中国科学技术大学 计算机科学与技术系
University of Science and Technology of China
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

并行计算
Parallel Computing

主讲人 孙广中

Spring, 2022

第二篇 并行算法的设计

第五章 并行算法与并行计算模型

第六章 并行算法基本设计策略

第七章 并行算法常用设计技术

第八章 并行算法一般设计过程

第七章 并行算法常用设计技术

- 7.1 划分设计技术
  - 7.1.1 均匀划分技术
  - 7.1.2 方根划分技术
  - 7.1.3 对数划分技术
  - 7.1.4 功能划分技术
- 7.2 分治设计技术
- 7.3 平衡树设计技术
- 7.4 倍增设计技术
- 7.5 流水线设计技术

均匀划分技术（1）

- 划分方法

  n个元素A[1..n]分成p组，每组A[(i-1)n/p+1..in/p]，i=1~p

- 示例：MIMD-SM模型上的PSRS排序

begin

(1)均匀划分：将n个元素A[1..n]均匀划分成p段，每个p_i处理 A[(i-1)n/p+1..in/p]

(2)局部排序：p_i调用串行排序算法对A[(i-1)n/p+1..in/p]排序

(3)选取样本：p_i从其有序子序列A[(i-1)n/p+1..in/p]中选取p个样本元素

(4)样本排序：用一台处理器对p^2个样本元素进行串行排序

(5)选择主元：用一台处理器从排好序的样本序列中选取p-1个主元，并播送给其他p_i

(6)主元划分：p_i按主元将有序段A[(i-1)n/p+1..in/p]划分成p段

(7)全局交换：各处理器将其有序段按段号交换到对应的处理器中

(8)归并排序：各处理器对接收到的元素进行归并排序

end.

均匀划分技术（2）

- 例7.1 PSRS排序过程。N=27，p=3，PSRS排序如下：

(a) 均匀划分：	15	46	48	93	39	6	72	91	14	36	69	40	89	61	97	12	21	54	53	97	84	58	32	27	33	72	20
(b) 局部排序：	6	14	15	39	46	48	72	91	93	12	21	36	40	54	61	69	89	97	20	27	32	33	53	58	72	84	97
(c) 正则采样：	6	39	72	12	40	69	20	33	72																		
(d) 采样排序：	6	12	20	33	39	40	69	72	72																		
(e) 选择主元：	33	69																									
(f) 主元划分：	6	14	15	39	46	48	72	91	93	12	21	36	40	54	61	69	89	97	20	27	32	33	53	58	72	84	97
(g) 全局交换：	6	14	15	12	21	20	27	32	33	39	46	48	36	40	54	61	69	53	58	72	91	93	89	97	72	84	97
(h) 归并排序：	6	12	14	15	20	21	27	32	33	36	39	40	46	48	53	54	58	61	69	72	72	84	89	91	93	97	97

第七章 并行算法常用设计技术

7.1 划分设计技术
7.1.1 均匀划分技术
7.1.2 方根划分技术
7.1.3 对数划分技术
7.1.4 功能划分技术
7.2 分治设计技术
7.3 平衡树设计技术
7.4 倍增设计技术
7.5 流水线设计技术

方根划分技术（1）

- 划分方法

  n个元素A[1..n]分成A[(i-1)n^(1/2)+1..in^(1/2)]，i=1~n^(1/2)

- 示例： SIMD-CREW模型上的 $k=\lfloor\sqrt{pq}\rfloor$ Valiant归并(1975年发表)

//有序组A[1..p]、B[1..q], (假设p<=q), 处理器数 $k=\lfloor\sqrt{pq}\rfloor$

begin

(1)方根划分: A,B分别按 $i\lfloor\sqrt{p}\rfloor$ 和 $j\lfloor\sqrt{q}\rfloor$ 分成若干段（$i=1\sim\lfloor\sqrt{p}\rfloor$, $j=1\sim\lfloor\sqrt{q}\rfloor$）;

(2)段间比较: A划分元与B划分元比较(至多 $\lfloor\sqrt{p}\rfloor\cdot\lfloor\sqrt{q}\rfloor$ 对),
确定A划分元应插入B中的区段;

(3)段内比较: A划分元与B相应段内元素进行比较，并插入适当的位置;

(4)递归归并: B按插入的A划分元重新分段，与A相应段(A除去原划分元)构成了成对的段组，对每对段组递归执行(1)~(3)，直至A组为0时，递归结束; 各组仍按 $k=\lfloor\sqrt{pq}\rfloor$ 分配处理器;

end.

方根划分技术（2）

- 示例: A={1,3,8,9,11,13,15,16},p=8; B={2,4,5,6,7,10,12,14,17},q=9

(1)(2)
A: 1 3 8* 9 11 13* 15 16 $(p=8, \lceil\sqrt{p}\rceil=3, \lfloor\sqrt{p}\rfloor=2)$
B: 2 4 5* 6 7 10* 12 14 17* $(q=9, \lceil\sqrt{q}\rceil=3, \lfloor\sqrt{q}\rfloor=3)$

⇩

(3)
A: 1 3 8* 9 11 13* 15 16
B: 2 4 5* 6 7 10* 12 14 17*

⇩

A_1: 1 3 (p_1=2) | A_2: 9 11 (p_2=2) | A_2: 15 16
B_1: 2 4 5 6 7 8(q_1=6) | B_2: 10 12 13(q_2=3) | B_3: 14 17

⇩

A_1: 1 3* (p_1=2) | ...... | ......
B_1: 2 4 5* 6 7 8*(q_1=6) | ...... | ......

⇩

A_{11}: 1* | A_{11}: | ...... | ......
B_{11}: 2 3* | B_{12}: 4 5 6 7 8 | ...... | ......

⇩

A:
B: 1 2 3 | 4 5 6 7 8 | 9 10 11 | 12 13 | 14 15 16 | 17

方根划分技术（3）

- 算法分析

(1)算法在并行递归过程中所需的处理器数≤ $k=\lfloor\sqrt{pq}\rfloor$

段间比较: $\lfloor\sqrt{p}\rfloor\cdot\lfloor\sqrt{q}\rfloor$ 比较对数≤ $\lfloor\sqrt{pq}\rfloor=k$；

段内比较: $\lfloor\sqrt{p}\rfloor\cdot(\lfloor\sqrt{q}\rfloor-1)\le\lfloor\sqrt{pq}\rfloor=k$

递归调用: 设 **A,B** 分成若干子段对为**(p_1,q_1), (p_2,q_2),……**

则Σ**p_i**≤**p**, Σ**q_i**≤**q**, 由 Cauchy 不等式=>

$$\sum\lfloor\sqrt{p_iq_i}\rfloor\le\lfloor\sum\sqrt{p_iq_i}\rfloor\le\lfloor\sqrt{\sum p_i\sum q_i}\rfloor\le\lfloor\sqrt{pq}\rfloor=k$$

综上，整个过程可用处理器数 $k=\lfloor\sqrt{pq}\rfloor$ 完成。

(2)时间分析

记 λ$_i$ 是第 **i** 次递归后的 **A** 组最大长度，=> $\lambda_0=p$，$\lambda_i\le\lfloor\sqrt{\lambda_{i-1}}\rfloor\le\cdots\le\lfloor p^{2^{-i}}\rfloor$

算法在 $\lambda_i=\text{常数}C$ 时终止递归, 即 $\text{常数}C\le p^{2^{-i}}\le\text{常数}C+1$ => $i\le\log\log p+\text{常数}C_1$

由**(1)**知算法中其他各步的时间为 **O(1)**，所以 **Valiant** 归并算法时间

$$t_k(p,q)=O(\log\log p)\qquad p\le q$$

第七章 并行算法常用设计技术

- 7.1 划分设计技术
  - 7.1.1 均匀划分技术
  - 7.1.2 方根划分技术
  - 7.1.3 对数划分技术
  - 7.1.4 功能划分技术
- 7.2 分治设计技术
- 7.3 平衡树设计技术
- 7.4 倍增设计技术
- 7.5 流水线设计技术

