ORDINAMENTO

Input: $A[1]$ $A[2]$ ... $A[n]$

P_1 P_2 P_n

Output: $cont(P_1) < cont(P_2) < \dots < cont(P_n)$

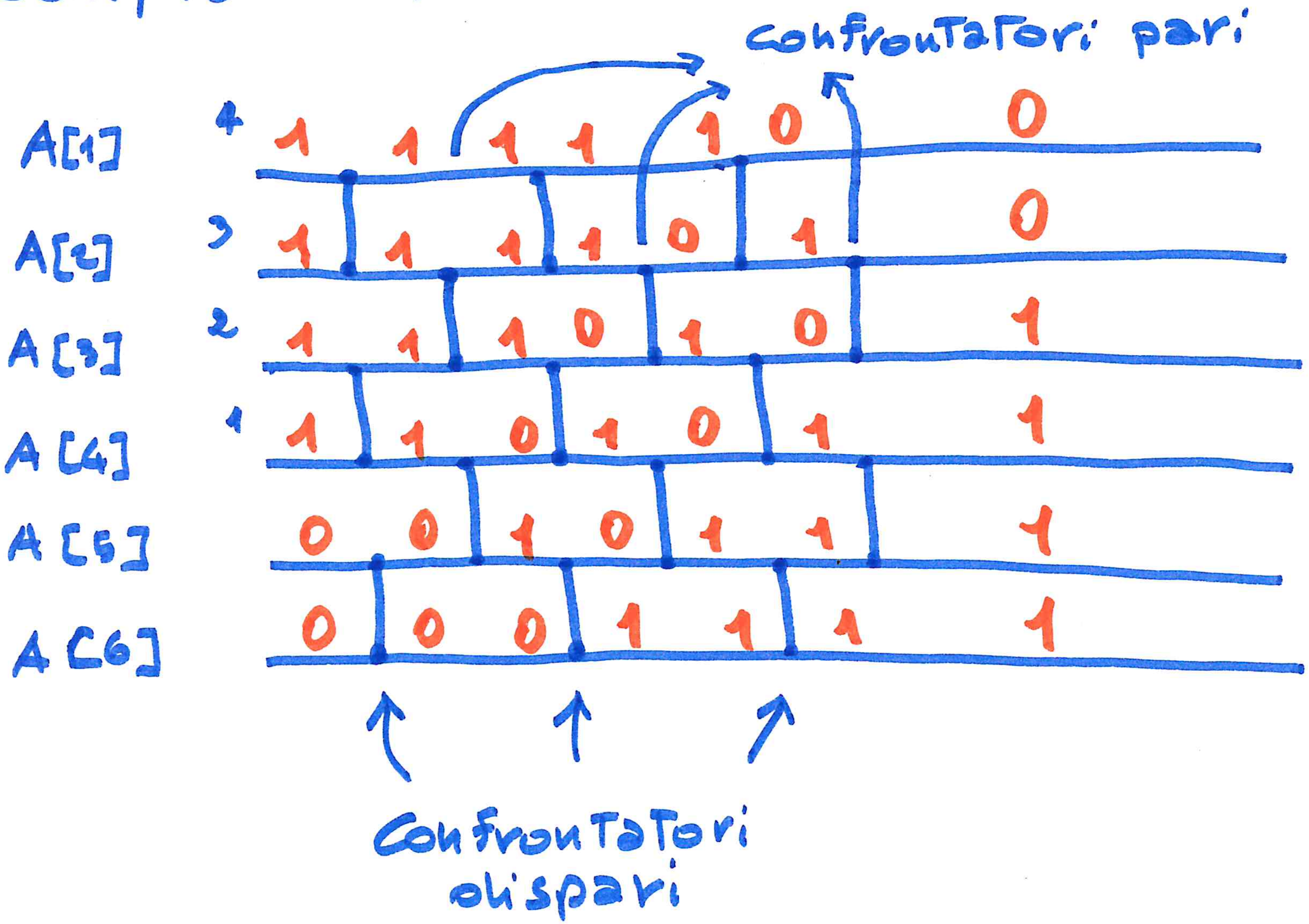
Diamo un algoritmo Test/swap oblivious

descritto da una sorting network

$\rightsquigarrow$

ODD/EVEN sorting network

Esempio con 6 elementi da ordinare:



