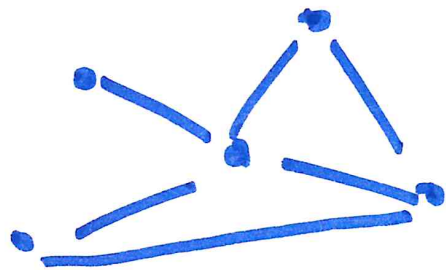


Architettura parallela a memoria distribuita



• = processori RAM
con istruzioni < calcolo
comunic.

— = connessioni
full-duplex

TOPOLOGIA DELLA RETE impatta sul Tempo
(velocità comunicazione)

γ = grado

δ = diametro ($T_{max} = \Omega(\delta)$)

β = ampiezza ($T_{ord} = \Omega(\frac{n}{\beta})$)

Per affrontare i problemi max e ordinamento
introduciamo: i confrontatori, e primitive

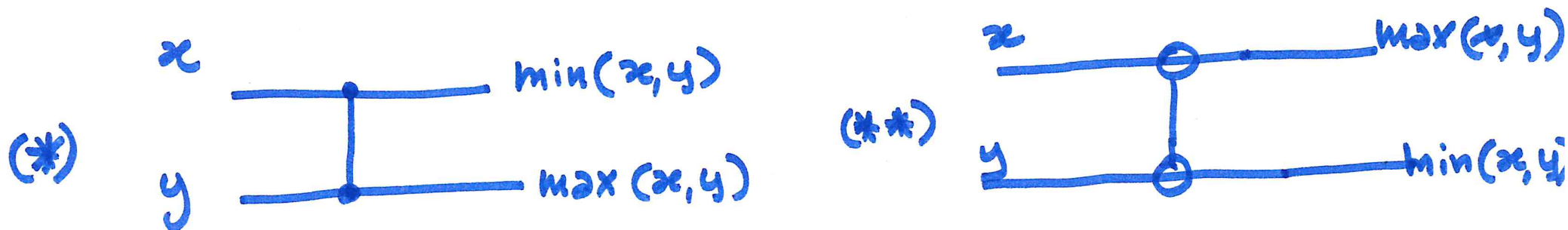
Def. le istruzioni di confronto per i confrontatori o
comparatori

(*) if ($A[i] > A[j]$) then SWAP ($A[i], A[j]$)

(**) if ($A[i] < A[j]$) then SWAP ($A[i], A[j]$)