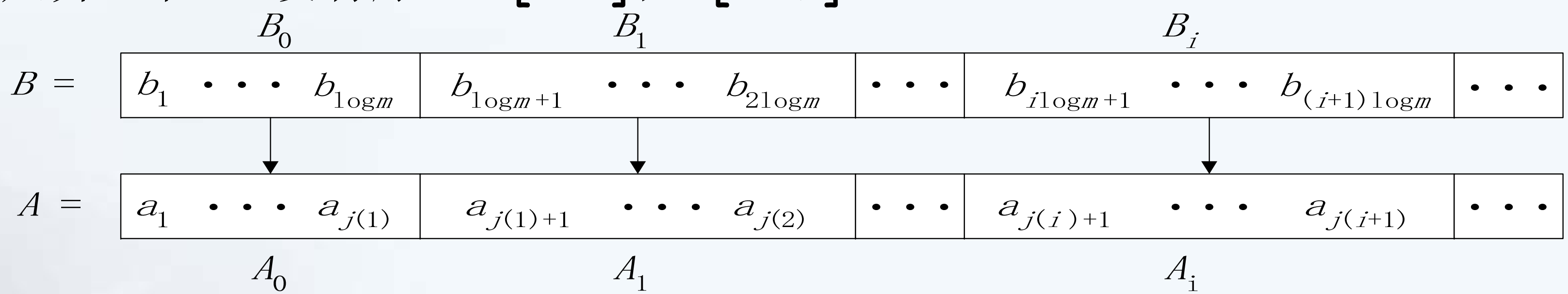
对数划分技术

- 划分方法

  n个元素A[1..n]分成A[(i-1)logn+1..ilogn]，i=1~n/logn

- 示例：PRAM-CREW上的对数划分并行归并排序

(1)归并过程：设有序组A[1..n]和B[1..m]



j[i]=rank(b_{ilogm}:A)为b_{ilogm}在A中的位序，即A中小于等于b_{ilogm}的元素个数

(2)例：A=(4,6,7,10,12,15,18,20), B=(3,9,16,21) n=8, m=4

=>logm=log4=2

=> j[1]=rank(b_{logm}:A)=rank(b_2:A)=rank(9:A)=3, j[2]=…=8

B_0: 3， 9　　B_1: 16， 21

A_0: 4， 6， 7　　A_1: 10， 12， 15， 18， 20

A和B归并化为(A_0，B_0)和(A_1，B_1)的归并

第七章 并行算法常用设计技术

- 7.1 划分设计技术
  - 7.1.1 均匀划分技术
  - 7.1.2 方根划分技术
  - 7.1.3 对数划分技术
  - 7.1.4 功能划分技术
- 7.2 分治设计技术
- 7.3 平衡树设计技术
- 7.4 倍增设计技术
- 7.5 流水线设计技术

功能划分技术（1）

- 划分方法

  n个元素A[1..n]分成等长的p组，每组满足某种特性。

- 示例：(m, n)选择问题(求出n个元素中前m个最小者)
  - 功能划分：要求每组元素个数必须大于m;
  - 算法：p194算法7.4

  输入：$A=(a_1,\ldots,a_n)$; 输出：前m个最小者；

  Begin

  (1) 功能划分：将A划分成g=n/m组，每组含m个元素；

  (2) 局部排序：使用Batcher排序网络将各组并行进行排序；

  (3) 两两比较：将所排序的各组两两进行比较，从而形成MIN序列；

  (4) 排序-比较：对各个MIN序列，重复执行第(2)和第(3)步，直至选出m个最小者。

  End

功能划分技术（2）

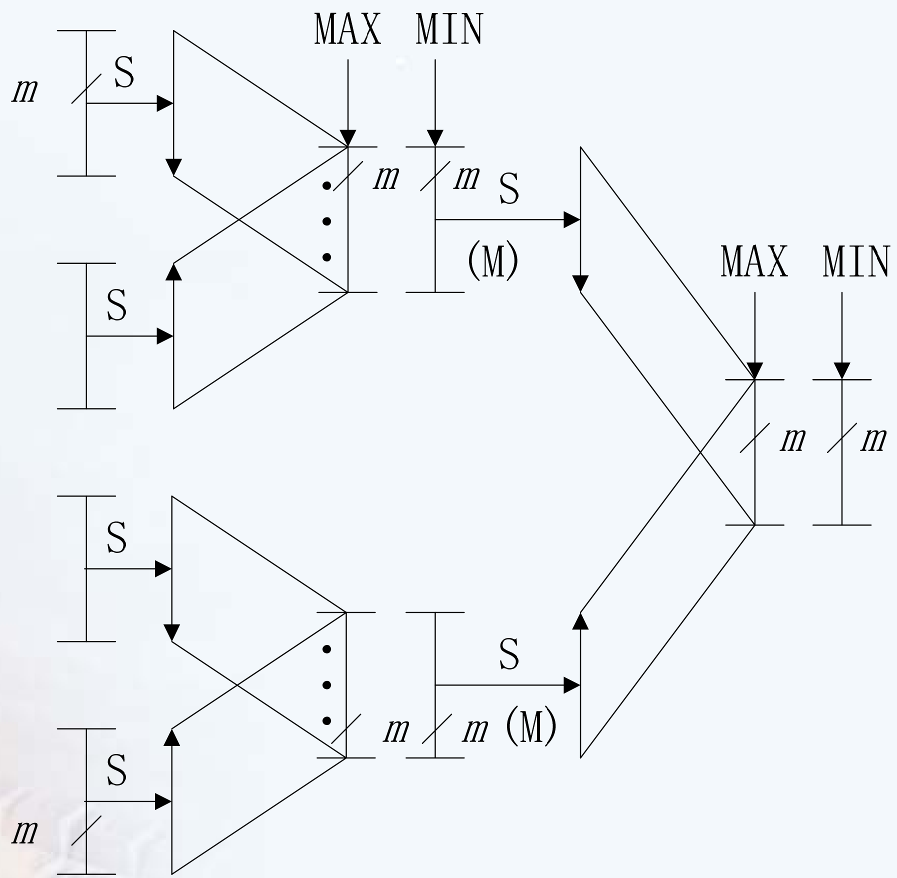


图6.3 (m-n)-选择过程

第七章 并行算法常用设计技术

- 7.1 划分设计技术
- 7.2 分治设计技术
  - 7.2.1 并行分治设计步骤
  - 7.2.2 双调归并网络
- 7.3 平衡树设计技术
- 7.4 倍增设计技术
- 7.5 流水线设计技术

并行分治设计步骤

- 将输入划分成若干个规模相等的子问题；
- 同时(并行地)递归求解这些子问题；
- 并行地归并子问题的解，直至得到原问题的解。

第七章 并行算法常用设计技术

- 7.1 划分设计技术
- 7.2 分治设计技术
  - 7.2.1 并行分治设计步骤
  - 7.2.2 双调归并网络
- 7.3 平衡树设计技术
- 7.4 倍增设计技术
- 7.5 流水线设计技术

双调归并网络（1）

- 双调序列(p195定义7.2)
  (1,3,5,7,8,6,4,2,0)
  (8,7,6,4,2,0,1,3,5)
  (1,2,3,4,5,6,7,8)
  以上都是双调序列
- Batcher定理

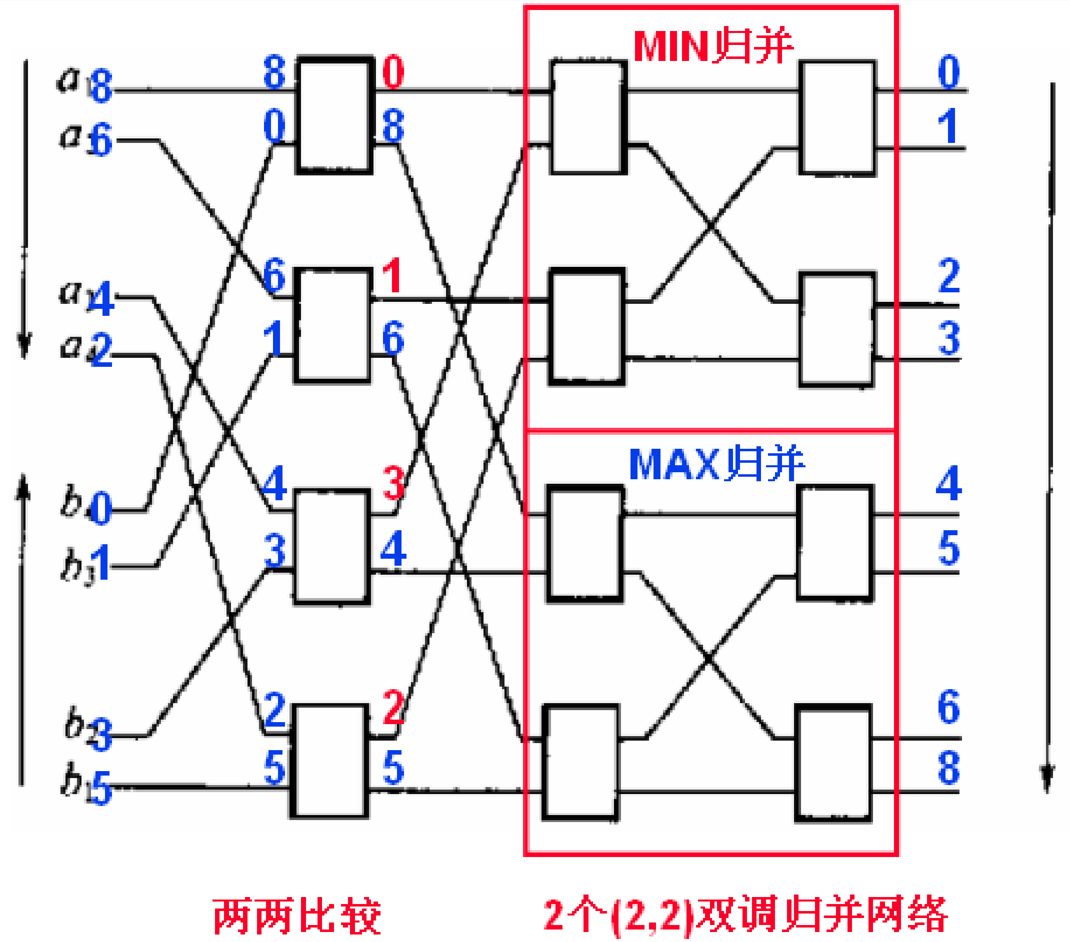
  给定双调序列$(x_0,x_1,\dots,x_{n-1})$, 对于$s_i=\min\{x_i,x_{i+n/2}\}$和$l_i=\max\{x_i,x_{i+n/2}\}$，则小数序列$(s_0,s_1,\dots,s_{n-1})$和大数序列$(l_0,l_1,\dots,l_{n-1})$有，

