RICERCA DI UN ELEMENTO

INPUT: $M[1], M[2], \dots, M[n]$, α

OUTPUT: $M[n] = 1$ se $\exists k$ t.c. $M[k] = \alpha$,
altrimenti $= 0$

- algoritmo sequenziale classico $t = n$

  __nota__: se l'input è ordinato (costo dell'ordinamento $O(n \log n)$)
  $\Rightarrow$ ricerca dicotomica $t = \log n$.

- algoritmo quantistico su input __non__ ordinato $t = \sqrt{n}$

interferenza quantistica

Algoritmi paralleli per RICERCA di α

- primo: CRCW (si usa un flag F)

```
F = 0;
for K = 1 to n par do
    if ( M[k] == α )     CR
        F = 1;   ← CW
M[n] = F;
```

Domanda: perché usiamo F?

Risposta:

....

Prestazioni:

$p = n$ $t =$ costante

• secondo : CREW

1) for $k = 1$ to n par do

$$M[k] = (\overset{CR}{M[k] == x} \ ? \ 1 : 0)$$

2) MAX-ITERATO (M[1], ... , M[n])

Prestazioni

1) $p = n$ $\quad t =$ costante $\Rightarrow$ Wyllie $p = \frac{n}{\log n}$ $\quad t = \log n$

2) $\qquad p = n/\log n \quad t = \log n$

$$p = O(n/\log n) \quad t = O(\log n)$$

$E \sim$ Costante $\neq 0$

- terzo : EREW

1) REPLICA $\alpha \rightarrow A[1], A[2], \dots, A[n]$

2) for $K = 1$ to n par do

   $M[K] = (M[K] == A[K] \; ? \; 1 : 0)$

3) MAX-ITERATO $(M[1], \dots, M[n])$

prestazioni:

2) + 3) $\quad p = n/\log n \quad t = \log n$

+ 1) $\quad p = n/\log n \quad t = \log n$

$= \quad n/\log n \quad \log n \qquad E = \text{Costante} \neq 0$

Varianti di questo codice per problemi legati

- conteggio di α in M
  il 3) diventa SOMMATORIA

- posizione max di α in M
  il 2) diventa $M[k] = K$ se c'è α

- posizione min di α in M
  come sopra modifico 2)
  il 3) diventa OP-ITERATA dove

$$OP(x,y) = \begin{cases} \dots & \dots \\ \dots & \dots \end{cases}$$

NUOVO PROBLEMA: ORDINAMENTO

Formalmente RANKING

Input: $M[1], \dots, M[n]$

Output: permutazione: $p : \{1, \dots, n\} \to \{1, \dots, n\}$ t.c.

$$M[p(1)] \leq M[p(2)] \leq \dots \leq M[p(n)]$$

dove $p(i)$ = indice dell'elemento del vettore M che va in posizione i

Osservazione:

In genere gli algoritmi di ordinamento sono basati sui CONFRONTI

Confronto: $M[j] \leq M[i]$? SI / NO

Algoritmi di ordinamento che usano i confronti:

$$t = \Omega(n \log n)$$

Θ

upper bound

lower bound

- dim. di lower bound (IDEA)

– ALBERO DI DECISIONE = ALGORITMO DI ORDINAMENTO

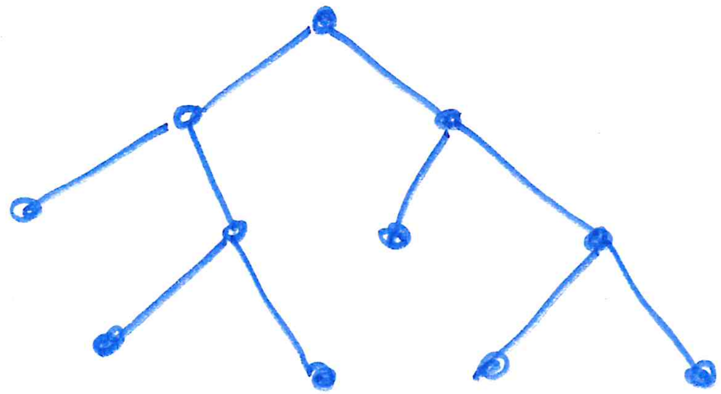
– i nodi:

$a \leq b$

si: $b \leq c$ no: $d \leq e$

NODO DI DECISIONE dove avviene un confronto

– algoritmo:



= albero di decisione

- foglie = permutazione dell'input = "p"
  Infatti una foglia individua un unico cammino a partire dalla radice e quindi i confronti che mi permettono di ordinare l'input

- altezza dell'albero = n° di confronti effettuati nel caso peggiore = Tempo dell'alg. di ordin.

OSSERVAZIONI :

$\#$ foglie $\geq n!$ = possibili ordinamenti di n elementi

t = altezza   il max $\#$ di foglie è 2^t

$$2^t \geq \# \text{foglie} \geq n! \Rightarrow t \geq \log n!$$

$$\log n! \geq \log \prod_{i=\frac{n}{2}+1}^{n} i \geq \log \left(\frac{n}{2}\right)^{\frac{n}{2}} = \frac{n}{2} \log \frac{n}{2} \sim n \log n$$

- Primo approccio parallelo
  (basato su algo di conteggio $t = \Theta(n^2)$)

<u>Assunzione</u>

n è una potenza di 2 e gli elementi sono $\neq$ tra loro

Sequenziale counting sort:

$M[i]$ va in posizione $K \iff$ k elementi sono $\leq M[i]$ in M

usiamo il vettore

$\underbrace{V[1], V[2], \dots, V[n]}_{P^{-1} \text{ - permutazione inversa di } P}$ dove $V[i]$ contiene K

Algoritmo sequenziale counting:

```
for i=1 to n
    V[i]=0                              (1)
for i=1 to n
    for j=1 to n
    if(M[j]≤ M[i]) then V[i]++          (2) → FINE!
for i=1 to n
    F[V[i]]=M[i]                        (3)
for i=1 to n
    M[i]= F[i]                          (4)
```

prestazioni : (2) è la fase + pesante $t = n^2$

versione parallela :

1) $\forall$ j, i ho un processore $P_{i,j}$ che effettua

$$M[j] \le M[i] ?$$

$$V[\,i,j\,] = (\; M[j] \le M[i] \;?\; 1 : 0 \;)$$

matrice booleana

domanda: cosa rappresenta la i-esima riga?
Risposta : individua gli elementi di M $\le$ a $M[i]$

2) $\forall$ i SOMMATORIA parallela della i-esima riga

$$\Rightarrow \quad \left.\begin{array}{l} V[1,n] \\ V[2,n] \\ \vdots \\ V[n,n] \end{array}\right\} \quad \text{vettor colonna che coincide con} \longrightarrow \quad \begin{array}{l} V[1] \\ V[2] \\ \vdots \\ V[n] \end{array}$$

di prima

Algoritmo parallelo: CODICE

- for $1 \le i, J \le n$ par do

  $V[i, J] = (M[J] \le M[i] \, ? \, 1 : 0)$

- for $i = 1$ to n par do

  $\text{SOMMATORIA}(V[i,1], V[i,2], \dots, V[i,n])$

- for $i = 1$ to n par do

  $M[V[i,n]] = M[i]$

Domanda: EREW?