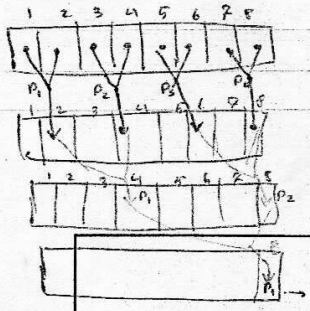
2/8/2014



محاسبه جمع و همه جمع ها

- محاسبه حاصل جمع عناصر یک آرایه به طول n ((A[1..n])

$$A[1]+A[2], A[1]+A[2]+A[3], \ldots, A[1]+A[2]+\cdots+A[n]$$

- استفاده از مدل EREW
- قرار دادن اعداد در برگهای یک درخت دودویی و انجام جمع در هر سطح درخت بطور موازی.
- استفاده از n/2 پردازنده.
- حاصل جمع در A[n] قرار خواهد گرفت.
- فرض شود n توانی از ۲ باشد: زمان اجرای الگوریتم O(logn) خواهد بود.

7

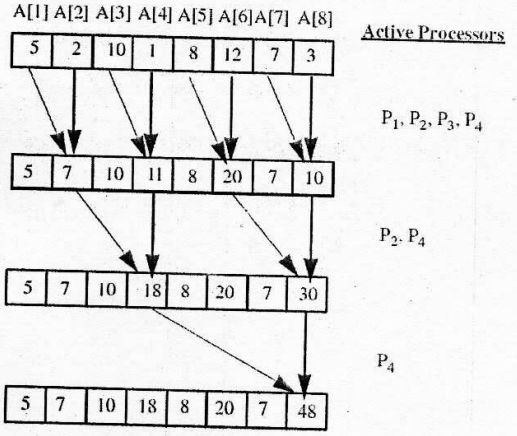
الگوریتم جمع آرایه ای از اعداد

```
Algorithm Sum_EREW
for i=1 to log n do
  forall Pj, where 1 ≤ j ≤ n/2 do in parallel
    if (2j modulo 2^i)=0 then
      A[2j] ← A[2j] + A[2j - 2^(i-1)]
    endif
  endfor
endfor
```

8

2/8/2014

الگوریتم جمع آرایه ای از اعداد



9

تحلیل پیچیدگی الگوریتم جمع آرایه ای از اعداد

- زمان اجرا: T(n)=O(logn) است چراکه حلقه for، log n با اجرا می شود.
- تعداد پردازنده ها: P(n)=n/2
- هزینه: C(n)=O(nlogn)

- **با توجه به اینکه یک الگوریتم متوالی مناسب می تواند حاصلجمع لیستی از اعداد را در O(n) محاسبه کند => الگوریتم فوق بهینه نیست.**

10

معایب الگوریتم قبل

- هزینه: C(n)=O(nlogn): بهینه نیست.
- اغلب پردازنده ها بیکار هستند.
- در طی تکرار iام، تنها n/(2^i) پردازنده فعال هستند.

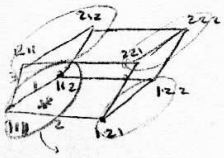
11

پروژه ۳-۱

- محاسبه بردار ۱۶ عنصری A با الگوریتم فوق با استفاده از نخها در C++ یا JAVA
- توجه: هر نخ مبین یک PE می باشد.

12

2/8/2014



$P_{i,j,k} = A[i,k] * b[k,j]$

الگوریتم ضرب ماتریس

- ضرب دو ماتریس n*n
- A[1..n, 1..n] و B[1..n,1..n]
- n توانی از 2 است.
- از مدل CREW استفاده می شود.
- استفاده از n^3 پردازنده
- پردازنده ها در یک آرایه سه-بعدی مرتب شده اند. P(i,j,k) داراي شاخص (i,j,k) است.
- یک آرایه سه-بعدی C[i,j,k] در حافظه مشترک ذخیره شده است و بعنوان فضاي کاري است. ماتریس حاصله در مکانهاي C[i,j,n] که 1≤i,j≤n ذخیره خواهد شد.

13

مراحل الگوریتم ضرب ماتریس

- مرحله اول:
  - تمامي n^3 پردازنده ، n^3 عمل ضرب را انجام مي دهد.
  - در ماتریس خروجي ، براي هر n^2 المان ، n حاصلضرب محاسبه مي شود.
  - پردازنده P(i,j,k) حاصلضرب A[i,k]*B[k,j] را محاسبه کرده و در C[i,j,k] قرار مي دهد.
- مراحل دوم:
  - N حاضرب محاسبه شده براي هر المان ماتریس خروجي ، براي تولید نتیجه نهایي این سلول ، جمع مي شوند.
  - ایده الگوریتم Sum-EREW در k بعد ، n^2 بار به طور موازي براي محاسبه C[i,j,n] بکار گرفته مي شود (1≤i,j≤n)

14

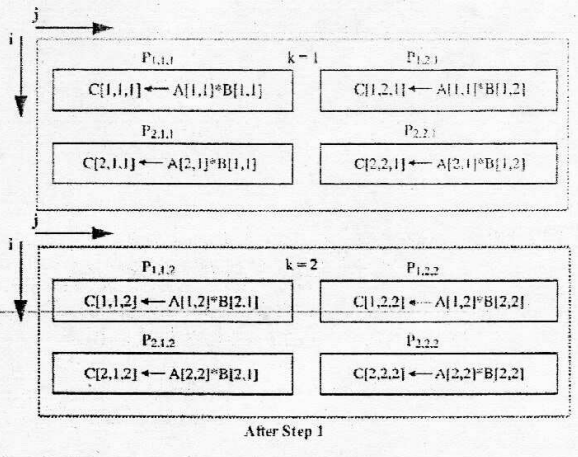
شبه کد الگوریتم ضرب ماتریس

```
Algorithm MatMult_CREW
/* Step 1 */
forall P(i,j,k), where 1 ≤ i, j, k ≤ n do in parallel
  C[i,j,k] ← A[i,k]*B[k,j]
endfor

/* Step 2 */
for l=1 to log n do
  forall P(i,j,k), where 1 ≤ i, j ≤ n & 1 ≤ k ≤ n/2 do in parallel
    if (2k modulo 2^l)=0 then
      C[i,j,2k] ← C[i,j,2k] + C[i,j, 2k − 2^(l−1)]
    endif
  endfor
  /* The output matrix is stored in locations
    C[i,j,n], where 1 ≤ i,j ≤ n      */
  endfor
```

15

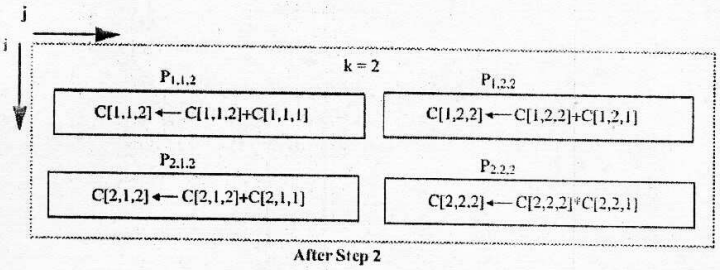
مرحله اول الگوریتم ضرب ماتریس



After Step 1

16

مرحله اول الگوریتم ضرب ماتریس



After Step 2

17

پیچیدگي الگوریتم ضرب ماتریس

- زمان اجرا: T(n)=O(logn)
- تعداد پردازنده ها: P(n)=n^3
- هزینه: C(n)=O(n^3 logn)

