

OSS Da C posso ottenere C' in una passata poiché

$$C'[i] = C[i-1] + C[i]$$

Ex:

	0	1	2
C	3	2	1
	↑	↑	↑
C'	3	5	6
	↑	↑	↑
	(0+3)	(3+2)	(5+1)

Una volta costruito C' posso procedere

Ex: Guardando A e C' precedenti, scelgo l'elemento 8 in A. Vado a guardare in $C'[8-6]$, quanti elementi sono presenti prima di 8 e $C'[2] = 5$.

Visto che ci sono 5 elementi minori uguali a 8, nell'array finito B, 8 sono in posizione $C[2]-1$, quindi 4

- Quindi una volta costruito C', posso anche inserire gli elementi in B in ordine sparso.

NOTA: Quando ho più elementi uguali, devo ovvero l'accortezza di diminuire il numero di elementi minori uguali di $C'[i]$. così da poter inserire anche gli altri elementi uguali.

Inoltre, partendo a inserire gli elementi da destra verso sinistra, questo algoritmo diventa stabile (elementi uguali da destra verso sinistra)

Quindi (Dati A e C' della pagina precedente)

B	0	1	2	3	4	5	6	7	8	9	10
	6	7	8	8	8	9	10	10	10	10	12

CODIFICA

COUNTING-SORT(A, m)

K = max(A)

m = min(A)

C = new array (K - m + 1)

For ($i = 0$ to $K - m$)

$C[i] = 0$

For ($i = 0$ to $m - 1$)

$C[A[i] - m] = C[A[i] - m] + 1$

For ($i = 1$ to $K - m$)

$C[i] = C[i] + C[i - 1]$ // C', ottenuto sommando gli elementi di C

B = new array (m)

sommare i valori di C

For ($i = m - 1$ down to 0)

$B[C[A[i] - m] - 1] = A[i]$ *

$C[A[i] - m] = C[A[i] - m] - 1$ // decremento per eventuali valori uguali

* $A[i]$ = valore

$[A[i] - m]$, è l'indice in C in cui è contenuto il m-^o elemento minore di $A[i]$

$B[C[A[i] - m] - 1]$, è l'indice in cui dovrà mettere $A[i]$,

COMPLESSITÀ

Lo spazio utilizzato è lo stesso e la complessità

è rimasta invariata perché abbiamo semplicemente aggiunto un ciclo in più di $O(K - m)$

In questo modo, l'algoritmo è lineare, usa gli stessi valori di partenza e non copie - inoltre è stabile

TEOREMA MASTER

Abbiamo visto il teorema master che permette di risolvere equazioni di ricorrenza della forma $T(m) = aT(\frac{m}{b}) + f(m)$ ottenute tramite la risoluzione con approccio D'Intre et impresa.

In quasi tutti i casi, la funzione $f(m)$ ha una forma particolare, ovvero $\Theta(m^k \cdot \log^k m)$, per questi particolari così il teorema Master assume una forma più semplice nel suo corollario.

COROLLARIO TEOREMA MASTER

Siamo $a \geq 1$, $b > 1$, $k \geq 0$, $h \geq 0$ costanti.

Sia inoltre $T(m)$ tale che:

$$T(m) = aT\left(\frac{m}{b}\right) + m^k (\log m)^h$$

$$T(m) = \begin{cases} \Theta(m^{\log_b a}) & \text{se } \log_b a > k \\ \Theta(m^k (\log m)^{h+1}) & \text{se } \log_b a = k \\ \Theta(m^k (\log m)^h) & \text{se } 0 \leq \log_b a < k \end{cases}$$

DIM

• Se $\log_b a > k$, allora,

$$m^k (\log m)^h = \Theta(m^{\log_b a - \varepsilon}) \quad \text{per qualche } \varepsilon > 0$$

$$\Rightarrow T(m) = \Theta(m^{\log_b a})$$

• Se $\log_b a = k$, allora $m^k (\log m)^h = \Theta(m^{\log_b a} (\log m)^h)$

$$\Rightarrow T(m) = \Theta(m^k (\log m)^{h+1})$$

• Se $0 \leq \log_b a < k$, allora,

$$m^k (\log m)^h = \Omega(m^{\log_b a + \varepsilon}) \quad \text{per qualche } \varepsilon > 0$$

inoltre vale la condizione di regolarità:

$$a\left(\frac{m}{b}\right)^k (\log \frac{m}{b})^h = \frac{a}{b^k} m^k (\log m - \log b)^h \leq \frac{a}{b^k} m^k (\log m)^h \leq c m^k (\log m)^h$$

$$\Rightarrow T(m) = \Theta(m^k (\log m)^h)$$

$$\text{con } \frac{a}{b^k} \leq c \ll 1 \quad \left| \begin{array}{l} \log_b a < k \Rightarrow a < b^k \\ \frac{a}{b^k} < 1 \end{array} \right.$$

ESERCIZIO D'ESAME

Si enumera il teorema Master e il suo corollario, quindi si risolva la seguente equazione di ricchezza del valore del parametro $\alpha \geq 1$

$$T(m) = \alpha \cdot T\left(\frac{m}{2}\right) + m^2 \log^2 m$$

Per quel valore α si ha: $T(m) = O(m^3)$; $T(m) = \Omega(m^2 \log^3 m)$; $T(m) = \Theta(m^2 \log^2 m)$

RISOLUZIONE

Applicando il corollario dell'equazione parametrica:

