Corso di Laurea in INFORMATICA

Algoritmi e Strutture Dati

MODULO 9

Dizionari

Specifiche, rappresentazione e confronto tra realizzazioni alternative

Questi lucidi sono stati preparati da Oronzo Altamura, Maria Francesca Costabile e Floriana Esposito, Università degli Studi di Bari, per uso didattico. Essi contengono materiale originale di proprietà dell'Università degli Studi di Bari e/o figure di proprietà di altri autori, società e organizzazioni di cui e' riportato il riferimento. Tutto o parte del materiale può essere fotocopiato per uso personale o didattico ma non può essere distribuito per uso commerciale. Qualunque altro uso richiede una specifica autorizzazione da parte dell'Università degli Studi di Bari e degli altri autori coinvolti.

ESISTONO DELLE APPLICAZIONI CHE, PUR RICHIEDENDO UNA STRUTTURA DATI DEL TIPO INSIEME, NON RICHIEDONO TUTTE LE OPERAZIONI DEFINITE SULL'INSIEME.

AD ESEMPIO, NEL TRATTAMENTO DEI TESTI SONO PRESENTI SPESSO DEI DIZIONARI DELLA LINGUA MESSI A DISPOSIZIONE PER CONTROLLARE LA CORRETTEZZA ORTOGRAFICA DELLO SCRITTO. E' CERTAMENTE UN PROBLEMA PER IL TIPO ASTRATTO DI DATI INSIEME, MA NON SEMBRANO NECESSARIE OPERAZIONI COME UNIONE, INTERSEZIONE E DIFFERENZA.

QUESTI SOTTOTIPI DEL TIPO INSIEME SONO I

DIZIONARI

GLI ELEMENTI SONO GENERALMENTE TIPI STRUTTURATI AI QUALI SI ACCEDE PER MEZZO DI UN RIFERIMENTO A UN CAMPO CHIAVE. GLI ELEMENTI ASSUMONO LA FORMA DI UNA COPPIA COSITUITA DA UNA CHIAVE - ATTRIBUTO.

LA CARATTERISTICA DELLA CHIAVE E' LEGATA ALLA APPLICAZIONE: NEI DIZIONARI DEI WORD PROCESSOR LA CHIAVE INDIVIDUA UNA PAROLA MENTRE IN UN DIZIONARIO DI MAGAZZINO LA CHIAVE PUO' ESSERE UN CODICE PEZZO.

L'ATTRIBUTO ASSOCIATO, INVECE, RAPPRESENTA L'INFORMAZIONE ASSOCIATA PER SCOPI DI GESTIONE O MANUTENZIONE.

ESEMPIO:

NELLE TABELLE DI SIMBOLI, UTILI AI COMPILATORI E AI LINKER, LA CHIAVE VIENE USATA PER L'IDENTIFICATORE MENTRE NELL'ATTRIBUTO VENGONO MEMORIZZATE LE INFORMAZIONI DI GESTIONE (INDIRIZZO DELLA LOCAZIONE, DIMENSIONE, POSIZIONI NELLE QUALI L'IDENTIFICATORE E' USATO NEL PROGRAMMA ETC.)

LE OPERAZIONI APPLICATE AD UN DIZIONARIO DEVONO CONSENTIRE LA VERIFICA DELL'ESISTENZA DI UNA DEFINITA CHIAVE E DEVE ESSERE POSSIBILE L'INSERIMENTO DI NUOVE COPPIE CHIAVE-ATTRIBUTO COME PURE LA CANCELLAZIONE.

PUO' ESSERE UTILE ANCHE IL RECUPERO DELLE INFORMAZIONI PRESENTI NELL'ATTRIBUTO OPPURE LA LORO EVENTUALE MODIFICA.

POICHE' POSSIAMO DEFINIRLI COME UN CASO PARTICOLARE DI INSIEME PER I DIZIONARI

LA SPECIFICA E' IDENTICA A QUELLA DEL TIPO DI DATO INSIEME. LE OPERAZIONI AMMESSE SONO:

CREA APPARTIENE INSERISCI CANCELLA

IN ALCUNI CASI TROVIAMO ANCHE OPERAZIONI COME

RECUPERA AGGIORNA

OLTRE ALLE REALIZZAZIONI VISTE PER L'INSIEME, CHE SI RIFANNO ALLA RAPPRESENTAZIONE CON VETTORE BOOLEANO (VETTORE CARATTERISTICO) E ALLA RAPPRESENTAZIONE MEDIANTE UNA LISTA (I CUI ELEMENTI SONO QUELLI DELL'INSIEME), CI SONO REALIZZAZIONI PIU' EFFICIENTI MEDIANTE VETTORI ORDINATI E TABELLE HASH.