- هزینه بهینه است؟

18

کاهش تعداد پردازنده ها در الگوریتم ضرب ماتریس

- در مرحله اول کلیه پردازنده ها مشغول هستند.
- در مرحله دوم تمامي پردازنده ها عمل جمع انجام نمي دهند.
- Log n تکرار وجود دارد. در تکرار اول (n^3)/2 پردازنده فعالند ، در تکرار دوم (n^3)/4 پردازنده فعالند => مي توان از ماشيني استفاده کرد که (n^3)/log n پردازنده دارد.
- روش:
- پردازنده P(i,j,k) که 1≤ k ≤n/logn است ، مجموع log n جاصلضرب را محاسبه مي کند. اين مرحله (n^3)/logn جمع جزئي توليد مي کند.
- مجموع حاصلضربهاي توليد شده در مرحله اول با هم جمع شده تا ماتريس حاصله بدست آيد.

19

پروژه 3-2

- محاسبه ضرب دو ماتريس 4*4 با الگوريتم فوق با استفاده از نخها در C++ يا JAVA
- توجه: هر نخ مبين يک PE مي باشد.

20

پیچیدگي الگوریتم ضرب ماتریس کاهش یافته

- زمان اجرا: T(n)=O(logn)
- تعداد پردازنده ها: P(n)= (n^3)/log n
- هزینه: C(n)=O(n^3)

- هزینه بهینه است؟

21

فصل پنجم

الگوریتم های مرتب سازی

موضوعات مرتب سازی روی کامپیوترهای موازی

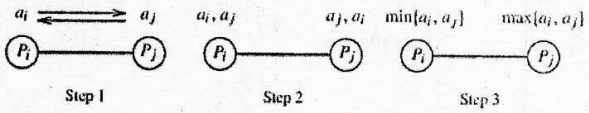
- در الگوریتم های مرتب سازی ترتیبی، دنباله های ورودی و خروجی در حافظه فرایند ذخیره می شوند.
- در الگوریتم های موازی دو مکان برای ذخیره سازی دنباله ها وجود دارد:
  1. در یکی از فرایندها فقط ذخیره شوند.
  2. بین فرایندها توزیع شوند.
- فرض می شود که عناصر دنباله خروجی روی فرایندها توزیع شده است، بعبارتی اگر Pi قبل از Pj باشد، آنگاه عنصر ذخیره شده در Pi کوچکتر از Pj است.

2

چگونگی انجام مقایسه در الگوریتم های موازی

- به ازای هر فرایند یک عنصر :

Figure 9.1. A parallel compare-exchange operation. Processes P_i and P_j send their elements to each other. Process P_i keeps $\min\{a_i, a_j\}$, and P_j keeps $\max\{a_i, a_j\}$.

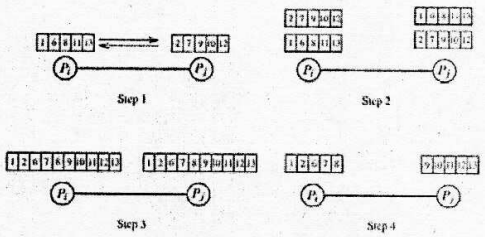


3

چگونگی انجام مقایسه در الگوریتم های موازی

- به ازای هر فرایند بیش از یک عنصر :

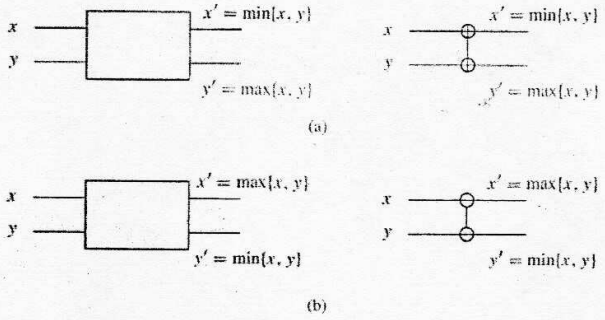
Figure 9.2. A compare-split operation. Each process sends its block of size n/p to the other process. Each process merges the received block with its own block and retains only the appropriate half of the merged block. In this example, process P_i retains the smaller elements and process P_j retains the larger elements.



4

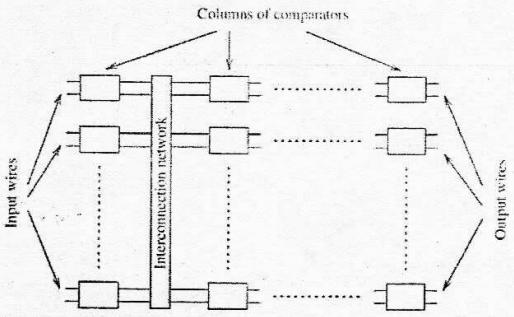
شبکه های مرتب سازی

•جزء اصلی شبکه های مرتب سازی، مقایسه گر (comparator) است.



A schematic representation of comparators: (a) an increasing comparator, and (b) a decreasing comparator.

شبکه های مرتب سازی



A typical sorting network. Every sorting network is made up of a series of columns, and each column contains a number of comparators connected in parallel.

6

مرتب سازی BITONIC

- n عنصر با مرتبه زمانی O((log n)^2) مرتب می شوند.
- دنباله Bitonic: یک دنباله ایی است از عناصر $< a_0, a_1, ..., a_{n-1} >$ با این خصوصیت که یا

(۱) یک اندیس i وجود دارد؛ $0 \le i \le n-1$ بطوریکه $< a_0, a_1, ..., a_i >$ بطور یکنواخت صعودی و $< a_{i+1}, a_{1+2}, ..., a_{n-1} >$ بطور یکنواخت نزولی مرتب هستند، یا

(۲) شیفت چرخشی روی دنباله (۱) باشد.

- مثال: (۱) <1, 2, 4, 7, 6, 0>
- مثال: (۲) <8, 9, 2, 1, 0, 4> ، زیرا شیفت چرخشی <0, 4, 8, 9, 2, 1> است.