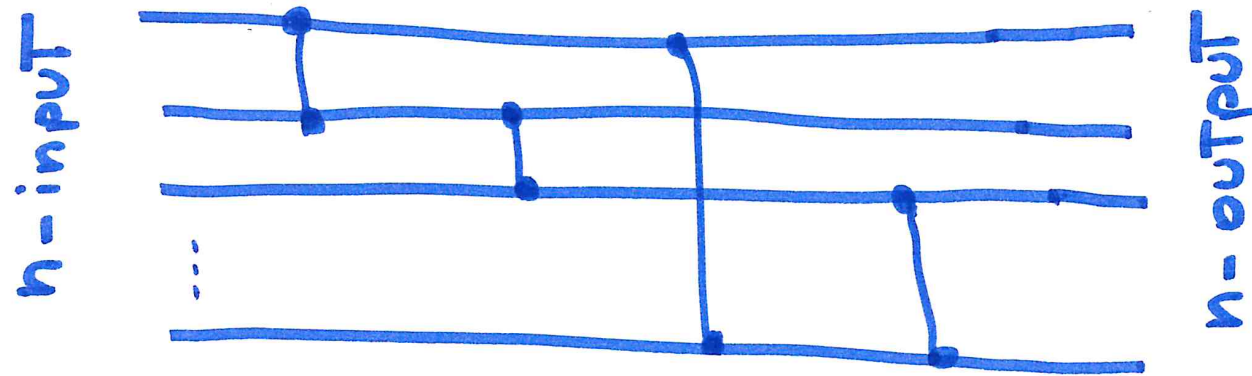
con $i < j$ -

Rappresentazione di (*) e (**) con i comparator

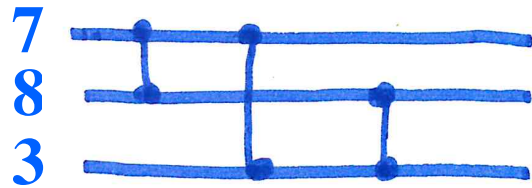


COMPARATORI / CONFRONTATORI

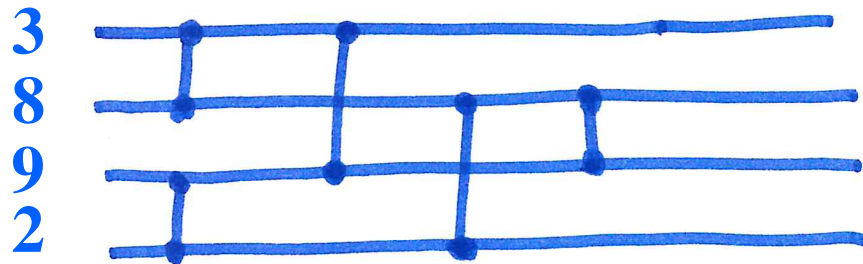
Reti di Confrontatori



Esempi

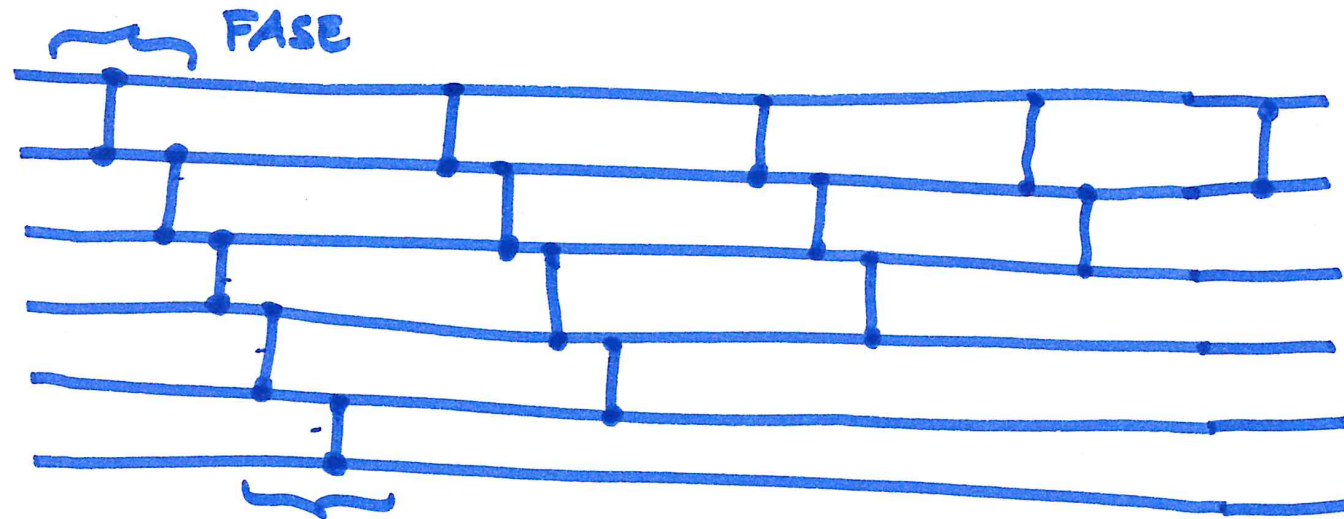


SORTING NETWORK
Rete di confrontatori
(per 3 elementi) che ordina



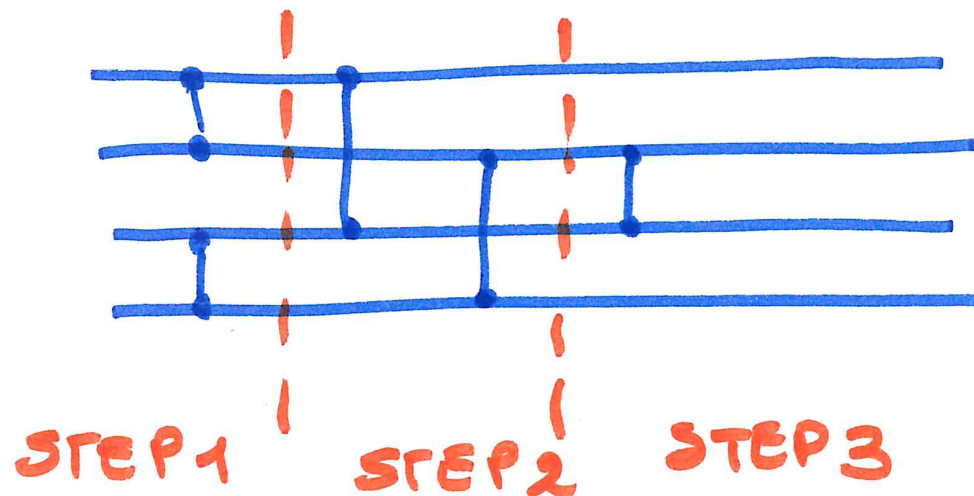
SORTING NETWORK
per 4 elementi

ALGORITHM DI ORDINAMENTO BUBBLE SORT con S.N.



Ad ogni fase
l'elemento +
pesante
viene spinto
verso il basso

SORTING NETWORK per 4 elementi



STEP: un filo
viene coinvolto
eventualmente
da un solo confrontatore

$$T(n) = \# \text{ STEP}$$
$$p(n) = 4$$