RAPPRESENTAZIONE CON VETTORE ORDINATO

SI UTILIZZA UN VETTORE CON UN CURSORE ALL'ULTIMA POSIZIONE OCCUPATA.

AVENDO DEFINITO UNA RELAZIONE DI ORDINAMENTO TOTALE "£" SULLE CHIAVI, QUESTE SI MEMORIZZANO NEL VETTORE IN POSIZIONI CONTIGUE E IN ORDINE CRESCENTE A PARTIRE DALLA PRIMA POSIZIONE. PER VERIFICARE L'APPARTENENZA DI UN ELEMENTO O CHIAVE K, SI UTILIZZA LA RICERCA BINARIA (DICOTOMICA, LOGARITMICA), SI CONFRONTA CIOE' IL VALORE DA RICERCARE K CON IL VALORE V CHE OCCUPA LA POSIZIONE CENTRALE DEL VETTORE E SI STABILISCE IN QUALE META' CONTINUARE LA RICERCA.

UNA POSSIBILE REALIZZAZIONE IN PASCAL

```
TYPE dizionario = record
        elementi: array [1..maxlung] of tipoelem;
        ultimo: 0..maxlung;
 end;
Function APPARTIENE (k: tipoelem; var D: dizionario): boolean;
        begin
                APPARTIENE := RICBIN(D.elementi, k, 1, D.ultimo);
        end;
Function RICBIN (var V: vettore; k: tipoelem; i,j: integer): boolean;
        var m: integer;
        begin
                if i>j then
                        RICBIN := false
                else begin
                        m := (i + j) div 2;
                        if k = V[m] then
                                RICBIN := true
                        else
                                if k < V[m] then
                        RICBIN := RICBIN (V, k, i, m-1)
                        else
                        RICBIN := RICBIN (V, k, m+1, j)
                    end
        end;
```

ESEMPIO: APPLICANDO RICBIN ALL'INSIEME DI 8 ELEMENTI INTERI (3, 5, 8, 11, 18, 23, 25, 30), CON K = 25 SI OTTIENE:

ITERAZIONE	i	j	m	V(m)
1	1	8	4	11
2	5	8	6	23
3	7	8	7	25

IL NUMERO DI CONFRONTI C(n) TRA LA CHIAVE K E LE CHIAVI PRESENTI NEL VETTORE E' DI ORDINE LOGARITMICO (O(log n)). ASSUMENDO PER SEMPLICITA' $n = 2^p$ E INDICANDO CON c UNA COSTANTE POSITIVA, SI HA:

$C(n) \leq C(n/2) + c$

$\leq C(n/2^2) + 2c$

$\leq \ldots$

$\leq C(n/2^p) + pc$

$= C(1) + pc.$

C(1) E' COSTANTE, p = log n, QUINDI C(n) E' O(log n). DUNQUE,REALIZZANDO IL DIZIONARIO CON UN VETTORE ORDINATO E USANDO LA RICERCA BINARIA, IL NUMERO DI CONFRONTI SI ABBASSA DA O(n) A O(log n). SE SI TIENE CONTO DEI VALORI DELLE CHIAVI E DELLA LORO DISTRIBUZIONE DI PROBABILITA' IL NUMERO DI CONFRONTI SI ABBASSA ULTERIORMENTE.

ESISTE UNA TECNICA DENOMINATA "HASH", CHE SI APPOGGIA SU DI UNA STRUTTURA DI DATI TABELLARE, CHE SI PRESTA AD ESSERE USATA PER REALIZZARE DIZIONARI. UTILIZZANDO QUESTA STRUTTURA, LE OPERAZIONI DI RICERCA E MODIFICA DI UN DIZIONARIO POSSONO OPERARE IN TEMPI COSTANTI E INDIPENDENTI SIA DALLA DIMENSIONE DEL DIZIONARIO CHE DALL'INSIEME DEI VALORI CHE VERRANNO GESTITI.

