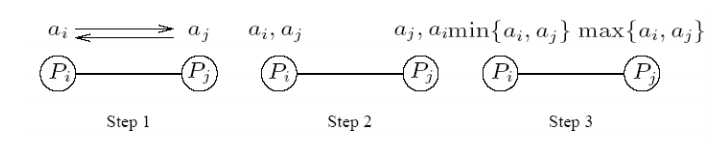
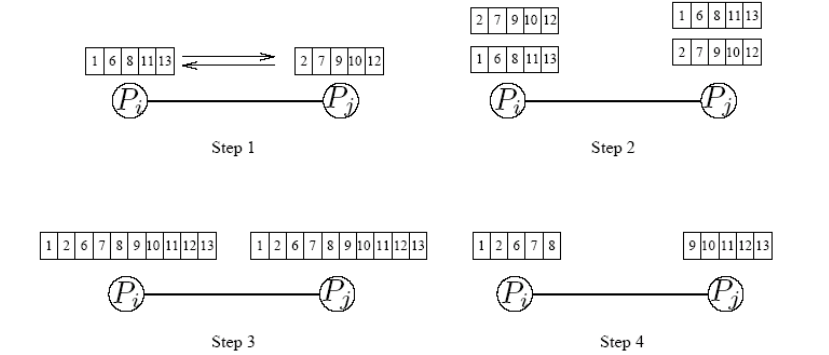
SORTING

# SORTING: BASICS

Una lista è ordinata se ogni sotto-lista è ordinata e ogni elemento di essa segue una regola: *un elemento sul processo Pi è sempre più piccolo di un elemento sul processo Pj se e solo se i<j.* Un algoritmo parallelo può persino raggiungere una complessità di **O (log n)**. L’operazione fondamentale alla base di questi algoritmi è il **compare-exchange :**



Se ci sono più elementi per ogni processo si effettua un’operazione chiamata **compare split**: si ordinano n/p elementi per ogni processo in modo che i più piccoli vadano sul processo Pi mentre i più grandi sul processo Pj.

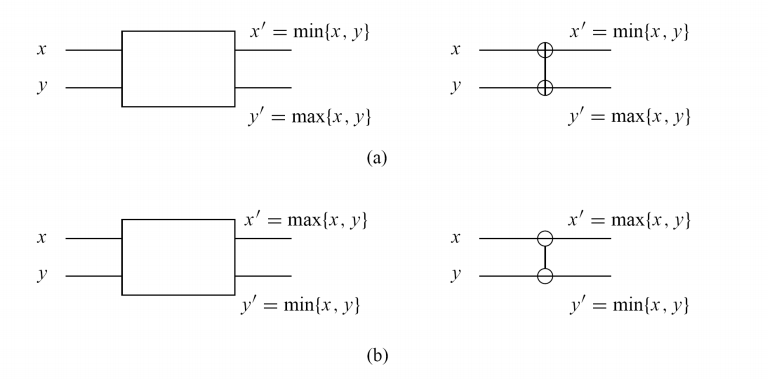


# SORTING NETWORKS

In particolare sono dei **comparator networks**, designati appositamente per il sorting.

**Comparatore:** dispositivo che dati in input 2 valori (x, y) ne produce altri 2 (x1, y1), esso può essere:

* Incresing +
  + X1 = min {x, y}
  + Y1 = max {x, y}
* Descreasing -
  + Viceversa