Dividiamo il problema in 3 casi, ovvero, troveremo un valore x e poi studieremo $\alpha < x$, $\alpha = x$, $\alpha > x$

Dato la natura di f , posso applicare il corollario

Per ottenere α : $\log_2 \alpha (k,=,>) k \Rightarrow \log_2 \alpha (k,=,>) 2 = \alpha (k,=,>) 4$

Quindi il valore di $\log_2 \alpha$ per $\alpha = 4$ risolve il \log)

$$T(m) = \begin{cases} \Theta(m \cdot \log_2^2) = \Theta(m) & \text{se } \log_2 \alpha > 2 \Leftrightarrow \alpha > 4 \\ \Theta(m^2 \cdot (\log_2 m)^3) & \text{se } \log_2 \alpha = 2 \Leftrightarrow \alpha = 4 \\ \Theta(m^2 \cdot (\log_2 m)^2) & \text{se } \alpha < \log_2 \alpha < 2 \Leftrightarrow 1 \leq \alpha < 4 \end{cases}$$

Una volta ottenuto α mi basta applicare le formule

RISOLUZIONE 2

CASO A: $O(m^3)$

$\alpha > 4$

Abbiamo $m \log_2 \alpha = O(m^3)$

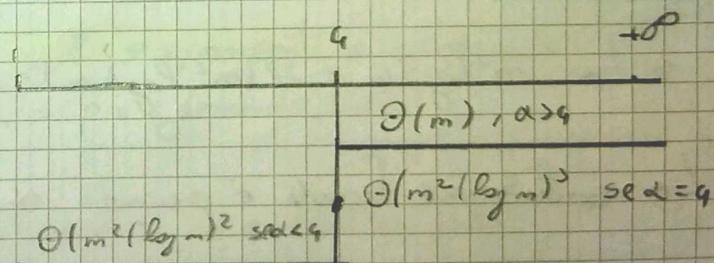
$$\Rightarrow \log_2 \alpha \leq 3$$

Il che è m deve essere < per rimanere O l' - sup.

$$\Rightarrow \alpha \leq 8$$

quindi per

$$4 < \alpha \leq 8 \Rightarrow O(m^3) = T(m)$$



$\alpha = 4$

Si ha $m^2 (\log m)^3 = O(m^3)$ / m^2 è un voto ma $> m$, quindi si per $\alpha = 4$, $T(m) = O(m^3)$

$1 \leq \alpha < 4$

Si ha $m^2 (\log m)^2 = O(m^3)$ // stesso caso, sarà sempre < per $1 \leq \alpha < 4$, $T(m) = O(m^3)$

Ora possiamo ottenere la soluzione finale unificando gli intervalli:

$$\Rightarrow 1 \leq \alpha < 8$$

CASO B $R(m^2 \log^3 m)$

$\alpha > 4$

Si ha $m^{\beta \alpha} = \Omega(m^2 \log^3 m)$ se $\beta \alpha > 2 \Rightarrow \alpha > 4$
per $\alpha > 4$, $T(m) = \Omega(m^2 \log^3 m)$

In questo caso $m^{\beta \alpha}$ dovrà essere superiore a R , quindi l'esponente dovrà essere maggiore

$\alpha = 4$

Si ha $m^2 \log^3 m = \Omega(m^2 \log^3 m)$, vero!
per $\alpha = 4$, $T(m) = \Omega(m^2 \log^3 m)$

$1 \leq \alpha < 4$

Si ha $m^2 \log^2 m = \Omega(m^2 \log^3 m)$ MAI

La soluzione finale è quindi $\alpha \geq 4$

CASO C: $\Omega(m^2 \log^4 m)$

$\alpha > 4$

si ha $m^{\log_2 \alpha} = \Omega(m^2 \log^4 m)$ per $\log_2 \alpha > 2 \Rightarrow \alpha > 4$
per $\alpha > 4$ si ha $f(m) = \Omega(m^2 \log^4 m)$

$\alpha = 4$

si ha $m^2 \log^3 m = \Omega(m^2 \log^4 m)$ NO

$1 \leq \alpha \leq 4$

si ha $m^2 \log^2 m = \Omega(m^2 \log^4 m)$ NO

La soluzione finale è $\alpha > 4$

COMMENTI ESERCIZI

1. Risolvendo la \sim comeza è semplicemente contando il teorema e il corollario, basta solo trovare i 3 casi di α , ricordando che $\log_2 (\leq, =, \geq) k$ sono i 3 casi.
2. Una volta trovata la soluzione per casi della \sim comeza vero: fatta una domanda di confronto con complessità totale.

Basta trovare gli intervalli in cui è valida l'uguaglianza per ogni caso e poi intersecare l'intervalle per trovare la soluzione finale.

Tuttavia è necessario confrontare gli esponenti e verificare se un valore è più piccolo (0), o più grande (1) asintoticamente.

Può succedere anche che il confronto sia falso per un determinato valore di α .