  (1) $b_i \leqslant c_j \quad (1\le i,\ j\le n)$

  (2)小数和大数序列仍是双调的

双调归并网络（2）

- (4, 4) 双调归并网络



双调归并网络（3）

- Batcher双调归并算法

输入：双调序列$X=(x_0,x_1,\ldots,x_{n-1})$

输出：非降有序序列$Y=(y_0,y_1,\ldots,y_{n-1})$

```
Procedure BITONIC_MERG(x)
Begin
  (1)for i=0 to n/2-1 par-do
     (1.1) s_i=min{x_i,x_{i+n/2}}
     (1.2) l_i=max{x_i,x_{i+n/2}}
     end for
  (2)Recursive Call:
     (2.1)BITONIC_MERG(MIN=(s_0,...,s_{n/2-1}))
     (2.2)BITONIC_MERG(MIN=(l_0,..., l_{n/2-1}))
  (3)output sequence MIN followed by sequence MAX
End
```

第七章 并行算法常用设计技术

7.1 划分设计技术
7.2 分治设计技术
7.3 平衡树设计技术
  7.3.1 设计思想
  7.3.2 求最大值
  7.3.3 计算前缀和
7.4 倍增设计技术
7.5 流水线设计技术

平衡树设计技术

- 设计思想

  以树的叶结点为输入，中间结点为处理结点，由叶向根或由根向叶逐层进行并行处理。

- 示例
  - 求最大值
  - 计算前缀和

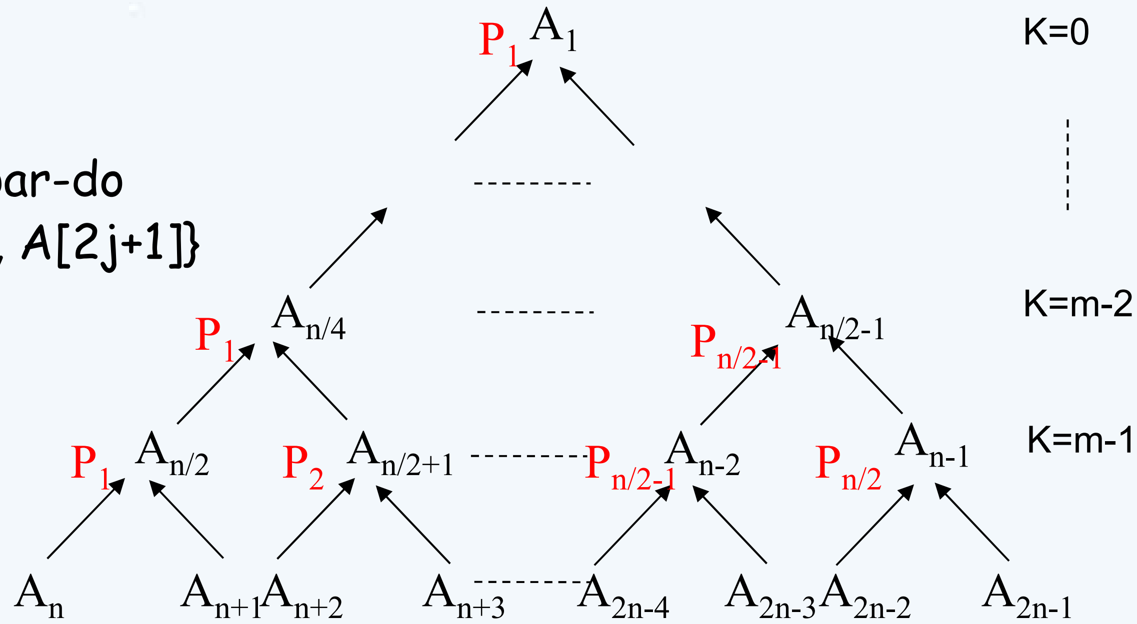
第七章 并行算法常用设计技术

7.1 划分设计技术
7.2 分治设计技术
7.3 平衡树设计技术
7.3.1 设计思想
7.3.2 求最大值
7.3.3 计算前缀和
7.4 倍增设计技术
7.5 流水线设计技术

求最大值（1）

- 算法7.8：SIMD-TC(SM)上求最大值算法

```
Begin
  for k=m-1 to 0 do
     for j=2^k to 2^(k+1)-1 par-do
        A[j]=max{A[2j], A[2j+1]}
     end for
  end for
end
```



- 图示
- 时间分析

  $t(n)=m\times O(1)=O(\log n)$

  $p(n)=n/2$

求最大值（2）

- SIMD-CRCW上常数时间求最大值算法

算法 SIMD-CRCW上枚举算法
//输入A[1..p]，p个不同元素
//B[1..p][1..p],M[1..p]为中间处理用的布尔数组, 如果M[i]＝1, 则A[i]为最大值
begin
  (1)for 1≤i, j≤p par-do //工作量O(p2); 时间O(1),因为允许同时读
      if A[i]≥A[j] then B[i, j]=1 else B[i, j]=0
      end if
    end for
  (2)for 1≤i≤p par-do   //工作量O(p2); 时间O(1),因为允许同时写
      M[i]=B[i,1]∧ B[i,2]∧… ∧B[i,p]
    end for
end