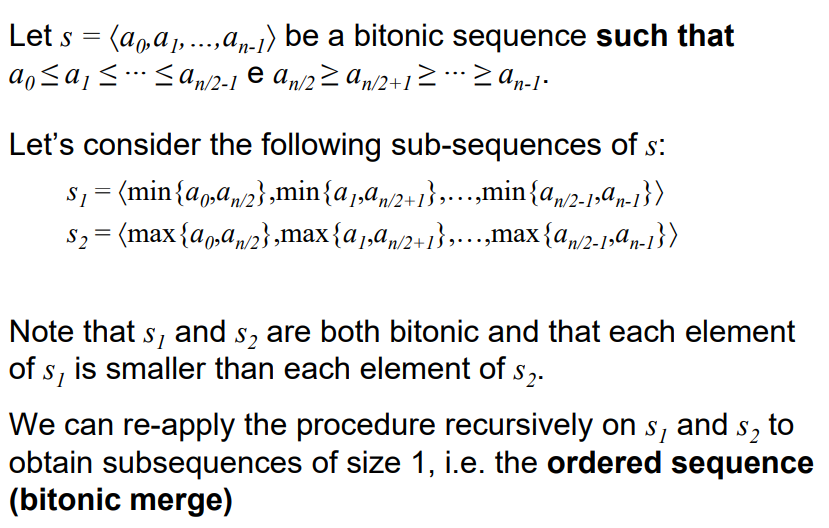


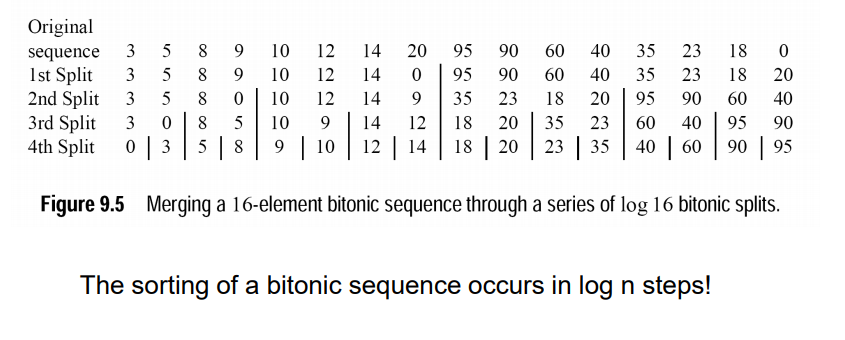
# BITONIC SORTING NETWORKS

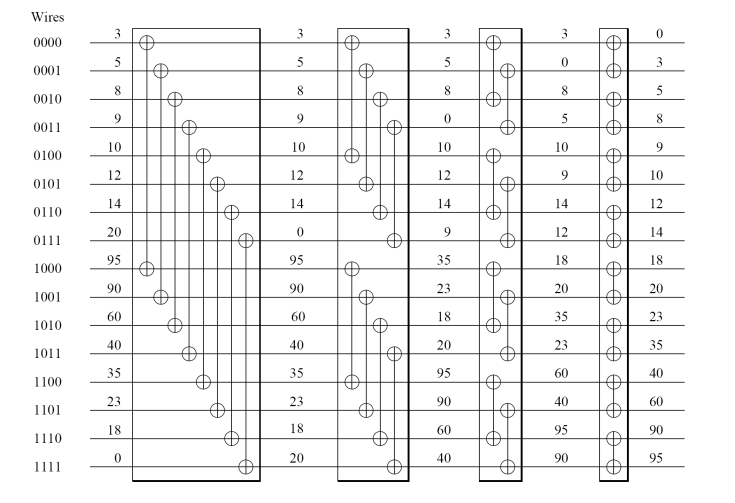
Serve ad ordinare n elementi con una complessità: **Θ(log2n)**.

**Bitonic sequence**: sequenza di numeri che ha due tonalità: ascendente, discendente. L’obbiettivo è quello di ordinare una sequenza bitonica in un’ordinata. 〈8,9,2,1,0,4〉 -> 〈0,4,8,9,2,1〉. Si dividono le sotto-sequenze, si esegue il **Compare Exchange**,si riapplica ricorsivamente la procedura fin quando non si ottiene una sotto-sequenza di lunghezza 1.