RAPPRESENTAZIONE CON TABELLA HASH

IDEA BASE: RICAVARE LA POSIZIONE CHE LA CHIAVE OCCUPA IN UN VETTORE DAL VALORE DELLA CHIAVE.

VANTAGGIO: OGNI OPERAZIONE RICHIEDE TEMPO O(1) NEL CASO MEDIO (O(n) NEL CASO PESSIMO).

ESISTONO DIVERSE VARIANTI CHE SI POSSONO FAR RISALIRE AD UNA FORMA *STATICA* E AD UNA FORMA *DINAMICA O ESTENSIBILE*. LA PRIMA FA USO DI STRUTTURE O TABELLE DI DIMENSIONE PREFISSATA, MENTRE L'HASH DINAMICO MODIFICA DINAMICAMENTE LE DIMENSIONI DELLA TABELLA HASH SULLA BASE DEL NUMERO DI ELEMENTI VIA VIA INSERITI O ELIMINATI.

NEL SEGUITO TRATTEREMO SOLO HASH STATICO

RAPPRESENTAZIONE CON TABELLA HASH

L'HASH (*STATICO*) PUO' ASSUMERE A SUA VOLTA DUE FORME DIVERSE DENOMINATE RISPETTIVAMENTE

- **HASH CHIUSO**: CONSENTE DI INSERIRE UN INSIEME LIMITATO DI VALORI IN UNO SPAZIO DI DIMENSIONE FISSA
- **HASH APERTO**: CONSENTE DI MEMORIZZARE UN INSIEME DI VALORI DI DIMENSIONE QUALSIASI IN UNO SPAZIO POTENZIALMENTE ILLIMITATO

AMBEDUE QUESTE VARIANTI PERO' UTILIZZANO UNA SOTTOSTANTE TABELLA HASH A *DIMENSIONE FISSA* COSTITUITA DA UNA STRUTTURA ALLOCATA SEQUENZIALMENTE IN MEMORIA E CHE ASSUME LA FORMA DI UN ARRAY.

RAPPRESENTAZIONE CON TABELLA HASH

- NEL CASO DI **HASH CHIUSO** LA STRUTTURA SARA' COMPOSTA DA UN CERTO NUMERO (**MAXBUCKET**) DI CONTENITORI DI UGUALE DIMENSIONE DENOMINATI **BUCKET**. OGNUNO DI QUESTI CONTENITORI PUO' MANTENERE AL PROPRIO INTERNO AL MASSIMO UN NUMERO $b \geq 1$ DI ELEMENTI CHE COMPRENDERANNO LA CHIAVE E IL CORRISPONDENTE ATTRIBUTO (NEL CASO b=1 OGNI BUCKET AVRA' UNA SOLA COPPIA CHIAVE-ATTRIBUTO).
- NEL CASO DI **HASH APERTO** LA STRUTTURA SARA' COMPOSTA DA UN CERTO NUMERO INDETERMINATO DI CONTENITORI **BUCKET**.

IN AMBEDUE I CASI VIENE USATA UNA FUNZIONE ARITMETICA ALLO SCOPO DI CALCOLARE, PARTENDO DALLA CHIAVE, LA POSIZIONE IN TABELLA DELLE INFORMAZIONI CONTENUTE NELL'ATTRIBUTO COLLEGATO ALLA CHIAVE.

RAPPRESENTAZIONE CON TABELLA HASH

E' UNA ALTERNATIVA EFFICACE ALL'INDIRIZZAMENTO DIRETTO IN UN VETTORE PERCHE' LA DIMENSIONE E' PROPORZIONALE AL NUMERO DI CHIAVI ATTESE.

K L'INSIEME DI TUTTE LE POSSIBILI CHIAVI DISTINTE

V IL VETTORE DI DIMENSIONE **m** IN CUI SI MEMORIZZA IL DIZIONARIO

LA SOLUZIONE IDEALE E' LA FUNZIONE DI ACCESSO

H: K → {1,...,m}

CHE PERMETTA DI RICAVARE LA POSIZIONE H(k) DELLA CHIAVE k NEL VETTORE V COSI' CHE, SE $k_1 \in K$ E $k_2 \in K$, $k_1 \neq k_2$ SI HA $H(k_1) \neq H(k_2)$.