- **T(n)=O(1)**
- **W(n)=O(p^2)**
- 可以用**p^2**个处理器实现
- 速度虽快，但不是**WT**最优

第七章 并行算法常用设计技术

7.1 划分设计技术

7.2 分治设计技术

7.3 平衡树设计技术

7.3.1 设计思想

7.3.2 求最大值

7.3.3 计算前缀和

7.4 倍增设计技术

7.5 流水线设计技术

计算前缀和（1）

- 问题定义

  n个元素{$x_1,x_2,\ldots,x_n$}，前缀和是n个部分和：

  $S_i=x_1*x_2*\ldots*x_i$，$1\le i\le n$　这里*可以是＋或×

- 串行算法：　$S_i=S_{i-1}*x_i$　　计算时间为 O(n)
- 并行算法：p199算法7.9 SIMD-TC上非递归算法

  令$A[i]=x_i$, $i=1\sim n$,

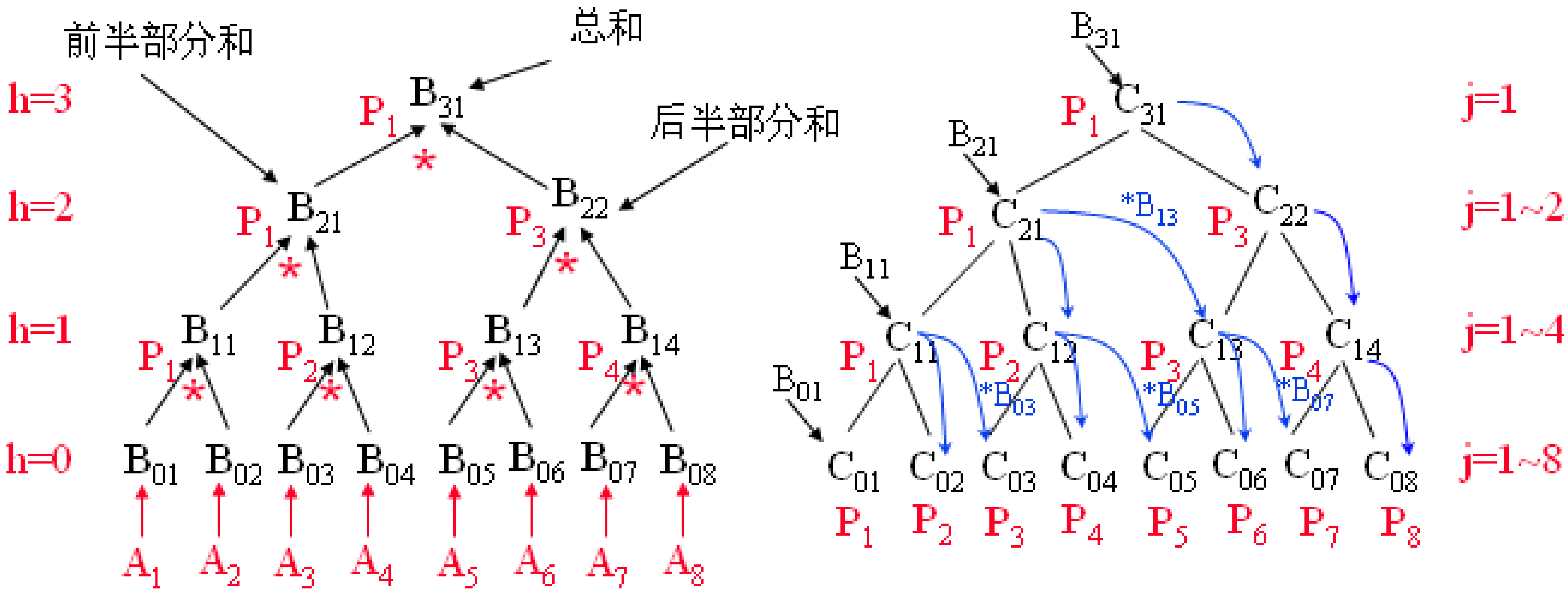
  B[h,j]和C[h,j]为辅助数组($h=0\sim\log n$, $j=1\sim n/2^h$)

  数组B记录由叶到根正向遍历树中各结点的信息(求和)

  数组C记录由根到叶反向遍历树中各结点的信息(播送前缀和)

计算前缀和（2）

- 例： n=8, p=8, C_{01}~C_{08}为前缀和



计算：B[h,j]=B[h-1,2j-1]*B[h-1,2j]

正向遍历(求和)

计算：C[h,1]=B[h,1] j=1

C[h,j]=C[h+1, j/2] j=even

C[h,j]=C[h+1,(j-1)/2]*B[h, j] j=odd>1

反向遍历(播送前缀和)

计算前缀和（3）

- SIMD-SM上非递归算法

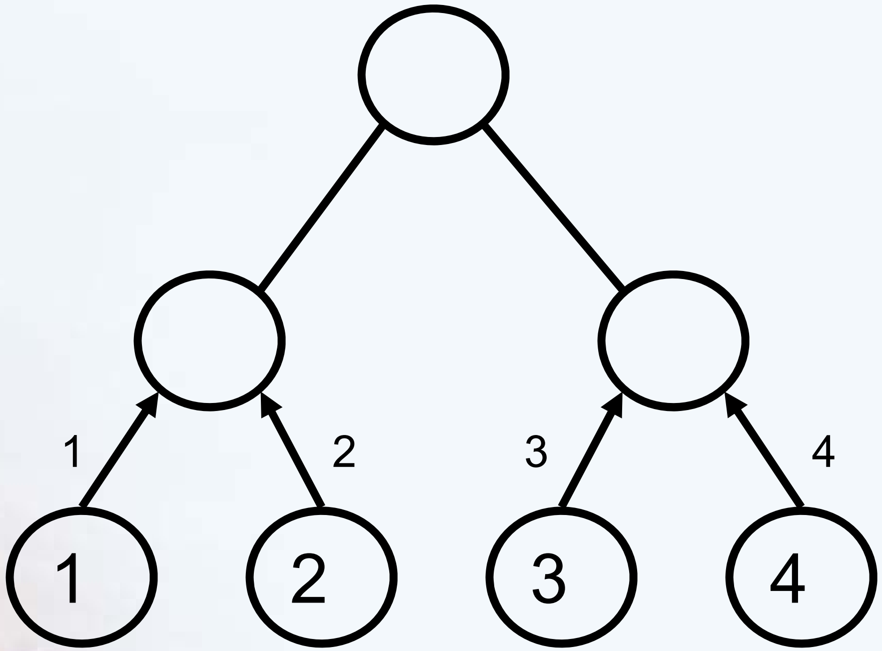
```
begin
  (1)for j=1 to n par-do    //初始化
       B[0,j]=A[j]
     end if
  (2)for h=1 to logn do   //正向遍历
        for j=1 to n/2h par-do
           B[h,j]=B[h-1,2j-1]*B[h-1,2j]
        end for
     end for
  (3)for h=logn to 0 do   //反向遍历
        for j=1 to n/2^h par-do
          (i)  if j=even then   //该结点为其父结点的右儿子
                  C[h,j]=C[h+1,j/2]
               end if
          (ii) if j=1 then       //该结点为最左结点
                  C[h,1]=B[h,1]
               end if
          (iii) if j=odd>1 then  //该结点为其父结点的左儿子
                  C[h,j]=C[h+1,(j-1)/2]*B[h,j]
                end if
        end for
     end for
end
```

时间分析:

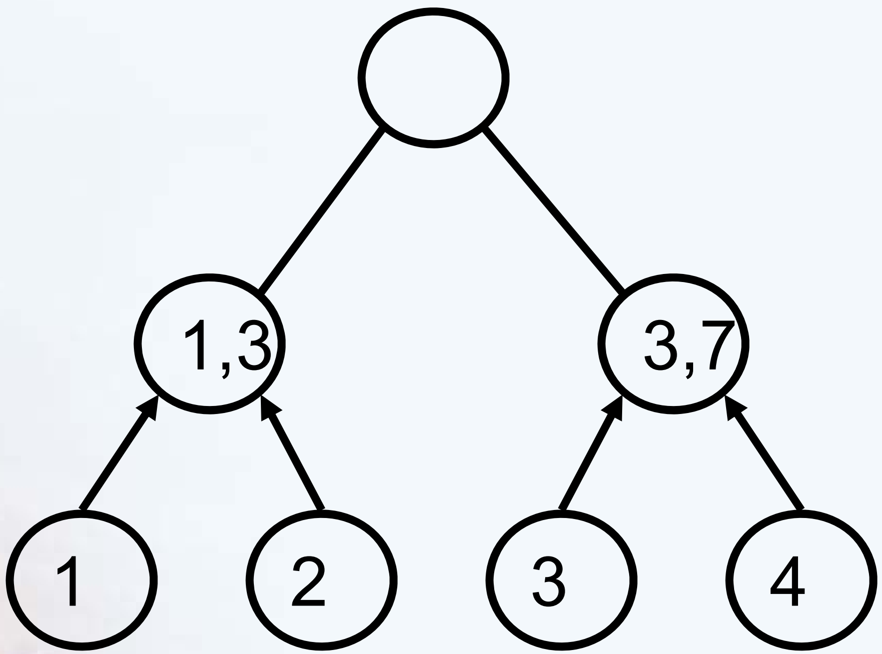
(1) O(1) (2) O(logn) (3) O(logn)

==> t(n)=O(logn)， p(n)=n， c(n)=O(nlogn)