METODO DI AKRA-BAZZI (GENERALIZZAZIONE)

È una generalizzazione in cui abbiamo più di un termine in t, possiamo avere un numero qualsiasi k.

$$\text{Sia } T(m) = g(m) + \sum_{i=1}^k \alpha_i T(b_i m + h_i(m)), \text{ per } m \geq m_0$$

dove:

- $\alpha_i > 0$, $0 < b_i < 1$ costanti, $i = 1, 2, \dots, k$

- $|g(m)| = \Theta(m^c)$

- $|h_i(m)| = O(m/\log^m)$ $i = 1, 2, \dots, k$

Allora

$$T(m) = \begin{cases} \Theta(m^p) & \text{se } c < p \\ \Theta(m^p \log^p m) & \text{se } c = p \\ \Theta(m^c) & \text{se } c > p \end{cases}$$

Sia inoltre p tale che $\sum_{i=1}^k \alpha_i b_i^{p_i} = 1$

p sono quindi soluzione di questa equazione

EX:

$$T(m) = m^{\frac{7}{4}} + \frac{7}{4} T(\lfloor \frac{1}{2}m \rfloor) + T(\lceil \frac{3}{4}m \rceil) \quad (m \geq 3)$$

$$\Rightarrow \frac{7}{4} \cdot \left(\frac{1}{2}\right)^x + \left(\frac{3}{4}\right)^x = 1 \quad \Rightarrow x=2 \quad (\text{fornito})$$

L'equazione non ha metodo risolutorio generale, bisogna approssimare

VERIFICHIAMO $x=2$

$$\frac{7}{4} \cdot \frac{1}{4} + \frac{9}{16} = \frac{7+9}{16} = \frac{16}{16} = 1 \Rightarrow x=2 \text{ è la soluzione}$$

Dunque abbiamo

$$x=p=2, c=2 \Rightarrow c=p \Rightarrow T(m)=\Theta(m^c \log m)$$

EX: 2

$$T(m) = T(\lceil \frac{m}{5} \rceil) + T\left(\frac{2m}{10} + c\right) + \Theta(m)$$

$$\left(\frac{1}{5}\right)^x + \left(\frac{2}{10}\right)^x = 1 \quad \stackrel{x=1}{\Rightarrow} \quad \frac{c+2}{10} = \frac{9}{10} < 1$$

$$\Rightarrow x=p < 1, \quad c=1 \Rightarrow c > p \Rightarrow \Theta(m^c) \Rightarrow T(m) = \Theta(m)$$

CAMBIAMENTO DI VARIABILI

Avendo: $T(m) = 2T(\lfloor \sqrt{m} \rfloor) + \log m$

$$\Rightarrow T(m) = 2T(\lfloor m^{1/2} \rfloor) + \log m$$

poniamo $m = 2^m$

$$\Rightarrow T(2^m) = 2T(2^{m/2}) + \overbrace{\log 2^m}^{m}$$

introduciamo $S(m) = T(2^m)$

$$\Rightarrow S(m) = 2S\left(\frac{m}{2}\right) + m \quad // qui possiamo applicare il t. mosten$$

$$\Rightarrow S(m) = \Theta(m \log m) \quad // adesso dobbiamo tornare a T da $S$$$

abbiamo che $m = 2^m$, quindi $m = \log m$

$$\Rightarrow T(2^{\log m}) = S(\log m) \quad // 2^{\log_2 2^m} = m^{\log_2 2} = m^1 = m$$

$$\Rightarrow S(\log m) = \Theta(\log m \cdot \log(\log m))$$

ma sappiamo che $T(2^{\log m}) = T(m)$

$$\Rightarrow T(m) = S(\log m) = \Theta(\log m \cdot \log(\log m))$$

$$\Rightarrow T(m) = \Theta(\log m \log(\log m))$$

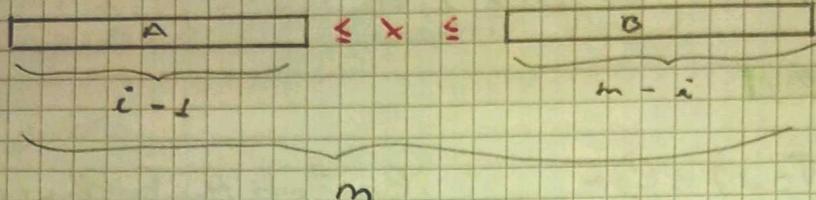
Il cambiamento di variabile ci permette quindi di ricordare
a una forma risolvibile tramite il teorema mosten

il metodo M. Follisce ma
cambiando variabile possiamo
riuscire

MEDIANE E STATISTICHE D'ORDINE

Dato un insieme di m elementi, l' i -esima statistica d'ordine è l' i -esimo elemento più piccolo.

Cioè, un elemento x dell'insieme tale che i rimanenti $(m-1)$ elementi possano esserne così suddivisi:



In altre parole, l'iesima statistica d'ordine è quell'elemento che, se noi ordiniamo gli m elementi in senso crescente, occupa la casella i .

Quando i occupa (cinca) la posizione $\frac{m}{2}$ parleremo di **mediana**
EX:

Potremo parlare di mediana perfetta solo se m è pari,
poiché, ad esempio, con $m=7 \rightarrow$ mediana = 4, poiché $|A|=|B|=3$

• Quando m è dispari, la mediana perfetta è $i=\lceil \frac{m+1}{2} \rceil$

• Quando m è pari avremo
mediana inferiore $i=\lfloor \frac{m+1}{2} \rfloor$
mediana superiore $i=\lceil \frac{m+1}{2} \rceil$

• Quando $i=1$, l'iesima statistica d'ordine si dice **minima**

• Quando $i=m$, l'emesima statistica d'ordine si dice **massima**

PROBLEMA DELLA SELEZIONE

Il problema della selezione, consiste in:

- Dati un insieme A di m elementi (distinti).
- un intero $i \leq m$ (rango)

Trovare l' i -esima statistica d'ordine di A , ovvero l'elemento di A che ha rango i .