Formalmente definiamo una rete di confrontatori con

$$R \left(\underbrace{x_1, \dots, x_n}_{n\text{-input}} \right) = \left(\underbrace{y_1, \dots, y_n}_{n\text{-output}} \right)$$

si dice che R è una sorting network se

$\forall (x_1, \dots, x_n) \in \mathbb{N}^n$ vale

$$R(x_1, \dots, x_n) = (y_1, \dots, y_n) \text{ con}$$

$$y_1 < y_2 < \dots < y_n$$

dette anche reti di ordinamento ~~di~~

TEST / SWAP "OBLIVIOUS" ←

Test fissati
a priori
che non dipendono
dall'input

Domanda: R è una sorting network?

Risposta: USA IL PRINCIPIO 0-1

Formalmente:

$\forall x \in \{0,1\}^n$ se $R(x)$ è ordinato

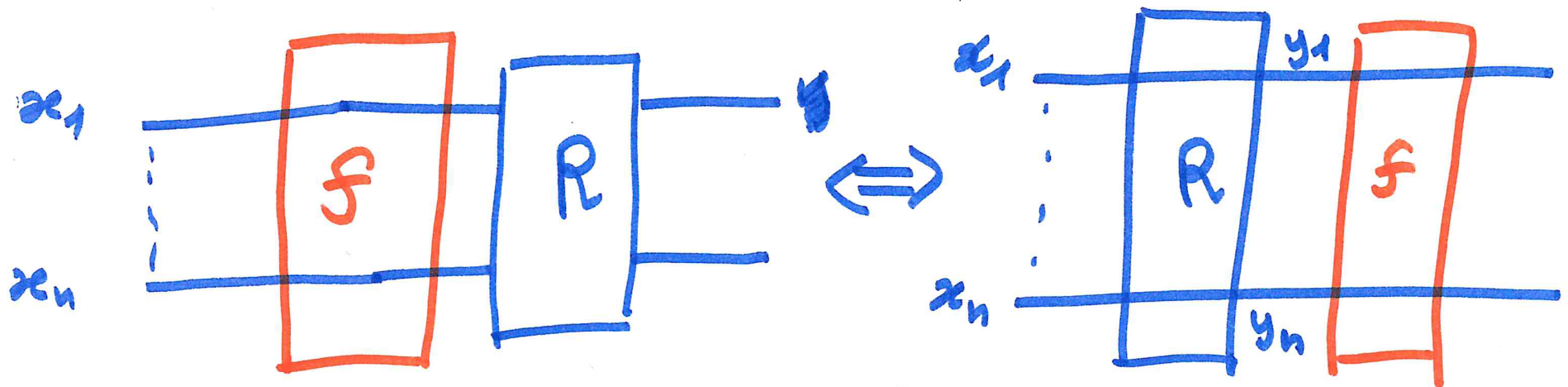
$\Rightarrow \forall x \in \mathbb{N}^n$ si ha $R(x)$ è ordinato