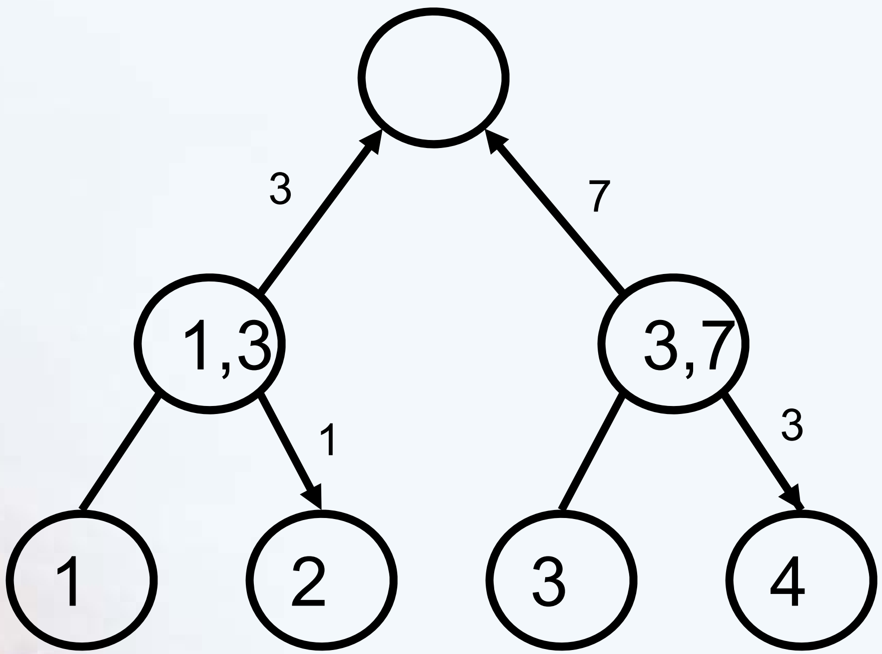
Prefix Sums on a Tree: More



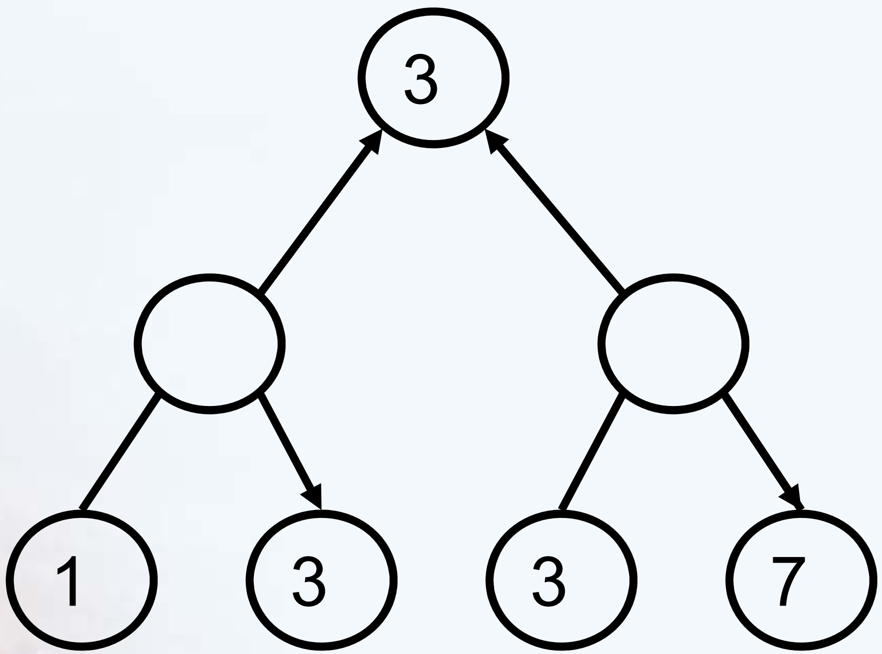
Prefix Sums on a Tree: More



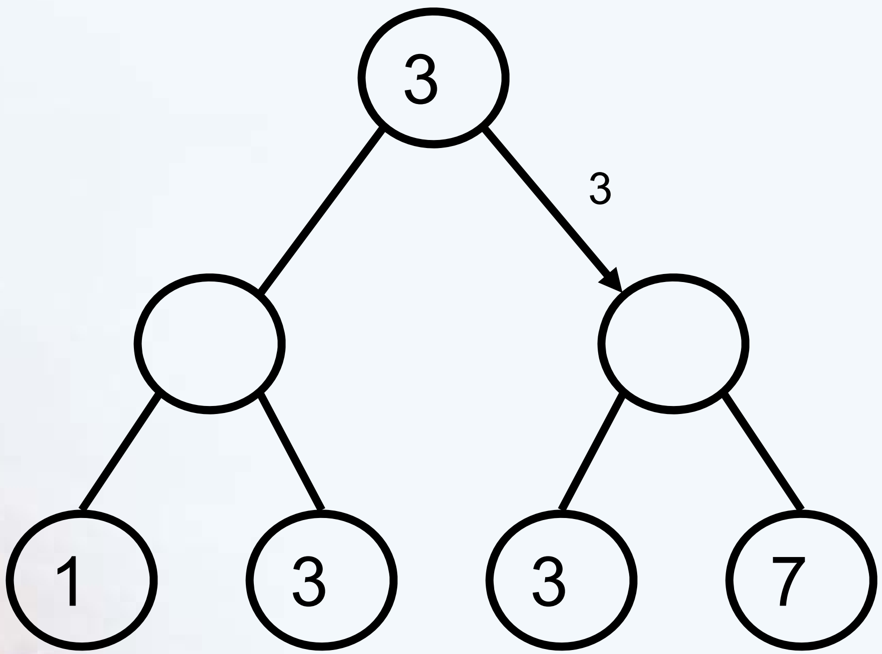
Prefix Sums on a Tree: More



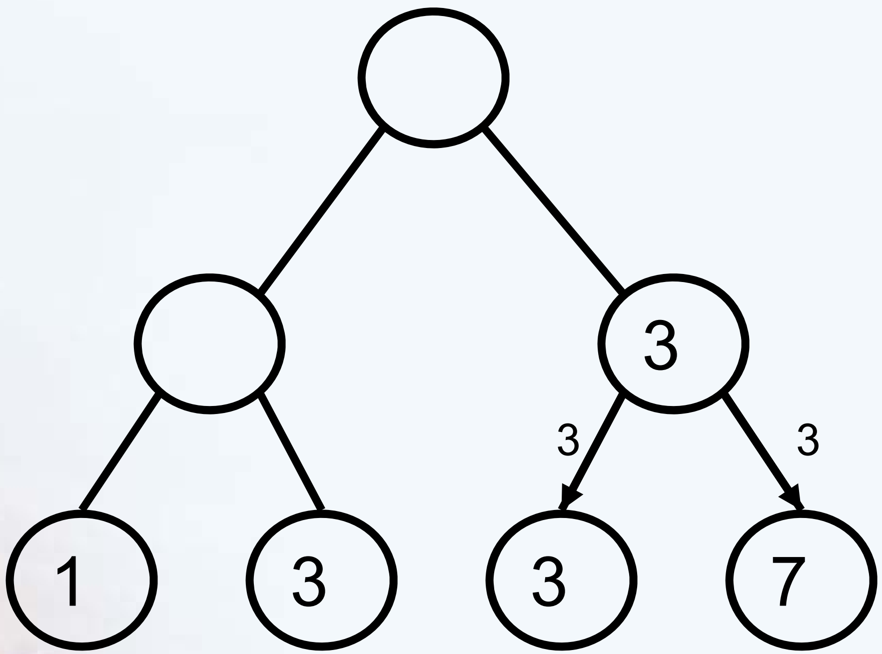
Prefix Sums on a Tree: More



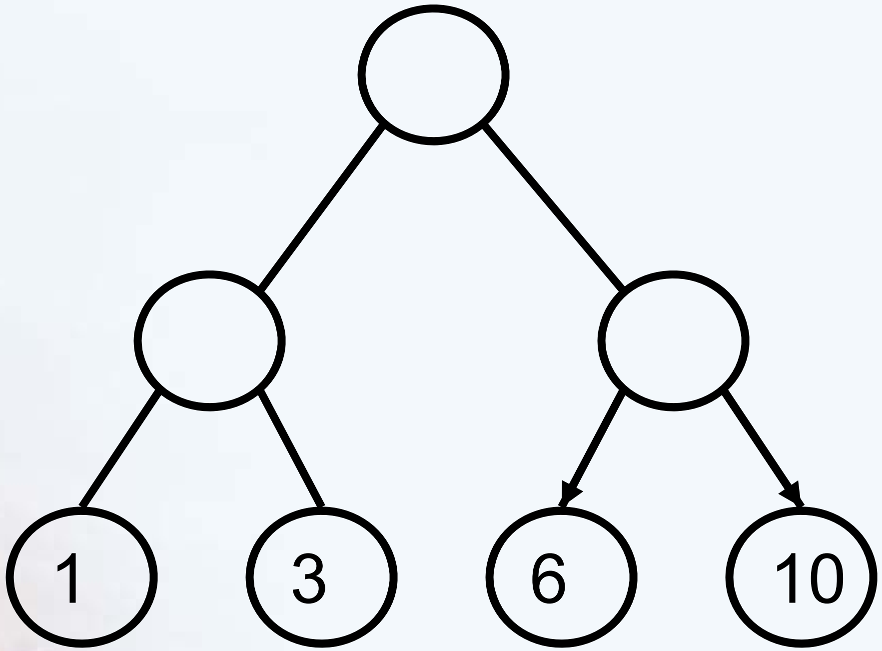
Prefix Sums on a Tree: More



Prefix Sums on a Tree: More

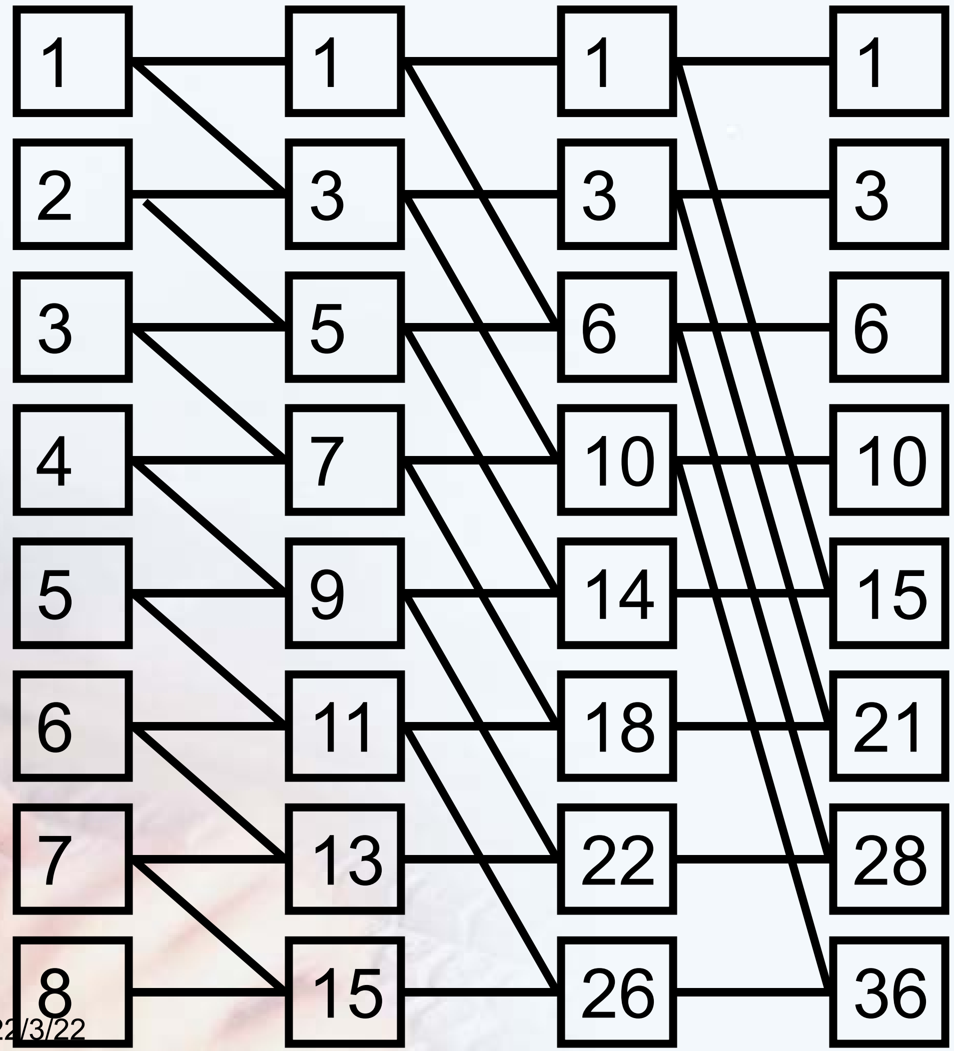


Prefix Sums on a Tree: More



Prefix Sums on a Tree: More

- Properties:
  - time: $2 \log n$
  - processors: $2n - 1$