## ORDINARE UNA SEQUENZA BITONICA

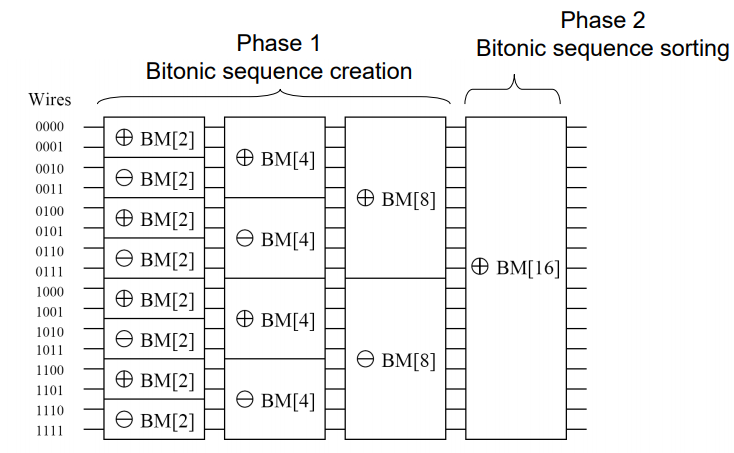


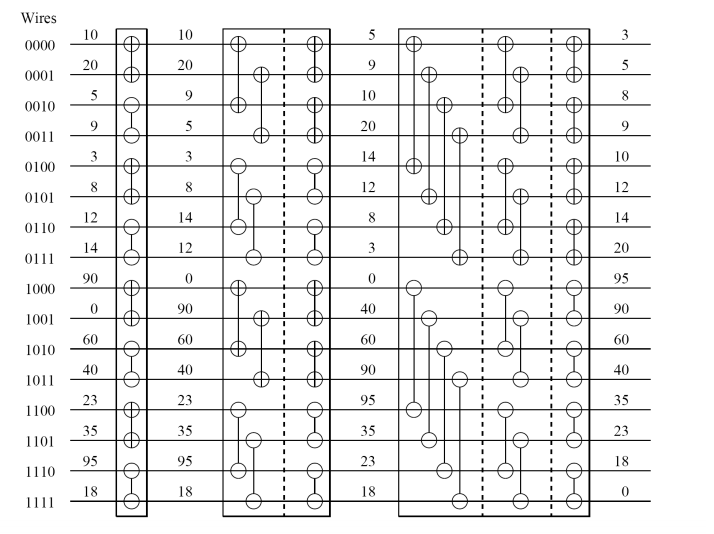
Per implementare il **Bitonic merge algorithm** si crea un network noto come **Bitonic merging network**: logn colonne dove ognuna ha n/2 comparatori.

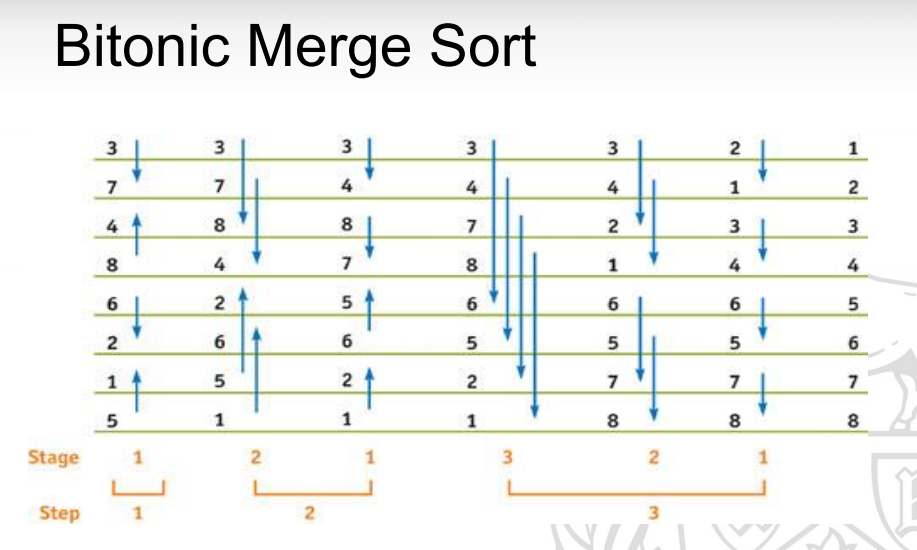


## ORDINARE UNA SEQUENZA NON BITONICA

L’idea: ***una sequenza di lunghezza 2 è bitonica! quindi possiamo pensare una sequenza di elementi non ordinati come una concatenazione di sotto-sequenze bitoniche di lunghezza 2.***







# BITONIC SORT FOR HYPERCUBES

Nel sorting tutti gli scambi avvengono tra processi che differiscono di un bit, negli **hypercubes** la distanza tra due nodi è uguale al numero di bit per cui differiscono. Perciò queste operazioni sono molto efficienti poiché le operazioni vengono effettuati tra nodi adiacenti.