\equiv

$\exists x \in \mathbb{N}^n$ t.c. $R(x)$ non è ordinato

$\Rightarrow \exists x \in \{0,1\}^n$ t.c. $R(x)$ non è ordinato

A livello di Rete
di confrontatori
si ha



f -shift su R

$$R(f(x_1), \dots, f(x_n)) = \vec{f}(R(x_1, \dots, x_n)) \\ = (f(y_1), \dots, f(y_n))$$

per f monotona crescente

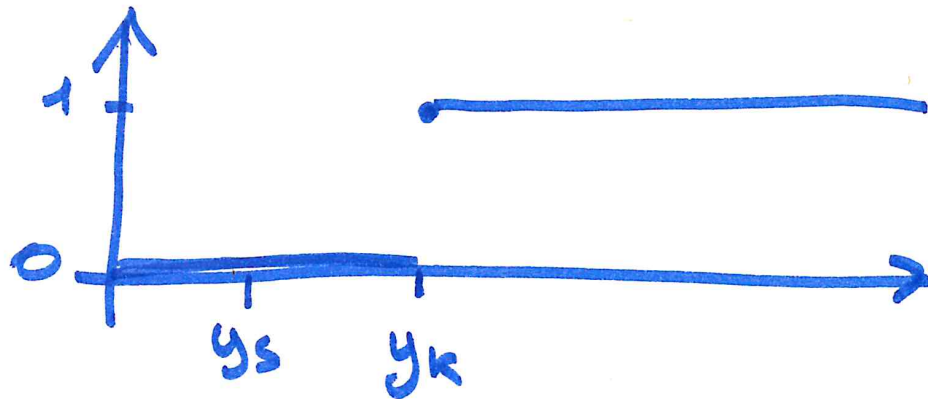
Se R NON corretta:

$$\exists \underline{x} \in \mathbb{N}^n \text{ t.c. } R(\underline{x}) = (y_1, \dots, \overset{\substack{\uparrow \\ \text{+ grande}}}{y_k}, \dots, \overset{\substack{\uparrow \\ \text{+ piccolo}}}{y_s}, \dots, y_n)$$

Definiamo $g: \mathbb{N} \rightarrow \{0, 1\}$:

$$g(x) = \begin{cases} 1 & x \geq y_k \\ 0 & \text{altrim.} \end{cases}$$

$$y_k > y_s \\ k < s$$




g è monotona crescente

Se ora applico g prima di R ottengo:

(*) $R(g(x_1), \dots, g(x_n)) \leftarrow$ applico R ad un vettore binario


ORA per la regola dello shift (g -shift)

(*) = $(g(y_1), \dots, g(y_k), \dots, g(y_s), \dots, g(y_n))$


vettore binario


1


0

non ordina
1 vettore binario 

OSSERVAZIONE:

Per testare una R e capire se è SORTING \Rightarrow
valuto R solo su input binari