Parallel Sorting Basics

Storing Input/output

- They are stored in distributed memory
  - 우리는 single processor의 memory에 store되었음

Parallel sorted sequence

- sequence는 processor 따라 partition된다
- each processor의 sub-sequence를 sort한다
  - k<j 라면, element value of subsequence of P_k < element value of subsequence of P_j

Element-wise approach (~~Compare-exchange~~)

1:1

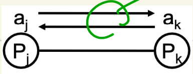
Compare-exchange

- 각 Processor가 1 element를 own함
- prev processor는 항상 next processor 보다 작은 value 갖고있음

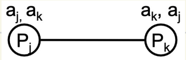
Steps

① Communication step

동시를 해도 될수있음!



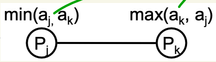
- bidirectional channel



- each has both elements after Communication

② Comparison step

select target

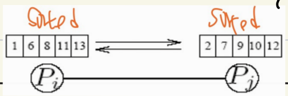


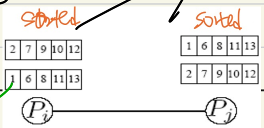
- both elements 중에서 각 Processor 순서에따라 Min/Max element를 select한다

Bulk approach (Compare-Split)

Split

① Communication step



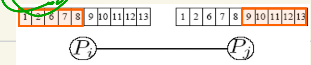


both를 각각 갖고있음

- 가지고있는 bulk block (n/p)를 communicate 한다.

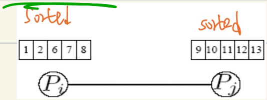
② Merge step

merge 각자하고



- 각자 Merge 진행
- Merge 결과 동일해야함

③ Split step



- Prev Processor가 least half를
- next Processor는 most half를 가져옴!

Sorting Network

Sorting Network

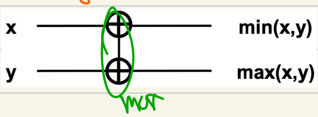
: Sorting을 Θ(nlgn) 보다 작게 하기위해서 고안된 sorting 전용 network

- Comparison이 network 단계에서 함께 수행된다.

Comparator

: Unsorted input (x,y)를 받고, sorted input (x',y')를 return함

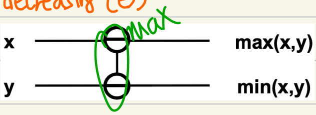
- Increasing (⊕)



=x'
=y'

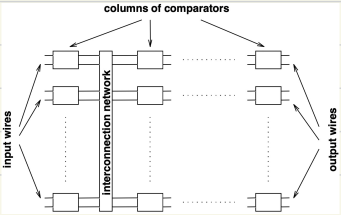
bulk/element 다 삽가능 ?

- decreasing (⊖)



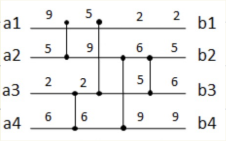
Network Structure

- series of columns
  consists of vector of comparator



- depth of network
  = # of columns

ex)



Sorting Network - Bitonic Sort

★★★✓ $\Theta(\log^2 n)$

Bitonic Sort

- Sorts n elements in $\Theta(\log^2 n)$ time
- rearrange bitonic sequence into sorted sequence

Can parallelize

각 comparator가 independent하기 때문?

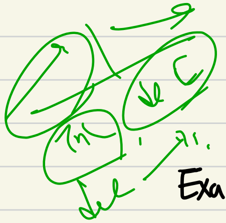
- Bitonic compares elements in predefined sequence (independent on data)

bitonic sequence : sequence $<a_0, a_1, \cdots, a_{n-1}>$ where either

① $<a_0, \cdots, a_i>$ 는 monotonic increasing and $<a_i, \cdots, a_M>$ 는 monotically decreasing

or

② there exists cyclic shift of indices so that ① is satisfied



Example

- <1, 2, 4, 7, 6, 0> is bitonic sequence (① 조건 만족)

increase | decrease

- <8, 9, 2, 1, 0, 4> is bitonic sequence (② 조건 만족)

decrease | increase

but cyclic shift of <0, 4, 8, 9, 2, 1>

inc | dec

048921

Sorting Network - Bitonic Sort (Cont.)

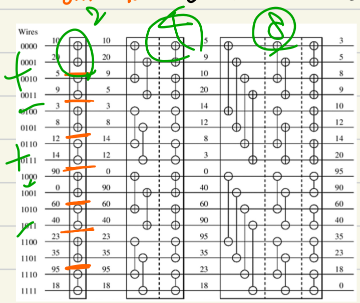
→ unsorted → bitonic sequence

1 2 4 8 . . .