مرتب سازی BITONIC

- فرض شود $S = < a_0, a_1, ..., a_{n-1} >$ یک دنباله Bitonic باشد. بنابراین:

$a_0 \le a_1 \le ... \le a_{n/2-1}$ and $a_{n/2} \ge a_{n/2+1} \ge ... \ge a_{n-1}$

- زیردنباله های S بصورت زیر هستند:

$$s_1 = \langle \min\{a_0, a_{n/2}\}, \min\{a_1, a_{n/2+1}\}, \ldots, \min\{a_{n/2-1}, a_{n-1}\} \rangle$$
$$s_2 = \langle \max\{a_0, a_{n/2}\}, \max\{a_1, a_{n/2+1}\}, \ldots, \max\{a_{n/2-1}, a_{n-1}\} \rangle$$

- در زیردنباله S1، عنصر $b_i = \min\{a_i, a_{n/2+i}\}$ وجود دارد بطوریکه عناصر قبل از bi از زیردنباله بخش صعودی و عناصر بعد از آن از بخش نزولی می باشد.
- در زیردنباله S2، عنصر $b_i' = \max\{a_i, a_{n/2+i}\}$ وجود دارد بطوریکه عناصر قبل از bi' از زیردنباله بخش نزولی و عناصر بعد از آن از بخش صعودی می باشد.
- بنابراین زیردنباله های S1 و S2 از نوع دنباله های Bitonic هستند.
- bi بزرگترمساوی همه عناصر S1 و bi' کوچکتر مساوی همه عناصر S2 است و بزرگتر مساوی است از bi.

مرتب سازی BITONIC

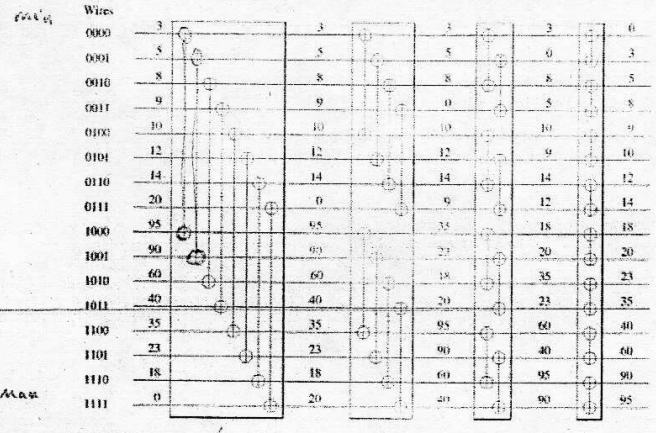
Merging a 16-element bitonic sequence through a series of log 16 bitonic splits.

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

9

مرتب سازی BITONIC

A bitonic merging network for $n = 16$. The input wires are numbered 0, 1 ..., $n - 1$, and the binary representation of these numbers is shown. Each column of comparators is drawn separately; the entire figure represents a $\oplus$BM[16] bitonic merging network. The network takes a bitonic sequence and outputs it in sorted order.

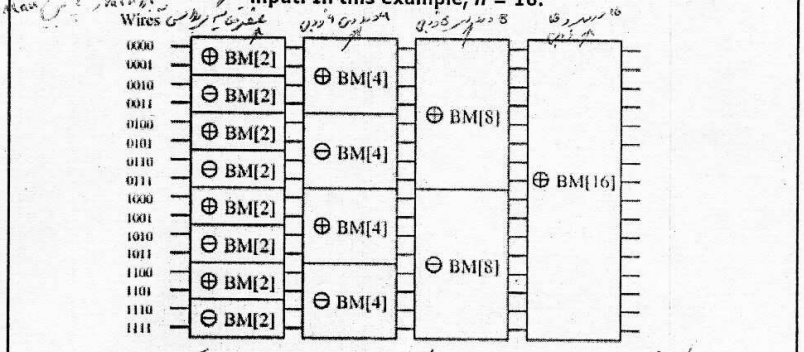


10

2/8/2014

BITONIC مرتب سازی

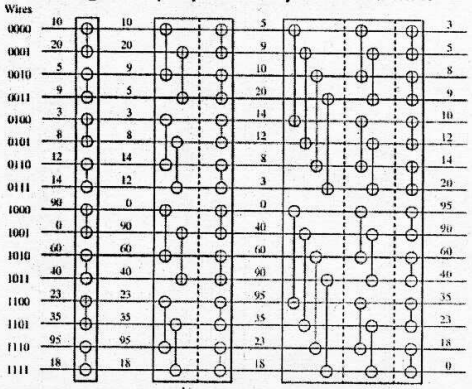
A schematic representation of a network that converts an input sequence into a bitonic sequence. In this example, ⊕BM[k] and ⊖BM[k] denote bitonic merging networks of input size k that use ⊕ and ⊖ comparators, respectively. The last merging network (⊕BM[16]) sorts the input. In this example, $n = 16$.



11

BITONIC مرتب سازی

The comparator network that transforms an input sequence of 16 unordered numbers into a bitonic sequence. In contrast to Slide 10 the columns of comparators in each bitonic merging network are drawn in a single box, separated by a dashed line.

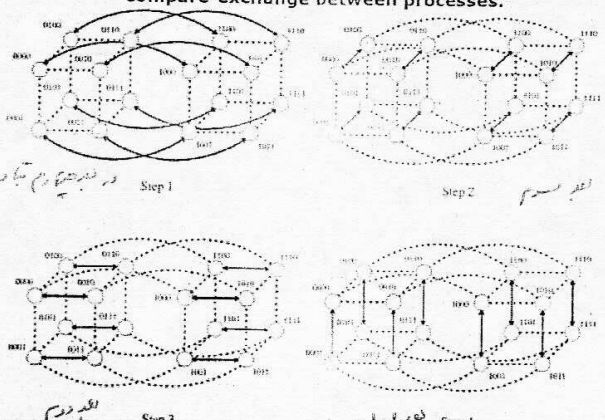


12

2/8/2014

نگاشت مرتب سازی Bitonic روی شبکه Hypercube

Communication during the last stage of bitonic sort. Each wire is mapped to a hypercube process; each connection represents a compare-exchange between processes.



13

نگاشت مرتب سازی Bitonic روی شبکه Hypercube

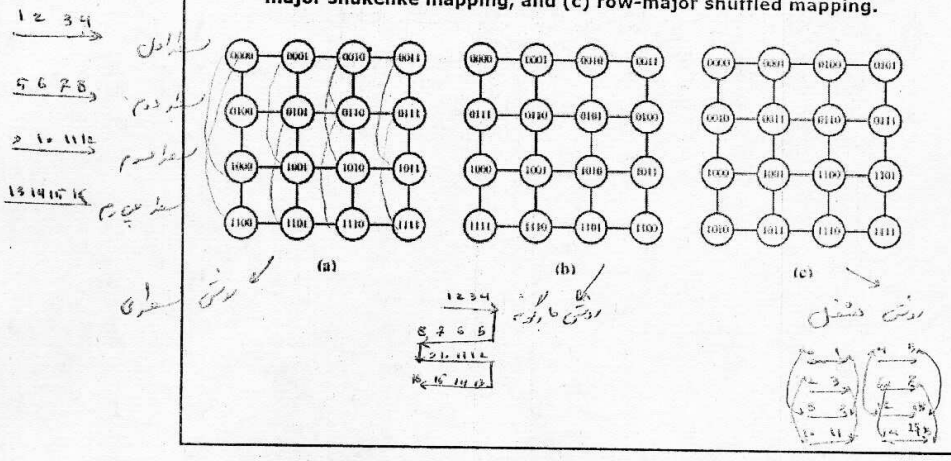
Parallel formulation of bitonic sort on a hypercube with $n = 2^d$ processes. In this algorithm, *label* is the process's label and *d* is the dimension of the hypercube.

```
1.  procedure BITONIC_SORT(label, d)
2.  begin
3.      for i := 0 to d - 1 do
4.          for j := i downto 0 do
5.              if (i + 1)st bit of label ≠ jth bit of label then
6.                  comp_exchange_min(j);
7.              else
8.                  comp_exchange_max(j);
9.  end BITONIC_SORT
```