ALGORITMO BANALE

- Si ordini A
- si restituisca l'elemento $A[i]$

La complessità di questo algoritmo deriva dal minimo per algoritmi basati su confronti, ovvero, $O(m \log m)$, esiste però un algoritmo lineare.

ALCUNI CASI PARTICOLARI (MIN E MAX)

CODIFICA

MINIMUM(A)

$mim = A[0]$

For $i = 0$ to $\text{length}(A)$

if ($mim > A[i]$)

$mim = A[i]$

return mim

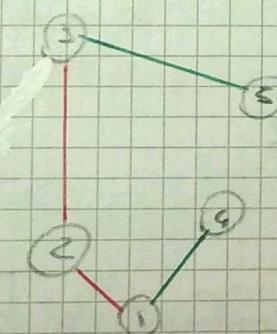
confronti eseguiti = $m-1$

DIM

L'algoritmo è ottimale perché $m-1$ confronti sono necessari.

Supponiamo di usare un grafo e creare un arco per ogni confronto eseguito

3	5	2	1	6



confronti pensi

confronti vint.

confrontaemo sempre
visti più piccoli

Alla fine mostriamo che il grafo è **connesso** poiché dobbiamo confrontare tutti gli elementi.

Per teorema supponiamo che un grafo connesso di m nodi ha $m-1$ archi. Abbiamo quindi dimostrato che con $m-1$ confronti l'algoritmo è ottimale analogamente per il max.

CALCOLARE MIN E MAX DI N ELEMENTI

1^a SOLUZIONE

- calcolare minimo di m elementi
- calcolare il massimo di $m-1$ elementi

$$\# \text{confronti} = (m-1) + (m-2) = \boxed{2m-3}$$

* poiché escludo
già il max

2^a SOLUZIONE (più efficiente in termine di confronti)

- Sia A un insieme di m numeri
- si suddividono gli m elementi in $\frac{m}{2}$ coppie, più un eventuale elemento sparato.
- Si confrontano gli elementi di ciascuna coppia mettendo i vincenti in un insieme B e i perdenti in un insieme C

• Se c'è un elemento sparato lo si aggiunga sia a B che a C

• Avremo che: $\max(B) = \max(A)$
 $\min(C) = \min(A)$

Quanti confronti avremo fatto?

$$\bullet \underbrace{\frac{m}{2}}_{\text{per}} + \underbrace{\left(\frac{m}{2}-1\right)}_{\max B} + \underbrace{\left(\frac{m}{2}-1\right)}_{\min C} = \frac{3m}{2} - 2 \quad \text{se } m \text{ pari}$$

che ne
B e C

$$\bullet \underbrace{\frac{m-1}{2}}_{\text{B e C}} + \underbrace{\left(\frac{m-1}{2}\right)}_{\max B} + \underbrace{\left(\frac{m-1}{2}\right)}_{\min C} = \frac{3m}{2} - \frac{3}{2} \quad \text{se } m \text{ dispari}$$

- el.
sparso

$$\Rightarrow \lceil \frac{3m}{2} \rceil - 2$$

In definitiva, il numero di confronti è

$$\boxed{\lceil \frac{3m}{2} \rceil - 2}$$

Si dimostra che questi confronti sono necessari quindi l'algoritmo è ottimale

- Questo è il miglior algoritmo che consente di trovare contemporaneamente max e min

SELEZIONE IN TEMPO ATTESO LINEARE SENZA GARANZIA

Guardiamo il **Randomized Select**, un algoritmo di selezione che si ispira al quicksort.

SPIEGAZIONE

Premettiamo un pivot randomico x ed effettuiamo una partizione di A dove a sinistra di x ci saranno valori minori e a destra maggiori, ovviamente NON ordinati.

- A questo punto, noi stiamo cercando l' i -esimo elemento, e sia r l'indice di x nella partizione.

- Se $i < r$, basterà cercare l' i -esimo elemento nella partizione $\leq x$.

- Se $i = r$, abbiamo trovato l'elemento.

- Se $i > r$, basterà cercare nella partizione $> x$.

Se sono presenti chiavi duplicate si inseriscono tutte al centro con x .

COMPLESSITÀ

Si osserva che una singola posso essere fatta in tempo lineare m , ma possono esserci, nel caso peggiore, fino a m posso, con m numero di elementi, ovvero quindi $\Rightarrow O(m \cdot m) = O(m^2)$

Nel caso medio invece, quando molte suddivisioni sono regolari, si dimostra probabilisticamente che la complessità è $O(m)$.

Pero non c'è garanzia di linearità.

CODIFICA

RANDOMIZED-SELECT(A, p, r, i)

if ($p == r$) return $A[p]$;

$q = \text{RANDOMIZED-PARTITION}(A, p, r);$

$K = q - p + 1;$

if ($i == K$) return $A[q]$;

else if ($i < K$)

return RANDOMIZED-SELECT($A, p, q-i, i$);

else

return RANDOMIZED-SELECT($A, q+1, r, i-K$);

SELEZIONE IN TEMPO LINEARE

$\text{SELECT}(A, i)$

- Si divide A in $\lceil \frac{n}{5} \rceil$ gruppi di 5 elementi, più un eventuale gruppo con i restanti $n \bmod 5$ elementi.