UTILIZZANDO m =½K½ SI HA GARANZIA DI BIUNIVOCITA' E DI POTER ACCEDERE DIRETTAMENTE ALLA POSIZIONE CONTENENTE LA CHIAVE. SE ½K½ E' GRANDE, SI HA SPRECO ENORME DI MEMORIA. LA DIMENSIONE m DEL VETTORE VA SCELTA IN BASE AL NUMERO DI CHIAVI ATTESE.

LA SOLUZIONE DI COMPROMESSO E' SCEGLIERE UN m MAGGIORE DI 1 MA MOLTO MINORE DI ½K½.

HASH CHIUSO

ESEMPIO: SIA K = { *Cognomi di musicisti* } E SI ASSUMA m= 26. UNA POSSIBILE FUNZIONE E' H(K)=h, $1 \leq h \leq 26$, SE IL CARATTERE INIZIALE DI K E' L'h-ESIMA LETTERA DELL'ALFABETO INGLESE. H NON E' BIUNIVOCA.

SEQUENZA DI ARRIVO	INDIRIZZI HASH
ALBINONI	H(ALBINONI) = 1
OFFENBACH	H(OFFENBACH) = 15
PALESTRINA	H(PALESTRINA) = 16
PROKOFEV	H(PROKOFEV) = 16
PUCCINI	H(PUCCINI) = 16
ROSSINI	H(ROSSINI) = 18

CI SONO COLLISIONI PER I COGNOMI CHE INIZIANO CON LA STESSA LETTERA

MEMORIZZAZIONE	
ALBINONI	1
:	
OFFENBACH	15
PALESTRINA	16
PROKOFEV	17
PUCCINI	18
ROSSINI	19
.	
	26

UNA **COLLISIONE** SI VERIFICA QUANDO CHIAVI DIVERSE PRODUCONO LO STESSO RISULTATO DELLA FUNZIONE. ESISTONO **FUNZIONI HASH** PIU' O MENO BUONE ANCHE SE LE COLLISIONI NON SI POTRANNO MAI EVITARE DEL TUTTO.

NELL'ESEMPIO VISTO SI E' ADOTTATA UNA SEMPLICE STRATEGIA PER LA RISOLUZIONE DELLE COLLISIONI (***SCANSIONE LINEARE***) :

SE H(k) PER QUALUNQUE CHIAVE k INDICA UNA POSIZIONE GIA' OCCUPATA, SI ISPEZIONA LA POSIZIONE SUCCESSIVA NEL VETTORE. SE LA POSIZIONE E' PIENA SI PROVA CON LA SEGUENTE E COSI' VIA FINO A TROVARE UNA POSIZIONE LIBERA O TROVARE CHE LA TABELLA E' COMPLETAMENTE PIENA.

UNA POSIZIONE LIBERA PUO' VENIRE FACILMENTE SEGNALATA IN FASE DI REALIZZAZIONE DA UNA CHIAVE FITTIZIA *LIBERO*.

PER LA CANCELLAZIONE E' PIU' SEMPLICE SOSTITUIRE L'OGGETTO CANCELLATO CON UNA CHIAVE FITTIZIA *CANCELLATO* CHE DOVREBBE ESSERE FACILMENTE DISTINGUIBILE DALLLE ALTRE CHIAVI REALI E DALL'ALTRA CHIAVE FITTIZIA *LIBERO*.

LA STRATEGIA LINEARE PUO' PRODURRE NEL TEMPO IL CASUALE ADDENSAMENTO DI INFORMAZIONI IN CERTI TRATTI DELLA TABELLA (**AGGLOMERATI**), PIUTTOSTO CHE UNA LORO DISPERSIONE.