Per la correttezza di ODD/EVEN usiamo il principio 0/1 (Knuth '72)

$$\{0,1\}^n \longrightarrow \boxed{\text{ODD/EVEN}} \longrightarrow 0^j 1^e$$

$$j+e=n$$

Partiamo dall'esempio

$$\underbrace{1111}_{e=4}\ \underbrace{00}_{j=2} \qquad \text{(caso peggiore)}$$

Osservazione

Nel nostro caso ogni 1 deve scendere di $n-e = j$ (# di zeri) posizioni

Esempio: $n-e=2$

	Tempo
1° 1 dal basso	$2+1$ ritardo
2° 1 dal basso	$2+2$
3° 1 dal basso	$2+3$
4° 1 dal basso	$2+4$

di uni nell'input $= e$

Regola:

i^o / l'iesimo uno dal basso impiega (*) $n-e+i$ passi per posizionarsi correttamente

(*) in generale è un "al più" $n-e+i$ passi

Dato che $i \leq e$ si ottiene:

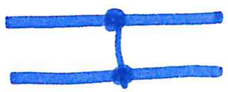
$n-e+e=n$ passi (*) al più n passi

$\Rightarrow$ n passi di ODD/EVEN sono necessari e sufficienti

Implementazione con codice:

- sequenziale $T(n) = O(n^2)$
- parallelo: Haberman '72

n passi paralleli o ROUND di

 comparatori implementati con MINMAX(K, K+1)

```
for i = 1 to n
    for K ∈ {2t - (i % 2) | 1 ≤ t ≤ n/2} par do
        MINMAX(K, K+1)
```

Tempo: $\underset{\text{ROUND}}{n} * \underset{\text{MINMAX in parallelo}}{4} = O(n)$

Efficienza: $\dfrac{\cancel{n} \log n}{\cancel{n} \cdot n} \to 0$

Osservazione:

Per n processori in un array lineare l'ORDIN. vuole Tempo: $\Omega\left(\dfrac{n}{2\underset{1}{\beta}}\right)$

Riduzione dei processori : p

1° step. : ogni processore si prende $\frac{n}{p}$ dati e li ordina

Tempo : $O\left(\frac{n}{p} \log \frac{n}{p}\right)$

Algoritmo: MERGE - SPLIT

- la primitiva merge-split avviene tra due processori contigui :

— il processore di sin. spedisce $\frac{n}{p}$ dati ordinati al proc. di des. $O(n/p)$

— il processore di des. riceve e fonde i nuovi n/p dati con i suoi n/p dati (ordinati) (MERGE) $O(n/p)$

- il processore di des invia i dati più piccoli (n/p) al processore di sin $O(n/p)$

(SPLIT)

Ogni singola primitiva merge-split viene ripetuta per p ROUND

È ODD-EVEN dove MINMAX è sostituito con la primit. MERGE-SPLIT

Tempo: $\frac{n}{p} \log \frac{n}{p} + \cancel{p} * \frac{n}{\cancel{p}} = O(n)$

Denominatore di E: $p \left(\frac{n}{p} \log \frac{n}{p} + n \right) =$

$= n \log \frac{n}{p} + n * p$

scegliamo $p = \log n$

Efficienza : $\dfrac{n \log n}{n \log n} \rightarrow C \neq 0$

OSSERVAZIONE : il tempo è rimasto $O(n)$

Ciò si spiega in quanto la riduzione dei processori agisce sul diametro e non sull'ampiezza di bisezione: $\beta = 1$