- si determinano le mediane degli $\lceil \frac{n}{5} \rceil$ gruppi e sia M la sequenza di tali mediane

- si effettui la chiamata ricorsiva $m = \text{SELECT}(M, \lceil \frac{1+i}{c} \rceil)$ (calcoliamo la mediana dell'index delle mediane)

- Partizioniamo A in A_1, A_2, A_3 , dove:

- $A_1 = [x \in A : x < m]$

- $A_2 = [x \in A : x = m]$

- $A_3 = [x \in A : x > m]$

- L'obiettivo è dunque cercare l' i -esimo elemento, quindi, se:

if $|A_1| \geq i \rightarrow \text{SELECT}(A_1, i)$ chiamata ricorsiva su A_1 ,

else if $|A_1| + |A_2| \geq i > |A_1| \rightarrow$ ritorno m , poiché in A_2 ho solo m e soprattutto non c'è i .

else $|A_1| + |A_2| < i \leq |A_1| + |A_2| \rightarrow \text{SELECT}(A_3, i - (|A_1| + |A_2|))$

SPIEGAZIONE GRAFICA

$$\textcircled{1} A = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

$$\textcircled{2} n=5 \text{ et}$$

$$0 \quad 0 \quad 0$$

$$0 \quad 0 \quad 0 \quad 0$$

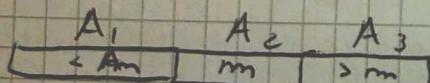
(3)

$$\textcircled{4} [0 \quad 0 \quad 0 \quad 0] \Rightarrow M[0 \quad \overset{m}{\bullet} \quad 0 \quad 0]$$

$$0 \quad 0 \quad 0$$

$$0 \quad 0 \quad 0$$

4
=>



COMPLESSITÀ

Notiamo che l'algoritmo effettua o 1 o al massimo 2 chiamate ricorsive, ovvero, quella del calcolo di m e, o quella se $n < m$ o se $n \geq m$

Ci aspettiamo quindi la seguente relazione

$$T(m) = T\left(\frac{m}{b_1}\right) + T\left(\frac{m}{b_2}\right) + \Theta(m)$$

→ calcolo mediante

Si nota che questa equazione è risolvibile tramite AKRA-BAZZI.

PARTITION Si può effettuare in tempo lineare

STABILITÀ

	0	1	2	3	4	5	6	7	8	9
A	2 ₁	5 ₁	3 ₁	2 ₂	3 ₂	4 ₁	3 ₃	4 ₂	3 ₄	5 ₂
A'	2 ₁	2 ₂	3 ₁	3 ₂	3 ₃	3 ₄	4 ₁	4 ₂	5 ₁	5 ₂

L'output di un algoritmo di ordinamento stabile non è una qualsiasi permutazione ordinata ma l'unica in cui gli elementi uguali mantengono lo stesso ordine tra di loro.

RADIX-SORT

L'idea del radix sort è quella di ordinare in base alle cifre dei numeri in input, ordinando prima per unità, poi per decine, fino al massimo grado. Il radix-sort per funzionare ha bisogno di un altro algoritmo di ordinamento stabile.

ESEMPIO

A	121	212	322	311	213	323
---	-----	-----	-----	-----	-----	-----

Per una maggiore comprensione poniamo i valori in colonna

A: 1 2 1

2 1 2

3 2 2

3 1 1

2 1 3

3 2 3

1° ordiniamo tutti i valori considerando solo la colonna delle unità con un algoritmo stabile.

①
②

Ora poniamo

1 2 1

3 1 1

2 1 2

3 2 2

2 1 3

3 2 3

2. Una volta ordinati i valori per la prima colonna, oppure chiamato l'algoritmo sulla seconda colonna.

Qui entra in gioco la stabilità, ordinando la seconda colonna cambiando la posizione dei numeri, ma le colonne già ordinate con valori uguali, mantengono l'ordine, in questo modo non vengono falsati i valori.

3	1	1
2	1	2
2	1	3
1	2	1
3	2	2
3	2	3

3. A questo punto ordinando la colonna delle cifre più significative avremo concluso l'ordinamento.
- Notiamo che grazie alla stabilità, i valori con cifre uguali hanno mantenuto la stessa posizione.

Conclusione

1	2	1
2	1	2
2	1	3
3	1	1
3	2	2
3	2	3

4. Dopo aver ordinato tutte le colonne otteniamo l'array ordinato

$$A' = \boxed{121 \ 212 \ 213 \ 311 \ 322 \ 323}$$

COMPLESSITÀ

Sia K il numero di cifre che compongono il numero. In caso di numeri con cifre diverse prendiamo K massimo e aggiungeremo degli 0 a sinistra di padding dove necessario.

Sia c il numero possibile di cifre e b la base in cui stiamo operando, avremo che

$$0 \leq c \leq b$$

$$K \leq b$$

Utilizzando il counting sort, che ha complessità $O(m+K)$ dove K è il massimo valore che una cifra può assumere nell'array da ordinare, noteremo che ragionando in base 10, K potrà assumere cifre che vanno da 0 a 9, il che abbassa la complessità del counting sort drasticamente.