14

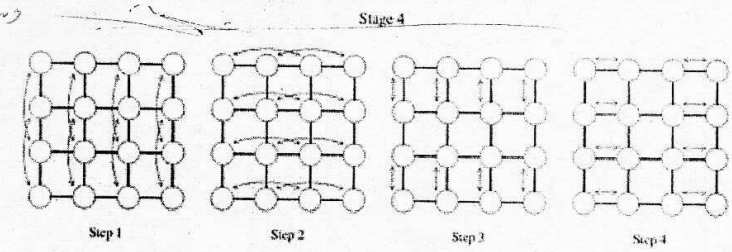
نگاشت مرتب سازی Bitonic روی شبکه Mesh

Different ways of mapping the input wires of the bitonic sorting network to a mesh of processes: (a) row-major mapping, (b) row-major snakelike mapping, and (c) row-major shuffled mapping.



نگاشت مرتب سازی Bitonic روی شبکه Mesh

The last stage of the bitonic sort algorithm for $n = 16$ on a mesh, using the row-major shuffled mapping. During each step, process pairs compare-exchange their elements. Arrows indicate the pairs of processes that perform compare-exchange operations.

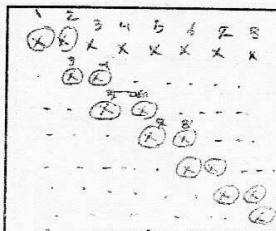


16

پروژه ۴-۱

- شبیه سازی شبکه مش و مکعب و پیاده سازی الگوریتم های مرتب سازی فوق با استفاده از [illegible] در C++ یا JAVA
- توجه: هر نخ مبین یک PE می باشد.

17



۲ مرتب سازی حبابی
Bubble Sort

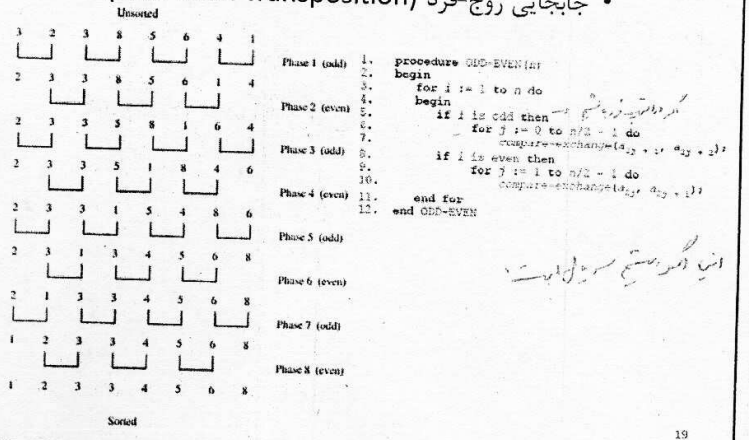
- الگوریتم ترتیبی

```
1.  procedure BUBELE_SORT(n)
2.  begin
3.     for i := n - 1 downto 1 do
4.        for j := 1 to i do
5.           compare-exchange(a[j], a[j+1]);
6.  end BUBBLE_SORT
```

18

مرتب سازی حبابی
Bubble Sort

- موازی سازی مرتب سازی حبابی
- جابجایی زوج-فرد (Odd-Even Transposition)



مرتب سازی حبابی
Bubble Sort

- موازی سازی مرتب سازی حبابی
- موازی سازی جابجایی زوج-فرد
- N تا پردازنده که هر کدام یک عنصر از دنباله را ذخیره کرده اند.

```
1.  procedure ODD-EVEN_PAR (n)
2.  begin
3.     id := process's label
4.     for i := 1 to n do
5.     begin
6.        if i is odd then
7.           if id is odd then
8.              compare-exchange_min(id + 1);
9.           else
10.             compare-exchange_max(id - 1);
11.       if i is even then
12.          if id is even then
13.             compare-exchange_min(id + 1);
14.          else
15.             compare-exchange_max(id - 1);
16.    end for
17. end ODD-EVEN_PAR
```

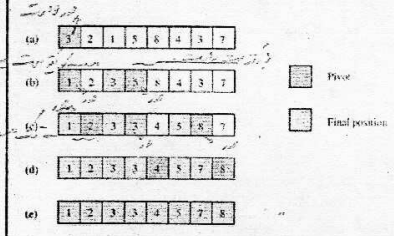
پروژه ۴-۲

- شبیه سازی و پیاده سازی الگوریتم های مرتب سازی فوق با استفاده از نخها در C++ یا JAVA
- توجه: هر نخ مبین یک PE می باشد.