  - cost $O(n \log n)$
- Comparison with PRAM algorithm
  - asymptotically equivalent
  - in practice, less efficient
    - weaker model

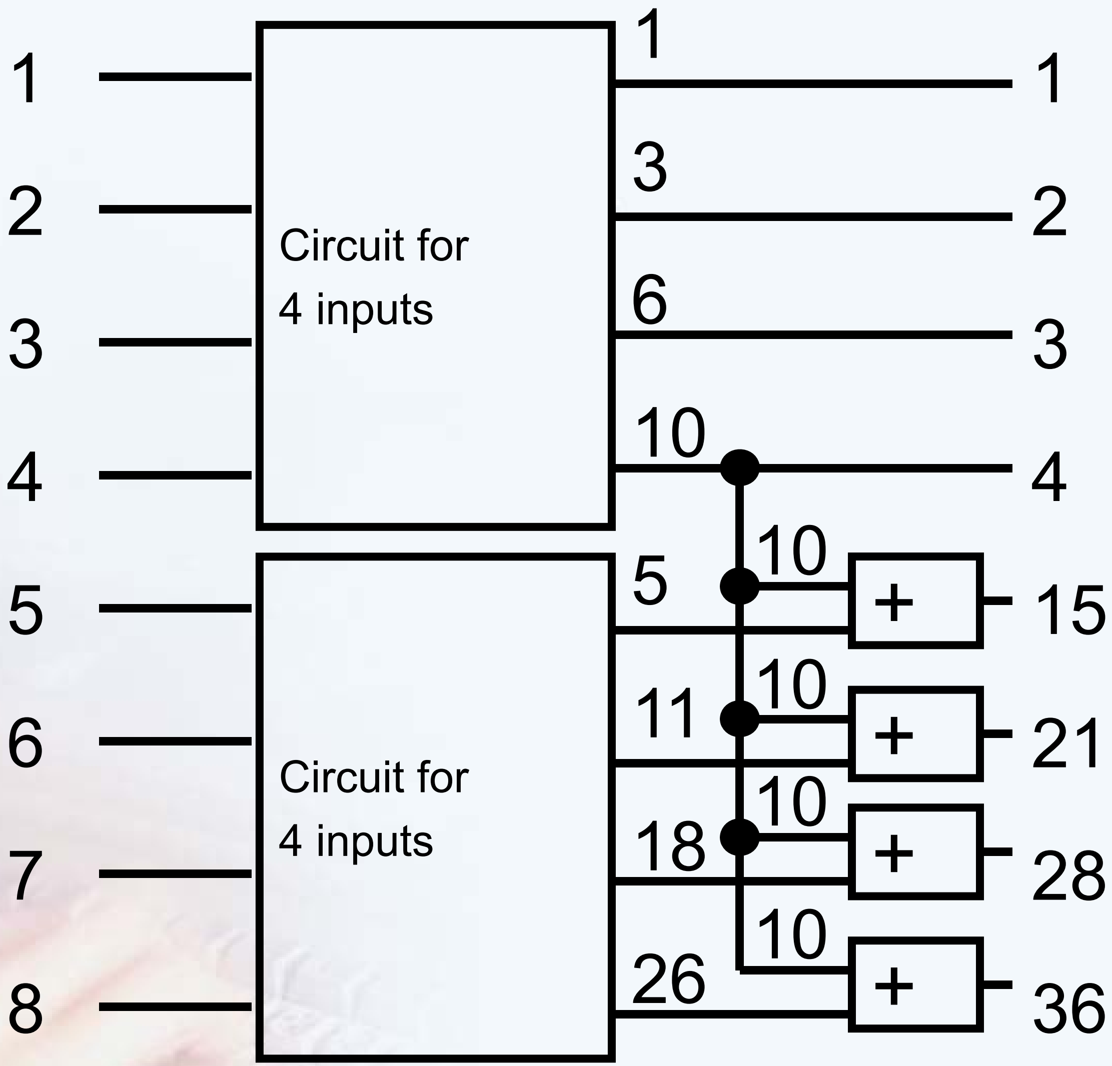


depth (time) = 3

complexity (cost) $= 3 \times 8$

$= 24$

Specialized Circuit



Recursive Circuit

第七章 并行算法常用设计技术

7.1 划分设计技术

7.2 分治设计技术

7.3 平衡树设计技术

7.4 倍增设计技术

7.4.1 设计思想

7.4.2 表序问题

7.4.3 求森林的根

7.5 流水线设计技术

倍增设计技术

- 设计思想
  - 又称指针跳跃(pointer jumping)技术，特别适合于处理链表或有向树之类的数据结构；
  - 当递归调用时，所要处理数据之间的距离逐步加倍，经过k步后即可完成距离为2^k的所有数据的计算。
- 示例
  - 表序问题
  - 求森林的根

