + a_0$$

```
integer H(ITEM[] k, integer ℓ)
    integer b ← ord(k[1])
    for j ← 2 to ℓ do
        b ← ((b · 64) + ord(k[j])) mod m
    return b
```

lunghezza chiave

base (in pratica scelto anche per migliorare distribuzione chiavi e performance)

Esercizio 2

- Modificare la classe precedente (refactoring) per usare diverse funzioni hash
  - iniziale della stringa (non è una buona funzione hash!)
  - estrazione di k bit
  - divisione
  - …

Esercizio 3

- Implementare una classe Java per gestire una tabella Hash in cui sia le chiavi che i valori sono stringhe.
- Metodi (dizionario):
  - `HashTable(Integer capacity)`
  - `String lookup(String key)`
  - `void insert(String key, String value)`
  - `void remove(String key)`

- Aggiungere un metodo `print()` utility per stampare il contenuto della tabella

  - **Hashing: modular hashing (divisione, regola di Horner)**
  - **Collisioni: memorizzazione interna, scansione lineare a passo unitario**

Esercizio 4

- Modificare la classe precedente per usare un metodo di scansione quadratica (con `h=1`), piuttosto che lineare
- (le altre specifiche restano invariate)

Scansione lineare: $H(k,i) = (H(k) + h \cdot i) \bmod m$

Scansione quadratica: $H(k,i) = (H(k) + h \cdot i^2) \bmod m$

Esercizio 5

- Modificare la classe precedente (refactoring) per usare un metodo di scansione con hashing doppio (con `h=1`)

Hashing doppio: $H(k,i) = (H(k) + i \cdot H'(k)) \bmod m$

- Nota: H'(k) deve essere primo rispetto a *m*