21

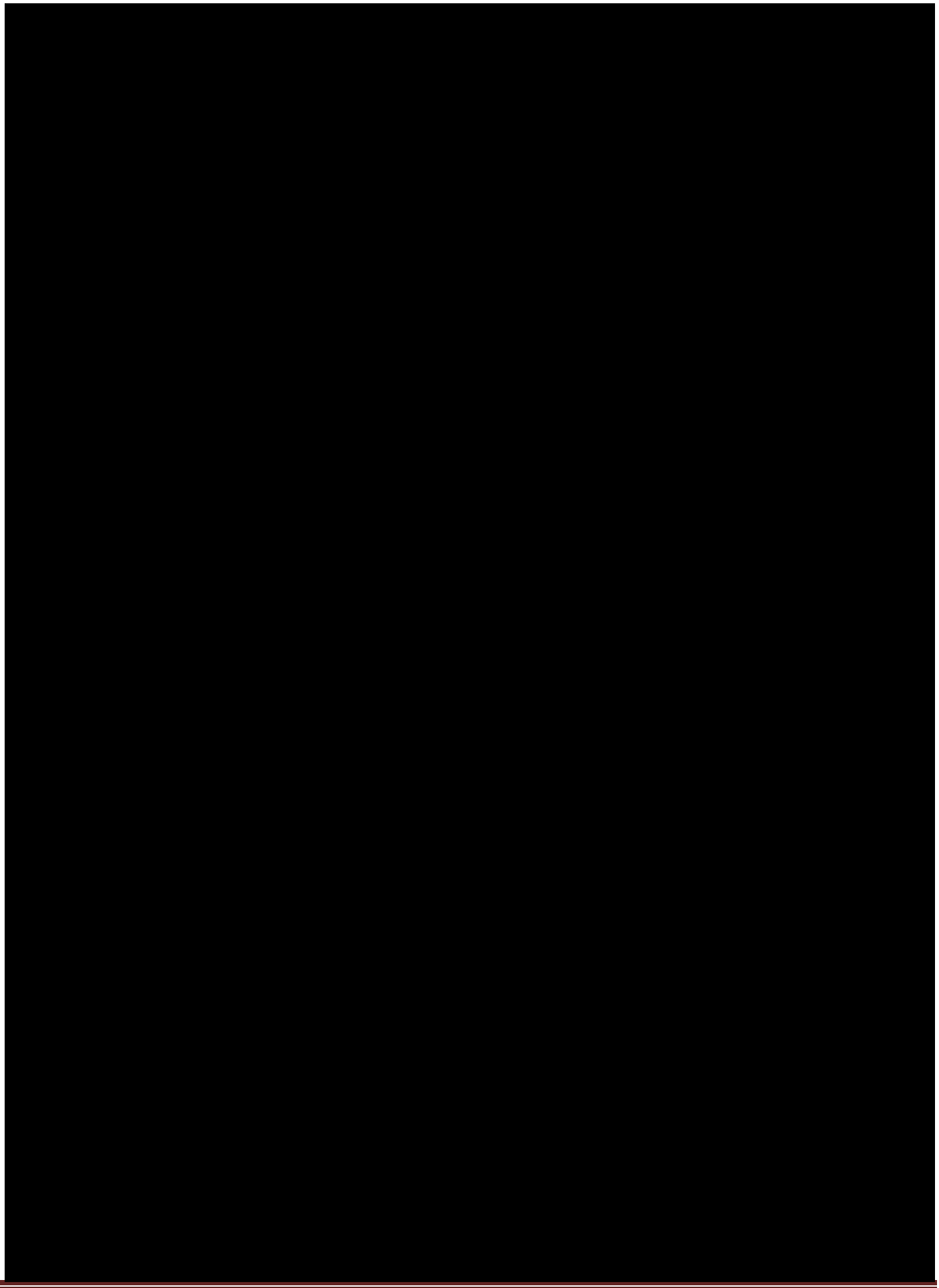
مرتب سازی سریع Quick Sort

Example of the quicksort algorithm sorting a sequence of size $n = 8$.




```
1.  procedure QUICKSORT (A, q, r)
2.  begin
3.     if q < r then
4.     begin
5.        x := A[q];
6.        s := q;
7.        for i := q + 1 to r do
8.           if A[i] ≤ x then
9.           begin
10.             s := s + 1;
11.             swap(A[s], A[i]);
12.          end if
13.       swap(A[q], A[s]);
14.       QUICKSORT (A, q, s);
15.       QUICKSORT (A, s + 1, r);
16.    end if
17. end QUICKSORT
```

22

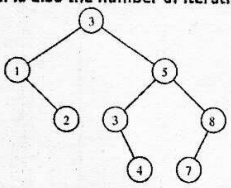


2/8/2014

مرتب سازی سریع
Quick Sort

- موازی سازی QuickSort برای یک CRCW PRAM
- از رویه اختیاری (arbitrary) برای نوشتن همزمان استفاده شده است.
- از یک درخت دودویی برای نمایش اجرای موازی QuickSort استفاده می شود.
- محور در ریشه و عناصر کوچکتر از آن در زیر درخت چپ و عناصر بزرگتر از آن در زیردرخت راست آن قرار دارند.ا پیمایش inorder روی درخت، دنباله مرتب شده بدست می آید.

A binary tree generated by the execution of the quicksort algorithm. Each level of the tree represents a different array-partitioning iteration. If pivot selection is optimal, then the height of the tree is $\Theta(\log n)$, which is also the number of iterations.



23

مرتب سازی سریع
Quick Sort

- موازی سازی QuickSort برای یک CRCW PRAM

The binary tree construction procedure for the CRCW PRAM parallel quicksort formulation.

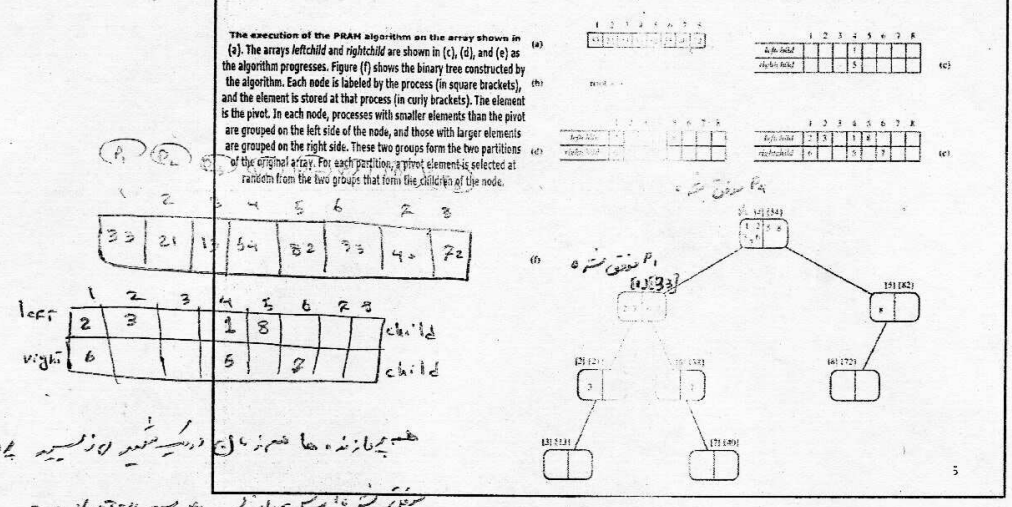
```
1.  procedure BUILD_TREE (A[1...n])
2.  begin
3.     for each process i do
4.     begin
5.        root := i;
6.        parent_i := root;
7.        leftchild[i] := rightchild[i] := n + 1;
8.     end for
9.     repeat for each process i ≠ root do
10.    begin
11.       if (A[i] < A[parent_i]) or
               (A[i]= A[parent_i] and i <parent_i) then
12.       begin
13.          leftchild[parent_i] :=i ;
14.          if i = leftchild[parent_i] then exit
15.          else parent_i := leftchild[parent_i];
16.       end for
```

24

2/8/2014

مرتب سازی سریع
Quick Sort

The execution of the PRAH algorithm on the array shown in (a). The arrays *leftchild* and *rightchild* are shown in (c), (d), and (e) as the algorithm progresses. Figure (f) shows the binary tree constructed by the algorithm. Each node is labeled by the process (in square brackets), and the element is stored at that process (in curly brackets). The element is the pivot. In each node, processes with smaller elements than the pivot are grouped on the left side of the node, and those with larger elements are grouped on the right side. These two groups form the two partitions of the original array. For each partition, a pivot element is selected at random from the two groups that form the children of the node.