Building a Bitonic Sequence

· How to build bitonic sequence from unsorted sequence?

① 2 length sequence는 언제나 bitonic sequence 이다. inc/dec 순서로 만들어주기



next length를 위한 준비

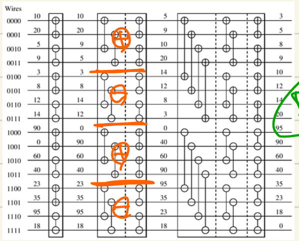
length 2 : + | −

(8) 4 : + | −

++ −−

② 4 length sequence는 step ①을 통해서 bitonic sequence가 되어있다.

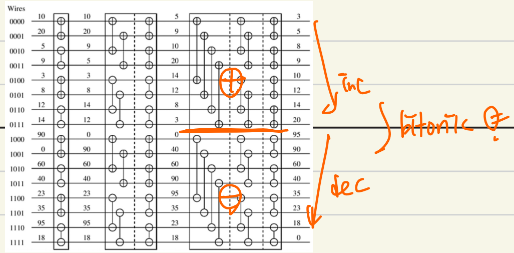
8 length sequence를 bitonic으로 만들기위해 ⊕ ⊖ length 4로 반복



BM(8)

③ 8 length sequence는 step ②를 통해서 bitonic sequence가 되어있다.

16 length sequence가 bitonic 되기위해 ⊕ ⊖ length 8로 반복

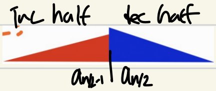


Sorting Network - Bitonic Sort (Cont.)

bitonic split : bitonic sequence를 2개의 bitonic subsequence로 split 하는 방법

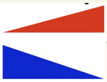
input로 → output로 bitonic sequence

- $S = \langle a_0, a_1, \cdots, a_{n-1} \rangle$은 bitonic sequence



—$a_0 \leq a_1 \leq \cdots \leq a_{n/2-1}$, and

—$a_{n/2} \geq a_{n/2+1} \geq \cdots \geq a_{n-1}$



- subsequences of s

$s_1 = \langle \min(a_0, a_{n/2}), \min(a_1, a_{n/2+1}), \ldots, \min(a_{n/2-1}, a_{n-1}) \rangle$

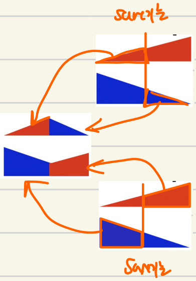
$s_2 = \langle \max(a_0, a_{n/2}), \max(a_1, a_{n/2+1}), \ldots, \max(a_{n/2-1}, a_{n-1}) \rangle$

min끼리

max는 [illegible]

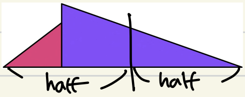
↓

S_1 and S_2 are bitonic

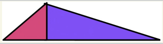


- repeat bitonic split until sorted sequence
- Increasing / decreasing 파트의 length가 달라도 bitonic split 가능하다

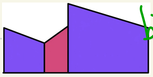
-ex



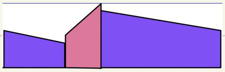
① S_1 : get min between two half →



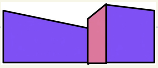
② S_2 : get max between two half →



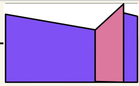
-ex



① S_1 : get min between two half ⇒



② S_2 : get max between two half ⇒

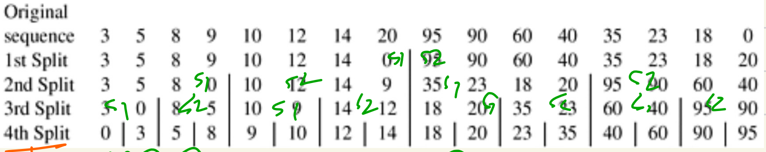


Sorting Network - Bitonic Sort (Cont.)