HASH CHIUSO (UN ESEMPIO DI REALIZZAZIONE) /1

```
CONST
        MAX  = {intero massimo indice della tabella};
        LIBERO= {qualunque valore di chiave per indicare il vuoto}
        CANCELLATO= {qualunque valore di chiave per indicare
        elementi eliminati}
TYPE
        TIPOCHIAVE = {qualcosa su cui è definibile HASH};
        ATOMO= RECORD
                CHIAVE: TIPOCHIAVE;
                ATTRIBUTO: {campi dati di interesse}
            END;
        BUCKET= 0..MAX;
        DIZIONARIO = ARRAY [BUCKET] OF ATOMO;
VAR
        T: DIZIONARIO;
```


HASH CHIUSO (UN ESEMPIO DI REALIZZAZIONE) /2

```
FUNCTION HASH(K:TIPOCHIAVE):BUCKET;
        BEGIN
                HASH:= {qualsiasi definizione}
        END
-------------------------------------------------------------------------------
PROCEDURE CREA (VAR T:DIZIONARIO);
        VAR
                INDICE : BUCKET;
        BEGIN
                FOR INDICE:= 0 TO MAX DO
                T[INDICE].CHIAVE:= LIBERO;
        END;
```


HASH CHIUSO (UN ESEMPIO DI REALIZZAZIONE) /3

```
PROCEDURE APPARTIENE (K:TIPOCHIAVE; T:DIZIONARIO;
                VAR INDICE:BUCKET; VAR TROVATO, PIENO: BOOLEAN);
VAR
  PROVA, POSTO : BUCKET;
  BEGIN
    PROVA:=HASH(K);
    INDICE:=PROVA;
    IF T[INDICE].CHIAVE:=K THEN
      TROVATO:=TRUE
    ELSE BEGIN
      PIENO:=TRUE;
      REPEAT
        INDICE:=(INDICE+1)MOD(MAX+1);
        IF PIENO AND ((T[INDICE].CHIAVE=CANCELLATO)OR
                                (T[INDICE].CHIAVE=LIBERO))
            THEN BEGIN
                PIENO:=FALSE;
                POSTO:=INDICE
            END
      UNTIL ((T[INDICE].CHIAVE=CANCELLATO) OR (T[INDICE].CHIAVE=LIBERO))
                                        OR (INDICE=PROVA))
```


HASH CHIUSO (UN ESEMPIO DI REALIZZAZIONE) /4

```
          IF (T[INDICE].CHIAVE=K THEN
              TROVATO:=TRUE
          ELSE BEGIN
              TROVATO:=FALSE;
              INDICE:=POSTO
          END
        END
  END;
-------------------------------------------------------------------------------
PROCEDURE INSERISCI (A:ATOMO; VAR T:DIZIONARIO);
  VAR   INDICE: BUCKET;
        TROVATO, PIENO: BOOLEAN;
  BEGIN
        APPARTIENE (A.CHIAVE, T, INDICE,TROVATO, PIENO);
        IF NOT TROVATO THEN
                IF PIENO THEN
                        WRITELN ('TABELLA PIENA')
                ELSE
                        T[INDICE]:=A
        ELSE
                WRITELN ('ATTENZIONE CHIAVE DUPLICATA')
  END;
```


HASH CHIUSO (UN ESEMPIO DI REALIZZAZIONE) /5

```
PROCEDURE CANCELLA (K:TIPOCHIAVE; VAR T:DIZIONARIO);
  VAR
        INDICE: BUCKET;
        TROVATO, PIENO: BOOLEAN;
  BEGIN
        APPARTIENE (K, T, INDICE,TROVATO, PIENO);
        IF TROVATO THEN
                T[INDICE].CHIAVE:=CANCELLATO
        ELSE
                WRITELN ('ATTENZIONE CHIAVE NON ESISTENTE')
  END;
```

QUALE CHE SIA LA FUNZIONE HASH ADOTTATA, DEVE ESSERE PREVISTA UNA STRATEGIA PER GESTIRE IL PROBLEMA DEGLI **AGGLOMERATI** E DELLE **COLLISIONI**. IN DEFINITIVA:

- OCCORRE UNA ***FUNZIONE HASH***, CALCOLABILE VELOCEMENTE E CHE DISTRIBUISCA LE CHIAVI UNIFORMEMENTE IN V, IN MODO DA RIDURRE LE COLLISIONI;
- OCCORRE UN METODO DI **SCANSIONE** PER LA SOLUZIONE DELLE COLLISIONI UTILE A REPERIRE CHIAVI CHE HANNO TROVATO LA POSIZIONE OCCUPATA E CHE NON PROVOCHI LA FORMAZIONE DI AGGLOMERATI DI CHIAVI;
- LA DIMENSIONE **m** DEL VETTORE **V** DEVE ESSERE UNA SOVRASTIMA DEL NUMERO DELLE CHIAVI ATTESE, PER EVITARE DI RIEMPIRE **V** COMPLETAMENTE.