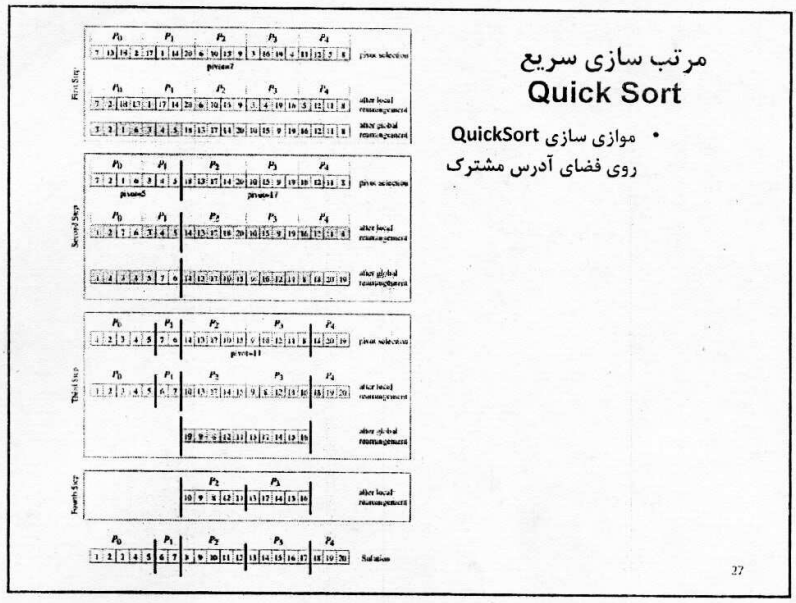


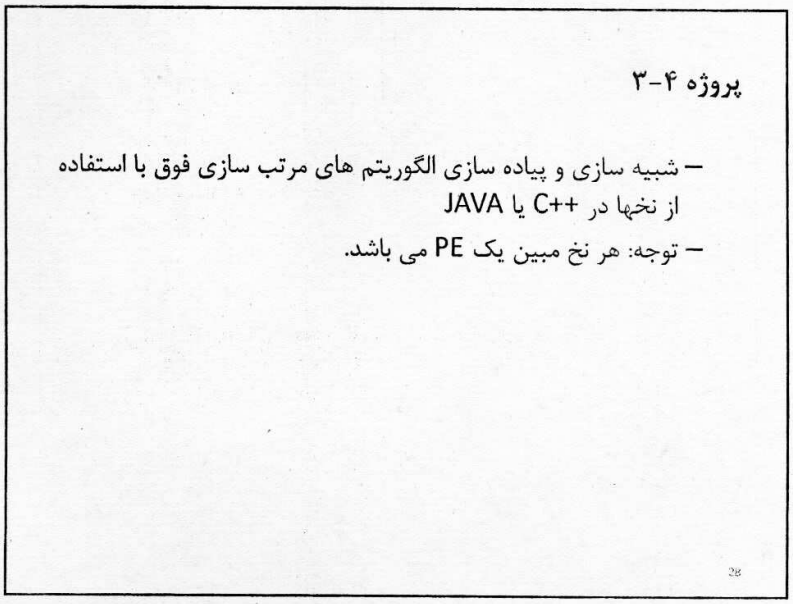
مرتب سازی سریع
Quick Sort

- موازی سازی QuickSort ... فضای آدرس مشترک
- هر فرایند p بلاک ... عنصر از آرایه n عنصری A را شامل می شود و برچسب فرایندها ترتیب ... دنباله مرتب شده را نشان می دهند.
- ابتدا محور تعیین و ... فرایندها ارسال می شود. هر **Pi** به محض دریافت محور، ... به دو زیر بلاک آرایش می دهد. مقادیر بلاک سمت چپ از ... و مقادیر بلاک سمت راست از محور بزرگتر هستند.
- برای هر زیربلاک ... صورت بازگشتی مراحل فوق اعمال می شود.

```
inorder(int i, int A[], int left[], int right[])
if i == 0 return
{ inorder(left[i], A, left, right);
  cout << A[i];
  inorder(right[i], A, left, right)
}
```

26


مرتب سازی سریع
Quick Sort
• موازی سازی QuickSort
روی فضای آدرس مشترک
27


پروژه ۴-۳
– شبیه سازی و پیاده سازی الگوریتم های مرتب سازی فوق با استفاده از نخها در C++ یا JAVA
– توجه: هر نخ مبین یک PE می باشد.
28

مرتب سازی مبتنی بر ایده شمارشی

- مکان هر عنصر ai در لیست مرتب شده را با محاسبه تعداد عناصر کوچکتر از آن تعیین می کند.
- اگر Ci عنصر کوچکتر از ai وجود داشته باشد، مکان ai برابر است با (Ci+1)ام در لیست مرتب شده است.
- اگر ai=aj باشد، اندیس ai بزرگتر در نظر گرفته می شود.
- از مدل CREW PRAM با n^2 پردازنده استفاده می شود.
- n^2 پردازنده در n ردیف که هر ردیف n عنصر دارد، مرتب می شوند.شماره گذاری پردازنده ها P(i,j)

29

مراحل الگوریتم مرتب سازی شمارشی

- هر ردیف i از پردازنده ها A[i] عناصر کوچکتر از A[i] می باشد، محاسبه می کند. پردازنده (i,j) A[i] و A[j] را مقایسه و C[i] را بهنگام می کند.
- اولین پردازنده در ردیف (i,1) عنصر A[i] را در مکان مناسبش در لیست مرتب شده قرار می دهد: C[i]+1

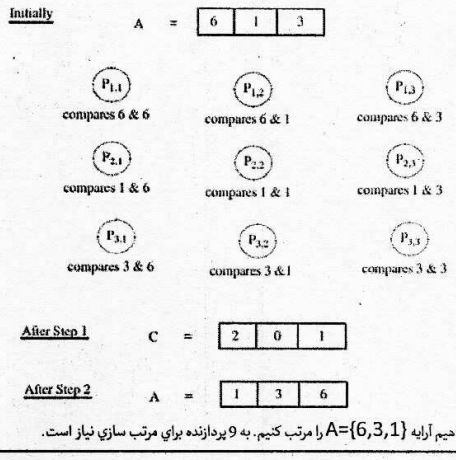
```
Algorithm Sort_CRCW

/* Step 1 */
forall Pi,j, where 1 ≤ i, j ≤ n do in parallel
  if A[i] > A[j] or (A[i]=A[j] and i > j) then
    C[i] ← 1
  else
    C[i] ← 0
  endif
endfor

/* Step 2 */
forall Pi,1, where 1 ≤ i ≤ n do in parallel
  A[C[i] + 1] ← A[i]
endfor
```

30

مثالی از الگوریتم مرتب سازی شمارشی



میخواهیم آرایه A={6,3,1} را مرتب کنیم. به 9 پردازنده برای مرتب سازی نیاز است.