Notiamo che, il counting sort va applicato K volte poiché essendo K il numero di cifre massimo, avremo K colonne quindi:

$$T(m) = K \cdot O(m) = O(Km) = O(m)$$

Essendo K costante si ignora

Possiamo applicare lo stesso ragionamento ma esprimendo i nostri elementi in base 2
Consideriamo che prima di confrontare l' i -esima colonna dell' i -esima iterazione, dovremmo estrarre l' i -esima cifra

E^x $b = 10$

$$i=2$$

$$c = \underline{2} \underline{4} \underline{3} \underline{6}$$

Se volessi estrarre la cifra i di c espresso in base b dovrà fare

$$\lfloor c/b^i \rfloor \% b^i$$

ovvero: $\lfloor \frac{2436}{10^2} \rfloor \% 10 = 4$

Considerando il numero in base 2 anziché base 10, potremo semplificare i calcoli, realizzandoli in un solo ciclo di clock

$$c \gg 1 \parallel 1$$

- c si divide per 2, il floor è automatico
 - $\parallel 1$, fa un OR con una subnet che prende solo il LSB
- Ovviamente il numero in binario avrà più cifre ma complessifremo abbassando notevolmente la complessità dei calcoli.

TABELLE HASH

Le tabelle hash vengono usati per la creazione di dizionari e sono la struttura dati più utilizzata in vari ambiti.

CONFRONTO DI COMPLESSITÀ

	INS	CANC	SEARCH
ARRAY NON ORD.	$O(1)$	$O(1)$	$O(m)$
ARRAY ORDINATO	$O(m)$	$O(m)$	$O(\lg m)$
BST	$O(m)$	$O(m)$	$O(m)$
BST BILANCIAZIO	$O(\lg m)$	$O(\lg m)$	$O(\lg m)$
HEAP	$O(\lg m)$	$O(\lg m)$	$O(\lg m)$
MASH	$O(1)$	$O(1)$	$O(m) / O(1)$

Le uniche operazioni possibili su hash sono INSERIMENTO, CANCELLAZIONE e RICERCA e hanno tutte complessità costante

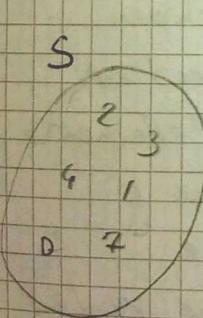
Quando implementiamo una tabella hash è importante tenere a mente che tutte le sue chiavi sono presenti in un insieme S che è sotto insieme dell'insieme universo U di tutte le chiavi.

TABELLA AD INDIRIZZAMENTO DIRETTO

È un particolare hash in cui la presenza di una chiave viene segnalata con un booleano in corrispondenza dell'indice con stesso valore della chiave.

Dato T tabella indirizzata e S insieme delle chiavi avremo che

$$T[i] = \text{true se } i \in S$$



T	0
0	1
1	2
1	3
1	4
0	5
0	6
1	7

Per vedere se un elemento è presente nella tabella bastano cercare il valore booleano al corrispondente indice.

PROCEDURE TABELLA AD INDIRIZZAMENTO DIRETTO

• INSERT (T, i)

$$T[i] = i;$$

• CANCEL (T, i)

$$T[i] = 0$$

• SEARCH (T, i)

$$\text{return } (T[i] == 1)$$

la f. hash, data una chiave k e T , restituisce la posizione della tabella in cui l'elemento con chiave k viene memorizzato.

Nelle tabelle a indirizzamento diretto ogni elemento occupa un solo bit, il problema è che le chiavi S potrebbero avere valori molto grandi e quindi la tabella potrebbe raggiungere dimensioni ingestibili.

Per risolvere il problema della dimensione, fissando a priori la dimensione di T , stimando se possibile la quantità di chiavi, per fare questo uso le tabelle hash.

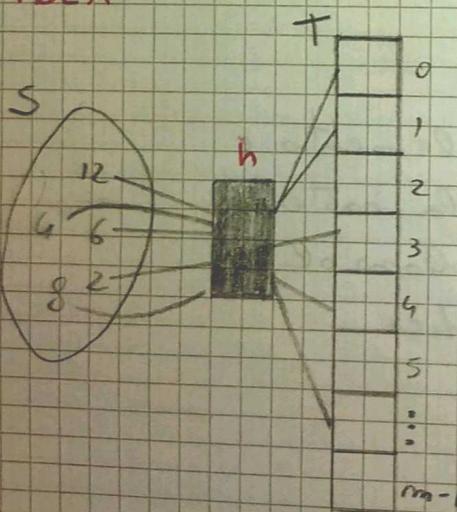
HASH TABLE BASE

L'idea è quella di limitare lo spazio ma garantire lo stesso le proprietà.

La prima operazione da fare è impostare un limite sulla dimensione di T , quindi

$$|T| = m$$

IDEA



L'idea è quella di far passare ogni elemento dell'insieme delle chiavi tramite una **black box** che decide a quale indice di T associare l'elemento di S , qui memorizziamo il vero e proprio valore (non un booleano) e l'associazione non avrà nulla a che vedere con l'indice.

La black box h non è altro che la funzione di hash

$$h: U \rightarrow \{0, 1, \dots, m-1\} \quad m < |U|$$

PROCEDURE HASH TABLE BASE

• $\text{INSERT}(T, K)$ $K \in U$
 $T[h(K)] = K;$

• $\text{CANC}(T, K)$
 $T[h(K)] = \text{NULL}$

$\text{SEARCH}(T, K)$
Return ($T[h(K)] \neq \text{NULL}$)

OSS Cosa succede se $|S| > m$? quindi se T non può ospitare tutte le chiavi?

Supponiamo di avere $K_1, K_2 \in S : h(K_1) = h(K_2)$, questo evento si chiama **collisione di chiavi**.