$\Downarrow$

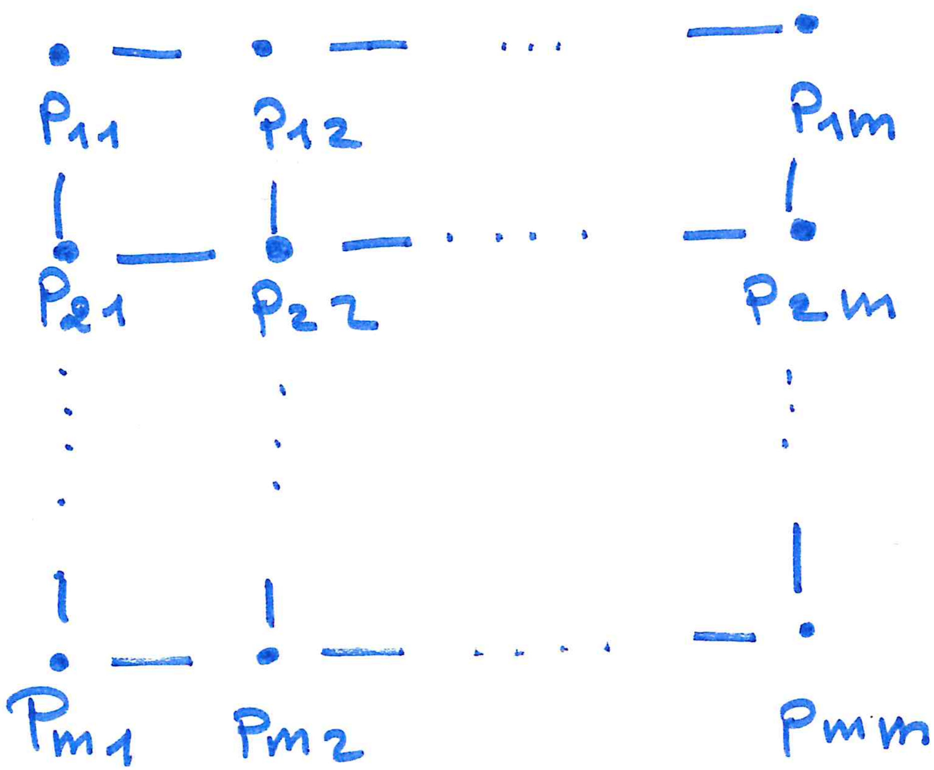
Architetture MESH

array bidimensionale ovvero griglia di processori

Utilizzate per i SUPERCOMPUTER

Architettura MESH

n processori



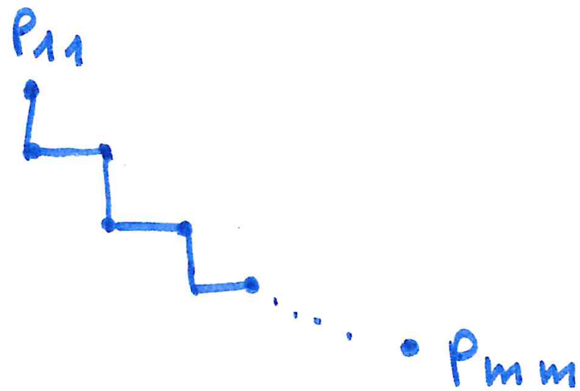
$m = \sqrt{n}$

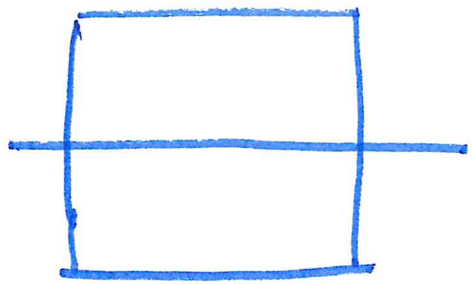
Se n non è un quadrato perfetto:

$$(\lfloor\sqrt{n}\rfloor+1)^2 \le 2n \quad \text{per} \quad n \geq 6$$

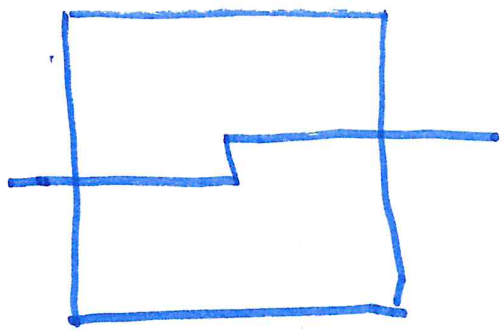
Parametri di rete:

- $\rho = 4$
- $\delta \approx 2\sqrt{n}$
- $\beta \sim \sqrt{n}$





se $\sqrt{n}$ è pari



se $\sqrt{n}$ è dispari

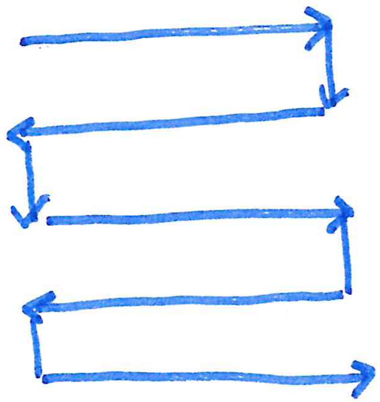
Lower bound per:

- MAX $\Omega(\sqrt{n})$
- ORDINAMENTO $\Omega(\sqrt{n})$

AFFRONTIAMO IL PRIMO PROBLEMA:

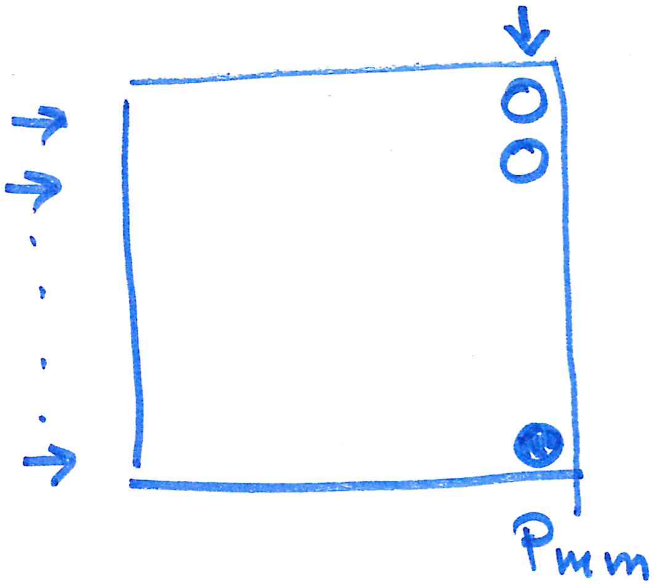
idea immediata usiamo la mesh come array lineare

di n processori



ma si ottiene $\Omega(n)$ per il Tempo di soluzione di MAX contro $\Omega(\sqrt{n})$

Algoritmo RIGHE-COLONNA per MAX



ogni riga è vista come un array lineare di $\sqrt{n}$ proces.

l'ultima colonna è un altro array lineare di $\sqrt{n}$ proces.

CODICE

TEMPO

for $i = 1$ To $\sqrt{n}$ pardo

$\sqrt{n}$ [MAX $(P_{i1}, P_{i2}, \ldots, P_{im})$

$+$

$\sqrt{n}$ [MAX $(P_{1m}, P_{2m}, \ldots, P_{mm})$

$m = \sqrt{n}$

$= O(\sqrt{n})$

Efficienza: $\frac{\cancel{n}}{\cancel{n} \cdot \sqrt{n}} \to 0$

$\Rightarrow$ operiamo una riduzione dei processori

$n \rightsquigarrow p$

- ogni processore calcola il max tra n/p dati
  Tempo $O(n/p)$

- si attiva l'algoritmo di prima sulla griglia $\sqrt{p} \times \sqrt{p}$
  Tempo $O(\sqrt{p})$

$t = \underbrace{\frac{n}{p} + \sqrt{p}}_{\text{Tempo TOT.}}$

Denom. di E. $= p\left(\frac{n}{p} + \sqrt{p}\right) = n + p^{1+\frac{1}{2}}$

$= n + p^{3/2}$

Se fosse n avrei : $E = \frac{n}{n} \to c \neq 0$

Richiedo (o scelgo)

$p^{3/2} = n \Rightarrow p = \sqrt[3]{n^2} = n^{2/3}$

Tempo TOT : $\frac{n}{p} \to n^{1-\frac{2}{3}} = n^{1/3} = \sqrt[3]{n}$

$+ \qquad\qquad\qquad\qquad\qquad\qquad +$

$\sqrt{p} \to \sqrt[3]{\sqrt{n^2}} = \sqrt[3]{n}$

$= O(\sqrt[3]{n})$

Lower bound per MESH

$\sqrt{p} \times \sqrt{p}$ è $\Omega(\sqrt{p})$ ~~per~~ il MAX con $p = n^{2/3}$

$= \sqrt[3]{\sqrt{n^2}} = \sqrt[3]{n} \Rightarrow \Omega(\sqrt[3]{n})$