*PER DEFINIRE FUNZIONI HASH, E' CONVENIENTE CONSIDERARE LA RAPPRESENTAZIONE BINARIA **bin(k)** DELLA CHIAVE **k**. NEL CASO DI CHIAVI NON NUMERICHE SI PUO' CONSIDERARE LA RAPPRESENTAZIONE BINARIA DEI CARATTERI E **bin(K)** E' DATA DALLA LORO CONCATENAZIONE. (IN PASCAL LA FUNZIONE PRIMITIVA **ord**, APPLICATA A UN CARATTERE, NE RESTITUISCE IL CODICE INTERO)*

4 BUONI METODI DI GENERAZIONE HASH

DENOTIAMO CON **int(b)** IL NUMERO INTERO RAPPRESENTATO DA UNA STRINGA BINARIA b.
INDICHIAMO DA **0** A **m-1** GLI ELEMENTI DI **V**.

a) **H(k)=int(b)**, DOVE **b** E' UN SOTTOINSIEME DI **p** BIT DI **bin(k)**, SOLITAMENTE ESTRATTI NELLE POSIZIONI CENTRALI

b) **H(k)=int(b)**, DOVE **b** E' DATO DALLA SOMMA MODULO 2, EFFETTUATA BIT A BIT, DI DIVERSI SOTTOINSIEMI DI **p** BIT DI **bin(k)**.

c) **H(k)=int(b)**, DOVE **b** E' UN SOTTOINSIEME DI p BIT ESTRATTI DALLE POSIZIONI CENTRALI DI **bin(int(bin(K))²)**.

d) **H(k)** E' UGUALE AL RESTO DELLA DIVISIONE DI **int(bin(K)) /m** (m E' DISPARI; SE FOSSE UGUALE A 2^p, DUE NUMERI CON GLI STESSI **p** BIT FINALI DAREBBERO SEMPRE LUOGO A UNA COLLISIONE).

L'ULTIMA FUNZIONE HASH DEFINITA E' LA MIGLIORE DAL PUNTO DI VISTA PROBABILISTICO E FORNISCE UN'ECCELLENTE DISTRIBUZIONE DEGLI INDIRIZZI H(k) NELL'INTERVALLO [0, m-1].

20

ESEMPIO

(FUNZIONE HASH): SI SUPPONGA CHE LE CHIAVI SIANO DI 6 CARATTERI ALFANUMERICI (CHIAVI PIU' LUNGHE SONO TRONCATE, MENTRE CHIAVI PIU' CORTE SONO ESPANSE A DESTRA CON SPAZI). SI ASSUMA ord(A)=1, ord(B)=2, … , ord(Z)=26 E ord(b)=32, DOVE b INDICA LO SPAZIO, CON RAPPRESENTAZIONE DI OGNI ORDINALE SU 6 BIT. PER LE CHIAVI WEBER E WEBERN SI OTTIENE:

bin(WEBER b) = 010111 000101 000010 000101 010010 100000
bin(WEBERN) = 010111 000101 000010 000101 010010 001110

DOVE SI SONO EVIDENZIATI PER CHIAREZZA I GRUPPI DI 6 BIT CHE RAPPRESENTANO CIASCUN CARATTERE.

CALCOLIAMO GLI INDIRIZZI HASH OTTENUTI CON LE FUNZIONI DEFINITE IN (a), (b) E (d):

(a) SIA m = 256 = 2^8. ESTRAENDO GLI 8 BIT DALLA POSIZIONE 15 ALLA 22 DI bin(k), H(WEBER) = H(WEBERN) = int(00100001) = 33. LE DUE CHIAVI DANNO PERTANTO LUOGO AD UNA COLLISIONE

(b) SIA ANCORA m = 256 = 2^8 E SI CALCOLI b RAGGRUPPANDO bin(k) IN CINQUE GRUPPI DI 8 BIT (DOPO AVER ESPANSO A DESTRA bin(k) CON QUATTRO ZERI)

bin(WEBER b) = 010111 000101 000010 000101 010010 100000 **0000**
bin(WEBERN) = 010111 000101 000010 000101 010010 001110 **0000**

H(WEBER) = int(11000011) = 195, POICHE'

11000011 = 01011100 Å 01010000 Å 10000101 Å 01001010 Å 00000000

DOVE Å INDICA LA SOMMA BIT A BIT MODULO 2.
ANALOGAMENTE, H(WEBERN) = int(00100001) = 33, POICHE'

00100001 = 01011100 Å 01010000 Å 10000101 Å 01001000 Å 11100000

BENCHE' H(WEBERN) SIA UGUALE A 33 COME NEL CASO (a), LA COLLISIONE E' STATA ELIMINATA.