La collisione da essere qualsiasi sovrascrittura o cancellazione non volute.

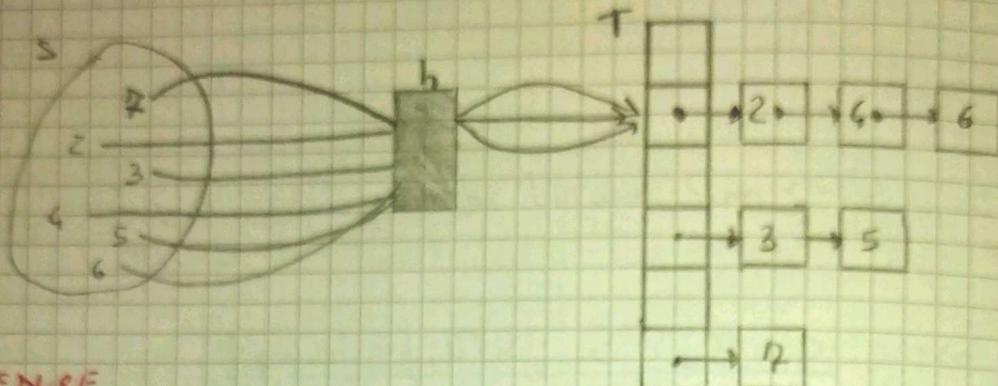
Il problema delle collisioni è l'unico delle tabella hash, in quanto la condizione perfetta sarebbe conoscere a priori $|S|$ e creare una **Tabella hash perfetta**.

Per risolvere il problema delle collisioni esistono 2 metodi:

- CONCATENAZIONE
- INDIRIZZAMENTO APERTO

CONCATENAZIONE

Questo metodo utilizza l'elenco concatenato, per evitare collisioni, ogni volta che più di un $h(K)$ è uguale, ammiche inserire K in $[h(K)]$, concatenando gli elementi a partire da quella posizione. Ogni volta che un nuovo $h(K)$ risulta uguale, si inserisce in testa alla lista.



PROCEDURE

- $\text{INSERT}(T, K)$

$\text{INSERT } K \text{ nella lista } [h(K)]$ $\text{ // } O(1)$, in testa

- $\text{CANC}(T, K)$

$\text{CANC } K \text{ nella lista } [h(K)]$ $\text{ // } O(1)$

- $\text{RICERCA}(T, K)$

$\text{RICERCA } K \text{ nella lista } [h(K)]$ $\text{ // } O(m)$

COMPLESSITÀ

tutto dipende dalla lista.

Nella ricerca, nel caso pessimo, tutti gli m elementi potrebbero essere in una sola lista. $\Rightarrow O(m)$
Il caso medio viene definito da

- $d = \text{Fattore di carico}$, ovvero quanti elementi sono presenti nella tabella

$$d = \frac{m}{m} / d = 0, \text{ Tabella leggera}$$

$$/ d = 1, \text{ Tabella piena}, (m = m \Rightarrow m/m = 1)$$

$d > 1$, il numero di elementi è maggiore di $|T|$, avremo quindi sicuramente una collisione.

I POTESI DI HASHING UNIFORME

Ogni chiave deve avere la stessa probabilità di vedersi assegnata una qualsiasi posizione ammmissibile, indipendentemente da altri valori hash già assegnati.

$$P_n \{ h(K) = i \} = \frac{1}{m} \quad 0 \leq i < m$$

Se la funzione hash rispetta questa proprietà: $\alpha = \frac{n}{m}$

Quindi:

$$k \xrightarrow{\underbrace{h(k)}_{\alpha}} \underbrace{|T[h(k)]|}_{\alpha} = \frac{m}{m} = \alpha \Rightarrow O(1 + \alpha)$$

La complessità dipende quindi da α , ovvero $\frac{n}{m}$. avendo quindi abbastanza spazio m . α si mantiene più piccolo, la complessità può diventare costante.

Il requisito di hashing uniforme è difficile da verificare perché raramente è nota la funzione di distribuzione di probabilità. Nella pratica è possibile usare delle approssimazioni per realizzare funzioni hash con buone prestazioni

- METODO DELLA DIVISIONE
- METODO DELLA MOLTIPLICAZIONE

METODO DELLA DIVISIONE

Consiste nell'associazione alla chiave K il valore hash

$$h(K) = K \bmod m$$

È un metodo semplice e veloce ma occorre evitare conti: se $m = 2^p$, $h(K)$ rappresenta solo i p bit meno significativi di K . Questo limita la casualità di h , in quanto è funzione di una porzione della chiave.

Se $m = 2^p$, $h(K)$ rappresenta solo i p bit meno significativi di K . Questo limita la casualità di h , in quanto è funzione di una porzione della chiave.

EX: $K_1: 1000 \ 1010$

$K_2: 1011 \ \underbrace{1010}_p \Rightarrow h(K)$ uguale per 2 valori K .

Bisogna rendere la funzione h dipendente da tutti i bit della chiave; una buona scelta per m è un numero primo non troppo vicino alla potenza di 2

METODO DI MOLTIPLICAZIONE

Consiste nell'associazione alla chiave K il valore hash

$$h(K) = L_m [KA \bmod 1]$$

• $[KA \bmod 1]$ è la parte frazionaria di KA , ovvero $KA - \lfloor KA \rfloor$

Ha il vantaggio che il valore di m non è critico, di solito si sceglie $m = 2^n$.