31

پیچیدگی الگوریتم مرتب سازی شمارشی

- زمان اجرا: T(n)=O(n)
- تعداد پردازنده ها: P(n)=n^2
- هزینه: C(n)=O(n^3)

- **هزینه بهینه است؟**

32

2/8/2014

پروژه ۴-۴

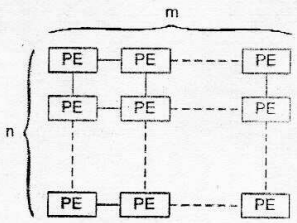
– مرتب سازی بردار ۱۶ عنصری با الگوریتم مرتب سازی مبتنی بر شمارشی با استفاده از C یا JAVA

– توجه: هر نخ مبین یک PE می باشد.

33

اجرای الگوریتم های مرتب سازی بر روی شبکه مش

– شبکه مش n*m



– پیچیدگی: ...

– Tr: زمان مسیریابی برای ارسال یک داده از یک PE به یکی از PEهای همسایه.

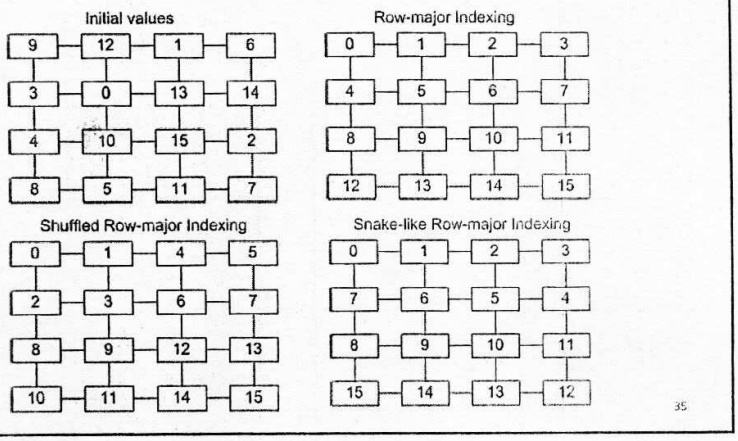
– Tc: زمان مقایسه در ... (با N=n^2 مقایسه همزمان اجرا

34

17

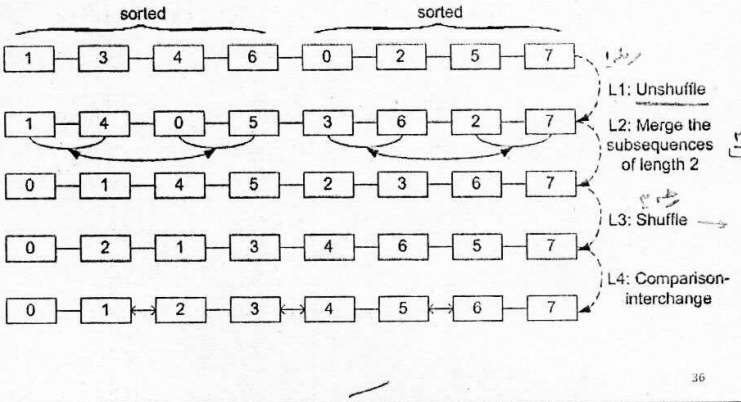
2/8/2014

شماهای مختلف شاخص مرتب سازی
(Sorting Indexing Schemes)



الگوریتم Batchers odd-even sort

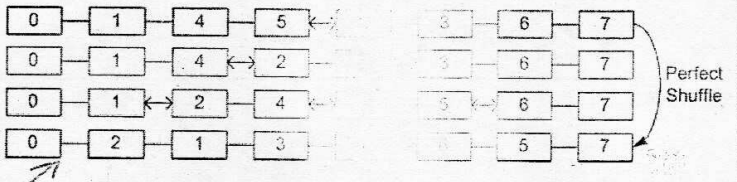
- مثال:



2/8/2014

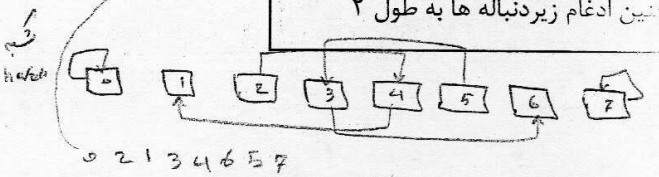
الگوریتم Batcher odd-even sort

- نحوه انجام یک Shuffle روی یک آرایه خطی (با استفاده از یک رشته از عملیات جا...



- تمرین:
  - نحوه انجام Unshuffle ... چنین ادغام زیردنباله ها به طول ۲ ... روی آرایه خطی؟

37



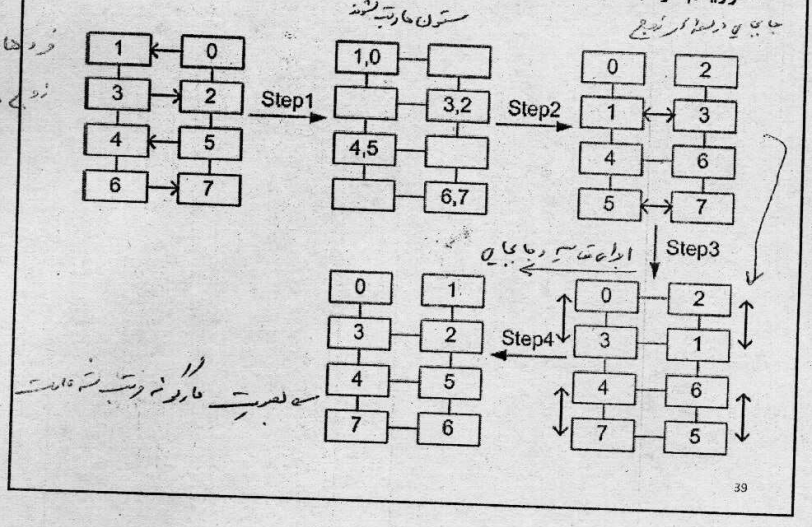
0 2 1 3 4 6 5 7

الگوریتم مرتب سازی Kung & Leiserson بر اساس روش Batcher

- M(j, k) یک الگوریتم ... برای مرتب کردن دو زیر آرایه j.k/2 (j و k هر دو ... و k>1).
- الگوریتم بصورت major ... Snake like است.
- در حالت خاص، (1, 2) ... مقایسه و جابجایی کافی است.
- اگر دو ستون مرتب شده ... (j≥2) : الگوریتم M(j, 2) شامل گامهای:
  1. تمام ردیفهای فرد خود را ستون سمت چپ و تمام ردیفهای زوج را به ستون راست انتقال دهد (زمان= Tc).
  2. با استفاده از مرتب سازی ... ردیف زوج و فرد، هر ستون مرتب شود (زمان= 2Tk+jTc).
  3. جابجایی در ... (زمان= 2Tk).