(d) SIA m = 383. UTILIZZANDO 6 BIT, int(bin(k)) E' UN NUMERO IN BASE 2^6 = 64.

$Int(bin(WEBERN)) = 23 * 64^5 + 5 * 64^4 + 2 * 64^3 + 5 * 64^2 + 18 * 64^1 + 14 * 64^0 =$
$64(64(64)64(23*64)+5)+2)+5)+18)+14$

H(WEBERN) E' OTTENUTO PRENDENDO IL RESTO DELLA DIVISIONE Int(bin(WEBERN))/383

22

HASH APERTO

UNA TECNICA CHE EVITA LA FORMAZIONE DI AGGLOMERATI E QUELLA DELL'HASH APERTO CHE RICHIEDE CHE LA TABELLA HASH MANTENGA LA LISTA DEGLI ELEMENTI LE CUI CHIAVI PRODUCONO LO STESSO VALORE DI FUNZIONE (TRASFORMATA).

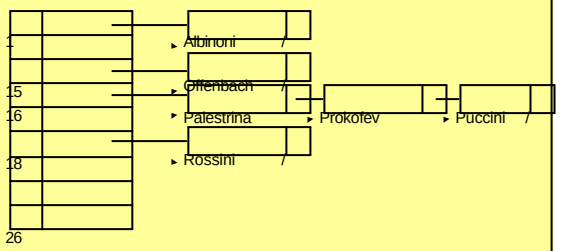
LA TABELLA HASH VIENE REALIZZATA DEFINENDO UN ARRAY DI **LISTE DI BUCKET (LISTE DI TRABOCCO)**.

LA FUNZIONE HASH VIENE UTILIZZATA PER DETERMINARE QUALE LISTA POTREBBE CONTENERE L'ELEMENTO CHE POSSIEDE UNA DETERMINATA CHIAVE IN MODO DA POTER ATTIVARE UNA SUCCESSIVA OPERAZIONE DI RICERCA NELLA LISTA CORRISPONDENTE E A RESTITUIRE LA POSIZIONE DEL BUCKET CHE CONTIENE LA CHIAVE.

23

LISTE DI TRABOCCO:

V CONTIENE, IN OGNI POSIZIONE, UN PUNTATORE AD UNA LISTA



APPARTIENE, INSERISCI E CANCELLA SI RIDUCONO A RICERCA INSERZIONE E CANCELLAZIONE DI LISTA

24

ESEMPIO DI REALIZZAZIONE DELL'OPERATORE CANCELLA

```
Procedure CANCELLA(k: tipochiave; var T: dizionario);
Var i: bucket; p: posizione; trovato: boolean;
Begin
        i := HASH(k);
        if T[i] <> nil then
            if T[i]^.chiave= k then
                T[i] := T[i]^.successivo
            else begin
                p := T[i]; trovato := false;
                while (p^.successivo <> nil) and (not trovato) do
                    if p^.successivo^.chiave = k then begin
                        p^.successivo := p^.successivo^.successivo;
                        trovato := true
                    end
                else
                    p := p^.successivo
            end
end;
```


METODI DI SCANSIONE:

UNA SISTEMATIZZAZIONE

NELLA REALIZZAZIONE CON HASH CHIUSO E LISTE DI TRABOCCO SI E' USATO UN METODO DI SCANSIONE ESTERNO CONTRAPPOSTO ALLA SCANSIONE LINEARE CHE E' UN METODO DI SCANSIONE INTERNO.