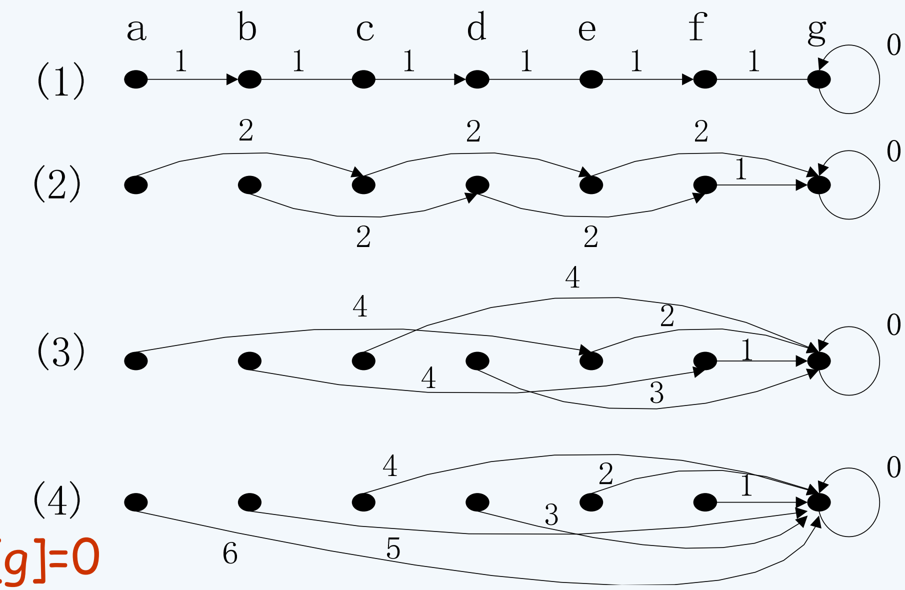
第七章 并行算法常用设计技术

7.1 划分设计技术
7.2 分治设计技术
7.3 平衡树设计技术
7.4 倍增设计技术
7.4.1 设计思想
7.4.2 表序问题
7.4.3 求森林的根
7.5 流水线设计技术

表序问题（1）

- 问题描述

n个元素的列表L，求出每个元素在L中的次第号(秩或位序或rank(k))，rank(k)可视为元素k至表尾的距离；



- 示例： n=7

(1)p[a]=b, p[b]=c, p[c]=d, p[d]=e,
p[e]=f, p[f]=g, p[g]=g
r[a]=r[b]=r[c]=r[d]=r[e]=r[f]=1, r[g]=0

(2)p[a]=c, p[b]=d, p[c]=e, p[d]=f, p[e]=p[f]=p[g]=g
r[a]=r[b]=r[c]=r[d]=r[e]=2, r[f]=1, r[g]=0

(3)p[a]=e, p[b]=f, p[c]=p[d]=p[e]=p[f]=p[g]=g
r[a]=4, r[b]=4, r[c]=4, r[d]=3, r[e]=2, r[f]=1, r[g]=0

(4)p[a]=p[b]=p[c]=p[d]=p[e]=p[f]=p[g]=g
r[a]=6, r[b]=5, r[c]=4, r[d]=3, r[e]=2, r[f]=1, r[g]=0

表序问题（2）

- 算法：P200算法7.10

(1)并行做：初始化p[k]和distance[k] //O(1)

(2)执行 $\lceil \log n \rceil$ 次 //O(logn)

(2.1)对k并行地做 //O(1)

如果k的后继不等于k的后继之后继，则

(i) distance[k]= distance[k]+ distance[p[k]]

(ii) p[k]=p[p[k]]

(2.2)对k并行地做

rank[k]=distance[k] //O(1)

运行时间：t(n)=O(logn) p(n)=n

第七章 并行算法常用设计技术

7.1 划分设计技术
7.2 分治设计技术
7.3 平衡树设计技术
7.4 倍增设计技术
7.4.1 设计思想
7.4.2 表序问题
7.4.3 求森林的根
7.5 流水线设计技术

求森林的根（1）

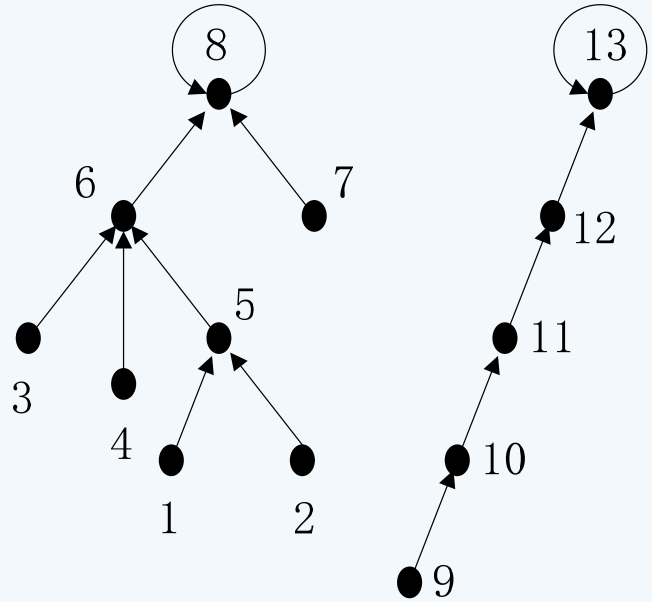
- 问题描述

一组有向树F中，如果<i, j>是F中的一条弧，则p[i]=j(即j是i的双亲)；若i为根，则p[i]=i。求每个结点j(j=1~n)的树根s[j].

- 示例

初始时

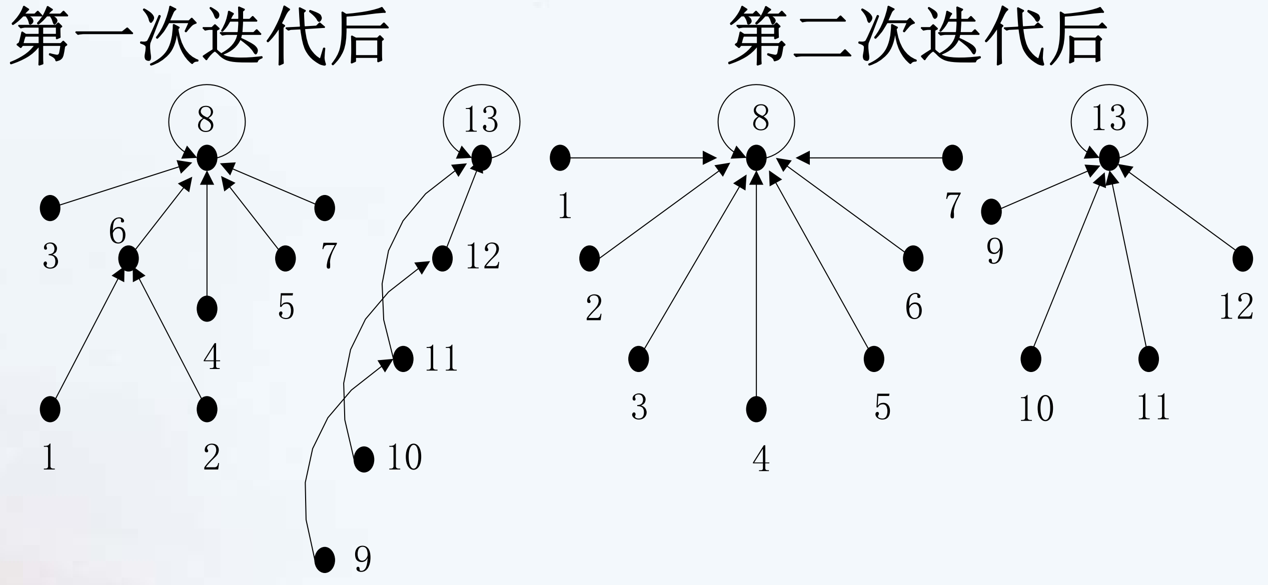
P[1]=p[2]=5　p[3]=p[4]=p[5]=6
P[6]=p[7]=8　p[8]=8　P[9]=10
p[10]=11 p[11]=12　p[12]=13　p[13]=13
s[i]=p[i]



(a)

求森林的根（2）

- 示例



(b) (c)

- 算法：P202算法7.11
- 运行时间：t(n)=O(logn)　W(n)=O(nlogn)

第七章 并行算法常用设计技术

7.1 划分设计技术

7.2 分治设计技术

7.3 平衡树设计技术

7.4 倍增设计技术

7.5 流水线设计技术

7.5.1 设计思想

7.5.2 5-point DFT的计算

7.5.3 多线程软件流水

流水线设计技术

- 设计思想
  - 将算法流程划分成p个前后衔接的任务片断，每个任务片断的输出作为下一个任务片断的输入；
  - 所有任务片断按同样的速率产生出结果。
- 评注
  - 流水线技术是一种广泛应用在并行处理中的技术；
  - 脉动算法(Systolic algorithm)是其中一种流水线技术；