A è una frazione uguale a $\frac{s}{2^n}$

Dove s è un intero $0 < s < 2^n$

Supponiamo che w sia il numero di bit di una parola da $K < w$

$$\Rightarrow 0 < A = \frac{s}{2^n} < 1$$

ANALISI DI COMPLESSITÀ ALGORITMO SELECTION
Questo algoritmo effettua, una chiamata

ANALISI DI ECONOMIA QUANTITATIVA

Calcoliamo quindi i limiti superiori di $|A_1|$ e $|A_2|$

LIMITE SUPERIORE DI A_3

Cerchiamo di capire quanti elementi sono $\leq m$

$$\begin{bmatrix} \frac{m}{3} \\ 2 \end{bmatrix} = \begin{bmatrix} m \\ 12 \end{bmatrix}$$

- Graficamente osserviamo che gli elementi \leq_m sono per quanto riguarda la base:
 L'altezza sono 3, ma in una quintupla potrebbero mancare elementi, quindi:

$$3 \left\lceil \frac{13}{5} \right\rceil - 2$$

Abbiamo dunque

$$|A_1| + |A_2| = 3 \lceil \frac{m}{10} \rceil - 2$$

$$\Rightarrow |A_3| = m - (|A_1| + |A_2|) \leq m - 3 \lceil \frac{m}{10} \rceil + 2 \leq m - 3 \frac{m}{10} + 2 = \frac{7m}{10} + 2$$

Dunque, per cercare il limite superiore di $|A_3|$, abbiamo cercato il limite inferiore di $|A_1| + |A_2|$

LIMITE SUPERIORE DI $|A_1|$

In maniera simmetrica potremo volutamente il numero di elementi $\geq m$

- Per quanto riguarda l' altezza vogliono le stesse considerazioni.
- Per la base avremo

$$|\Gamma_{\frac{m}{5}}| - |\Gamma_{\frac{m}{10}}| + 1$$

ovvero tutto $\frac{m}{5}$ meno la parte degli elementi c, e quelli uguali ovvero tutto $\frac{m}{5}$ meno la parte degli elementi c, e quelli uguali

Averemo quindi:

$$3 \left(|\Gamma_{\frac{m}{5}}| - |\Gamma_{\frac{m}{10}}| + 1 \right) - 2$$

$$\geq \frac{3m}{10} - 2$$

Dunque

$$|A_2| + |A_3| = 3 \frac{m}{10} - 2$$

$$|A_1| = m - (|A_2| + |A_3|) \leq m - 3 \frac{m}{10} + 2 = \frac{7m}{10} + 2$$

Averemo ottenuto che $|A_1|$ e $|A_3|$ comunque, possiamo dunque scrivere la ricchezza

$$T(m) \leq T\left(\left\lceil \frac{m}{5} \right\rceil\right) + T\left(\frac{7m}{10} + 2\right) + \Theta(m)$$

Notiamo subito che è risolvibile tramite AKS-Bazzi.

$$\alpha_1 = \alpha_2 = 1 > 0 ; b_1 = \frac{1}{5}, b_2 = \frac{7}{10}; h_1(m) = 0, h_2(m) = 2; g(m) = \Theta(m) = \Theta(m') = 0$$

Sia ptole che $\left(\frac{1}{5}\right)^p + \left(\frac{7}{10}\right)^p = 1$

$\left(\frac{1}{5}\right)' + \left(\frac{7}{10}\right)' = \frac{9}{10} < 1$, segue quindi, dato che $\left(\frac{1}{5}\right)^x + \left(\frac{7}{10}\right)^x$ è decrescente

$p < 1$, quindi

$$T(m) = \Theta(m)$$

DI MOSTRAZIONE TRAMITE INDUZIONE

Dimostriamo per induzione che

$$T(m) \leq cm$$

$$\begin{aligned} T(m) &\leq T\left(\lceil \frac{m}{5} \rceil\right) + T\left(\frac{2m}{10} + z\right) + \Theta(m) \\ &\leq c\lceil \frac{m}{5} \rceil + c\left(\frac{2m}{10} + z\right) + \alpha(m) \\ &< c\frac{m}{5} + c + \frac{2cm}{10} + 2c + \alpha m = \frac{9cm}{10} + 3c + \alpha m \\ &= cm + \left(-\frac{cm}{10} + 3c + \alpha m\right) \end{aligned}$$

Dobbiamo aggiustare l'ipotesi, in particolare, se $c > 40a$

$$\frac{c}{10} > 4a \Rightarrow -\frac{c}{10} < -4a \Rightarrow -\frac{c}{10} + a < -3a < 0$$

per $m_0 > 40a$

$$\begin{aligned} -\frac{cm}{10} + 3c + \alpha m &= \left(-\frac{c}{10} + a\right)m + 3c \leq \left(-\frac{c}{10} + a\right)40 + 3c \\ &= -c + 40a < 0 \end{aligned}$$

Quindi

$$T(m) \leq cm \quad \forall m \in \mathbb{N}^*$$

con $c > \max(40a, T(1), T(\frac{c}{2}), \dots, \frac{T(m_0-1)}{m_0-1})$

$$T(1) < c$$

$$T(\frac{c}{2}) < c+z \rightarrow c > \frac{T(c)}{2}$$

$$T(\frac{c}{3}) < c_3 \rightarrow c > \frac{T(c)}{3}$$

⋮

$$T(m_0-1) \leq c(m_0-1) \dots$$