ALTRI METODI INTERNI (O CHIUSI) SONO

SCANSIONE QUADRATICA
SCANSIONE PSEUDOCASUALE
HASHING DOPPIO

SCANSIONE INTERNA

CHIAMIAMO f_i LA FUNZIONE CHE VIENE UTILIZZATA L'i-ESIMA VOLTA CHE SI TROVA OCCUPATA UNA POSIZIONE DEL VETTORE V, $i \geq 0$, (PER i=0, f_0 = H).

f_i VA SCELTA IN MODO DA TOCCARE TUTTE LE POSIZIONI DI V (UNA SOLA VOLTA).

SCANSIONE LINEARE:

f_i = (H(k) + h*i) MOD m

h E' UN INTERO POSITIVO PRIMO CON **m**

h RAPPRESENTA LA DISTANZA TRA DUE POSIZIONI SUCCESSIVE ESAMINATE NELLA SCANSIONE (SE **h = 1**, *SCANSIONE A PASSO UNITARIO*).

ESSENDO **h** E **m** PRIMI TRA LORO, VENGONO ESAMINATE TUTTE LE POSIZIONI DI **V** PRIMA DI RICONSIDERARE LE POSIZIONI GIA' ESAMINATE.

SVANTAGGIO: NON RIDUCE LA FORMAZIONE DI AGGLOMERATI

SCANSIONE QUADRATICA

f_i = (H(k) + h*i + i(i-1)2) MOD m

m E' PRIMO.

LA DISTANZA TRA DUE POSIZIONI SUCCESSIVE NELLA SEQUENZA E' VARIABILE, QUINDI LA POSSIBILITA' DI AGGLOMERATI E' RIDOTTA.
SVANTAGGIO: LA SEQUENZA DI SCANSIONE NON INCLUDE TUTTE LE POSIZIONI DI **V** (SVANTAGGIO TRASCURABILE PER **m** NON TROPPO PICCOLO).

SCANSIONE PSEUDOCASUALE

fi = (H(k) + p_i) MOD m

p_i E' L'i-ESIMO NUMERO GENERATO DA UN GENERATORE DI NUMERI PSEUDOCASUALI, CHE GENERA GLI INTERI TRA **1** E **m** UNA SOLA VOLTA IN UN ORDINE QUALUNQUE.

HASHING DOPPIO

f_i = (H(k) + i * F(k)) MOD m

F E' UN'ALTRA FUNZIONE HASH DIVERSA DA **H**.

PROBLEMI PER LA CANCELLAZIONE DI CHIAVI

USANDO METODI DI SCANSIONE INTERNA E POTENDO CANCELLARE CHIAVI, NON SI E' MAI SICURI CHE, RAGGIUNTA UNA POSIZIONE VUOTA NELLA RICERCA DI k, TALE CHIAVE NON SI TROVI IN UN'ALTRA POSIZIONE DI V, POICHE' LA POSIZIONE ORA VUOTA ERA OCCUPATA QUANDO k E' STATA INSERITA.

BISOGNA DUNQUE SCANDIRE ANCHE LE POSIZIONI IN CUI SI E' CANCELLATO E FERMARSI O SULLA POSIZIONE MAI RIEMPITA O DOPO ESSERE TORNATI SU UNA POSIZIONE GIA' SCANDITA.

CIO' DETERMINA UN AUMENTO DEL TEMPO DI RICERCA.

UTILIZZARE IL METODO DI SCANSIONE ESTERNA SE SONO PREVISTE MOLTE CANCELLAZIONI.

CONFRONTO DI COMPLESSITÀ (1/2)

	APPARTIENE	INSERISCI	CANCELLA
Vettore booleano	O(1)	O(1)	O(1)
Liste non ordinate	O(n)	O(1)^	O(n)
Liste ordinate	O(n)	O(n)	O(n)
Vettore ordinato	O(log n)	O(n)	O(n)
Hash (caso medio)	O(1)	O(1)	O(1)
Hash (caso pessimo)	O(max[m,n])	O(max[m,n])	O(max[m,n])

^ se si sa che l'elemento non è presente

CONFRONTO DI COMPLESSITÀ (2/2)

- Per le tabelle hash è stato dimostrato che il tempo medio di ricerca dipende dal fattore di carico della tabella hash

  $a = n/m$

  n = numero di chiavi memorizzate nella tabella
  m = dimensione della tabella

- Tra i metodi di scansione interna, quello lineare è il più lento. Anche per gli altri metodi è bene non avere

  $a > .75$

- Quando a cresce, è conveniente ristrutturare la tabella

31