19

الگوریتم مرتب سازی Kung & Thompson بر اساس روش Batcher



39

الگوریتم مرتب سازی Kung & Thompson بر اساس روش Batcher

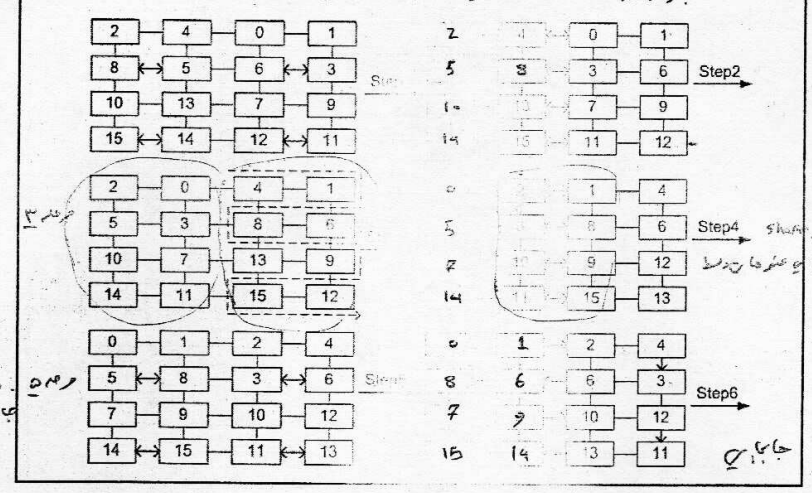
- اجرای الگوریتم مرتب سازی M(j, k) برای j>2 و k>2.

1. اگر j>2 است، تنها یک گام جابجایی روی سطرهای زوج انجام بده.
   اگر j=2 باشد، ستونها قبلا جدا شده اند (زمان= 2Tk).
2. برای هر سطر Unshuffle صورت گیرد (زمان= (k-1).Tk).
3. عمل ادغام با خواندن M(j, k/2) روی هر نیمه آرایه (زمان= T(j, k/2)).
4. برای هر سطر Shuffle صورت گیرد (زمان= (k-2).Tk).
5. در سطرهای زوج جابجایی انجام بده (زمان= (k-2).Tk).
6. مقایسه و جابجایی المانهای مجاور هم (هر زوج با فرد بعدی) (زمان= 4(Tk+Tc)).

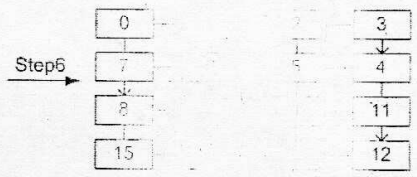
40

2/8/2014

الگوریتم مرتب سازی Kung & Leiserson بر اساس روش Batcher



الگوریتم مرتب سازی Kung & Leiserson بر اساس روش Batcher



پروژه ۴-۵

- شبیه سازی شبکه مش و پیاده سازی الگوریتم های مرتب سازی فوق با استفاده از نخها در C++ یا JAVA
- توجه: هر نخ مبین یک PE می باشد.

43

فصل ششم

الگوریتم جمع چند جمله ایی

1

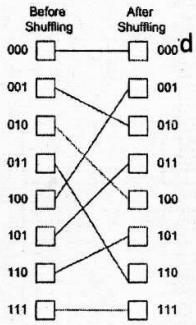
شبکه هاي ایستا

- شبکه Shuffle-Exchange:

- توابع:

$$S(p_{m-1}p_{m-2}\cdots p_1p_0) = p_{m-2}p_{m-3}\cdots p_1p_0p_{m-1}$$

$$E(p_{m-1}p_{m-2}\cdots p_1p_0) = p_{m-1}p_{m-2}\cdots p_1\overline{p_0}$$



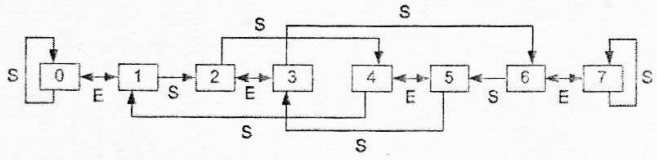
- Shuffle مشابه با بر زدن کارتها است (d Shuffling)

2

2/8/201

شبکه هاي ايستا

- شبکه Shuffle-Exchange:



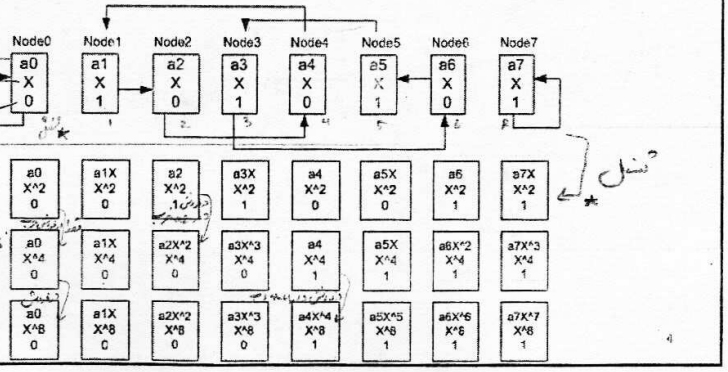
- مثال: محاسبه چند جمله اي: $a_{N-1}X^{N-1} + a_{N-2}X^{N-2} + ... + a_1X + a_0$
  - هر پردازنده داراي ثبات ضريب (ai) ، متغير X و همچنين بيت ماسک (m=0 يا m=1).
  - اجراي عمليات در دو فاز.
  - فاز اول محاسبه a_iX^i در هر گره i.
  - فاز دوم جمع همه ترم ها براي محاسبه نتيجه نهايي.

3

شبکه هاي ايستا

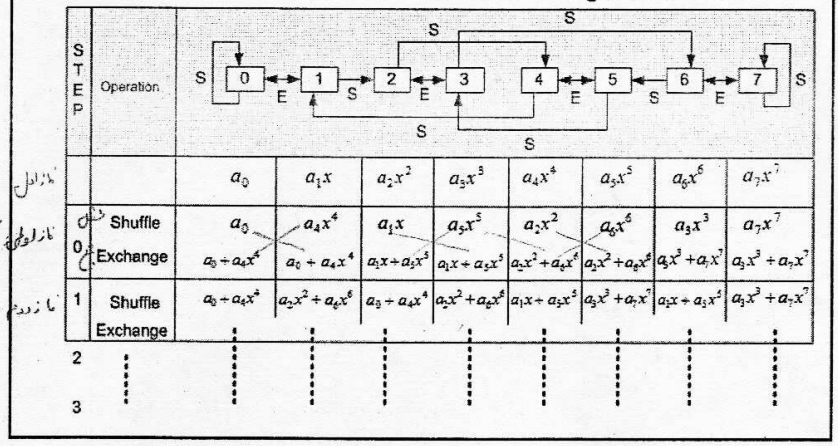
- عمليات فاز اول:
  - ضريب در متغير ضرب مي شود.
  - متغير در خودش ضرب مي شود.
  - بيتهاي ماسک Shuffle مي شود.

اگر m=0 باشد: فقط متغير در خودش ضرب مي شود.
اگر m=1 باشد: متغير در خودش و در ضريب ضرب مي شود.



شبکه هاي ايستا

- عمليات فاز دوم: جمع با استفاده از Shuffle & Exchange



پروژه 2-2

- محاسبه يک چند جمله اي با الگوريتم فوق با استفاده از نخها در C++ يا JAVA
- توجه: هر نخ مبين يک PE مي باشد.