Bitonic Merge

: Procedure of Sorting bitonic sequence using bitonic splits

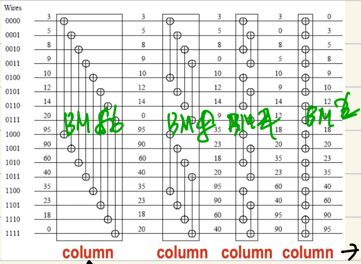
- log n 만큼만 split 하면됨



Original sequence	3	5	8	9	10	12	14	20	95	90	60	40	35	23	18	0
1st Split	3	5	8	9	10	12	14	0	95	90	60	40	35	23	18	20
2nd Split	3	5	8	0	10	12	14	9	35	23	18	20	95	90	60	40
3rd Split	3	0	8	5	10	9	14	12	18	20	35	23	60	40	95	90
4th Split	0	3	5	8	9	10	12	14	18	20	23	35	40	60	90	95

sorted 됨

- network 모습 (bitonic merging network)



→ log n columns

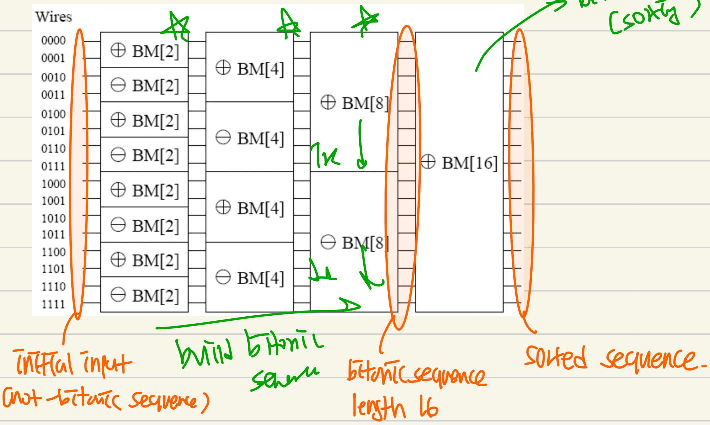
each column contains n/2 Comparator (항상 동일함)

* ⊕ → ⊖ 바꾸면 decreasing order 됨

Sorting Network - Bitonic Sort (Cont.)

Complexity

- depth of network is $\theta(\log^2 n)$



- $\log_2 n$ merge stages
- jth merge stage is $\log_2 2^j = j$

$$\downarrow$$

$$\text{depth} = \sum_{j=1}^{\log_2 n} \log_2 2^j = \sum_{j=1}^{\log_2 n} j = (\log_2 n + 1)\cdot(\log_2 n)/2 = \theta(\log^2 n)$$

- Complexity of Implementation, $\theta(n \log^2 n)$ ← # of depth

$\frac{n}{2}$ comparators per stage

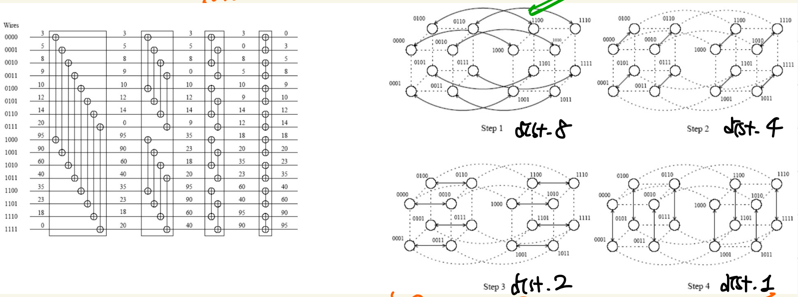
stage별 comparator 개수

Mapping Bitonic Sort to Hypercube

Bitonic Sort with Hypercube

- direct mapping of wires to processors

one-to-one mapping



Step 1 dist. 8

Step 2 dist. 4

Step 3 dist. 2

Step 4 dist. 1

모든 Sorting network는 Compare-and-exchange ⊖

(1bit 다른 node끼리 comm함)

Sample Sort

Sample Sort

Steps

- each processor sorts its local data
- each processor selects sample vector of size p-1

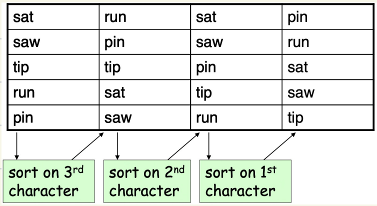
Radix Sort

radix sort

- Start at least significant digit (오른쪽)
  - *radix = digit or position to sort
- Sort numbers in current digit
- move to next least significant digit
- most significant digit 정렬하면, sequence is sorted

모든 item의 길이가 같아야함

Example



sat	run	sat	pin
saw	pin	saw	run
tip	tip	pin	sat
run	sat	tip	saw
pin	saw	run	tip

sort on 3rd character

sort on 2nd character

sort on 1st character

- Cost = $O(\underbrace{\# \text{keys}}_{\# \text{rows}} * \underbrace{\# \text{characters}}_{\# \text{columns}})$