第七章 并行算法常用设计技术

7.1 划分设计技术
7.2 分治设计技术
7.3 平衡树设计技术
7.4 倍增设计技术
7.5 流水线设计技术
7.5.1 设计思想
7.5.2 5-point DFT的计算
7.5.3 多线程软件流水

5-point DFT的计算（1）

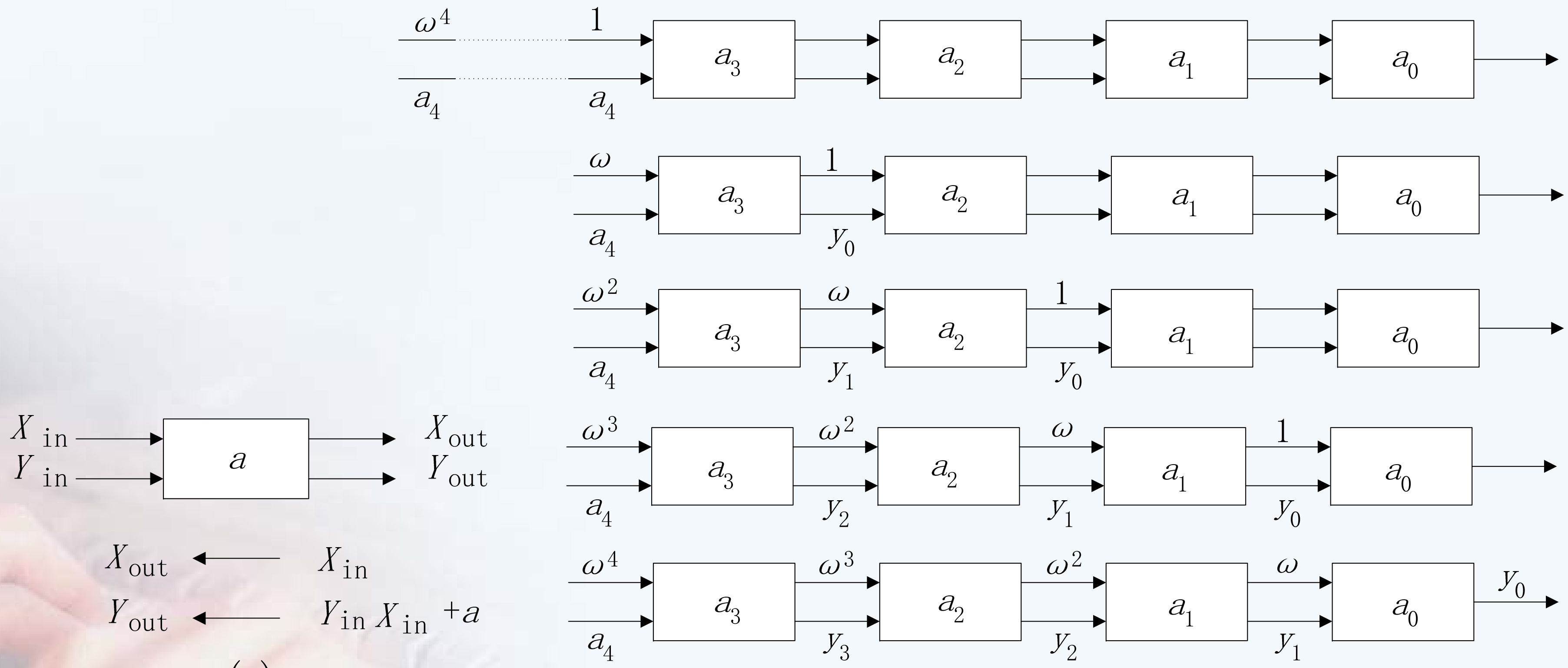
- 问题描述

5-point DFT的计算。应用秦九韶(Horner)法则，

$$\begin{cases} y_0 = b_0 = a_4\omega^0 + a_3\omega^0 + a_2\omega^0 + a_1\omega^0 + a_0 \\ y_1 = b_1 = a_4\omega^4 + a_3\omega^3 + a_2\omega^2 + a_1\omega^1 + a_0 \\ y_2 = b_2 = a_4\omega^8 + a_3\omega^6 + a_2\omega^4 + a_1\omega^2 + a_0 \\ y_3 = b_3 = a_4\omega^{12} + a_3\omega^6 + a_2\omega^4 + a_1\omega^2 + a_0 \\ y_4 = b_4 = a_4\omega^{16} + a_3\omega^6 + a_2\omega^4 + a_1\omega^2 + a_0 \end{cases} \Rightarrow \begin{cases} y_0 = (((a_4\omega^0 + a_3)\omega^0 + a_2)\omega^0 + a_1)\omega^0 + a_0 \\ y_1 = (((a_4\omega^1 + a_3)\omega^1 + a_2)\omega^1 + a_1)\omega^1 + a_0 \\ y_2 = (((a_4\omega^2 + a_3)\omega^2 + a_2)\omega^2 + a_1)\omega^2 + a_0 \\ y_3 = (((a_4\omega^3 + a_3)\omega^3 + a_2)\omega^3 + a_1)\omega^3 + a_0 \\ y_4 = (((a_4\omega^4 + a_3)\omega^4 + a_2)\omega^4 + a_1)\omega^4 + a_0 \end{cases}$$

5-point DFT的计算（2）

- 示例： p(n)=n-1, t(n)=2n-2=O(n)



(a)

(b)

第七章 并行算法常用设计技术

7.1 划分设计技术

7.2 分治设计技术

7.3 平衡树设计技术

7.4 倍增设计技术

7.5 流水线设计技术

7.5.1 设计思想

7.5.2 5-point DFT的计算

7.5.3 多线程软件流水

多线程软件流水例子

- 问题描述：用多线程流水方法计算下面阵列：

$$A:\begin{pmatrix} 0 & 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 2 & 5 & 11 & 21 & 36 & 57 \\ 3 & 5 & 10 & 21 & 42 & 78 & 135 \\ 6 & 11 & 21 & 42 & 84 & 162 & 297 \end{pmatrix}$$

计算公式如下：

$$\mathrm{A}[i,\ j]=\begin{cases} 0 & if\ \ i=0\ and\ \ j=0 \\ A[0,\ j-1]+j & if\ \ i=0\ and\ \ j>0 \\ A[i-1,0]+i & if\ \ i>0\ and\ \ j=0 \\ \mathrm{A}[i-1,\ j]+\mathrm{A}[i,\ j-1] & other \end{cases}$$

多线程软件流水例子

- 问题描述：线程和流程设计：

线程 \步骤	$S0$	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$			
$T0$	0	1	3	6	10	15	21			
$T1$		1	2	5	11	21	36	57		
$T2$			3	5	10	21	42	78	135	
$T3$				6	11	21	42	84	162	297

计算公式：

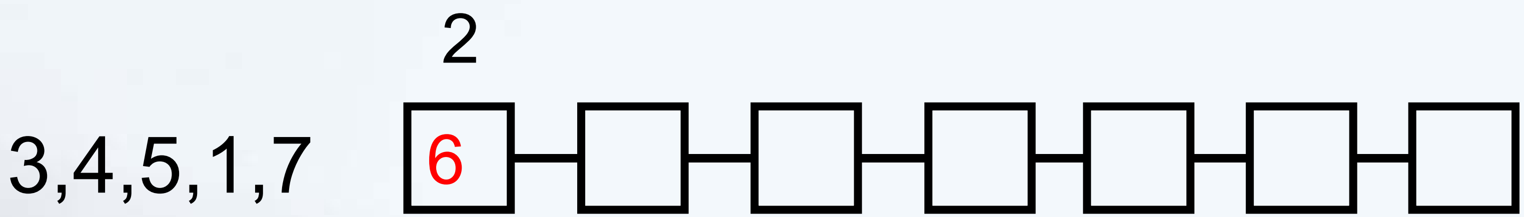
$$\mathrm{data}[T_i, S_j] = \begin{cases} 0 & \textit{if } i = 0 \textit{ and } j = 0 \\ \mathrm{data}[T_0, S_{j\text{-}1}] + j & \textit{if } i = 0 \textit{ and } j > 0 \\ \mathrm{data}[T_{i\text{-}1}, S_0] + i & \textit{if } i > 0 \textit{ and } j = 0 \\ \mathrm{data}[T_{i\text{-}1}, S_j] + \mathrm{data}[T_i, S_{j\text{-}1}] & \textit{other} \end{cases}$$

Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 1

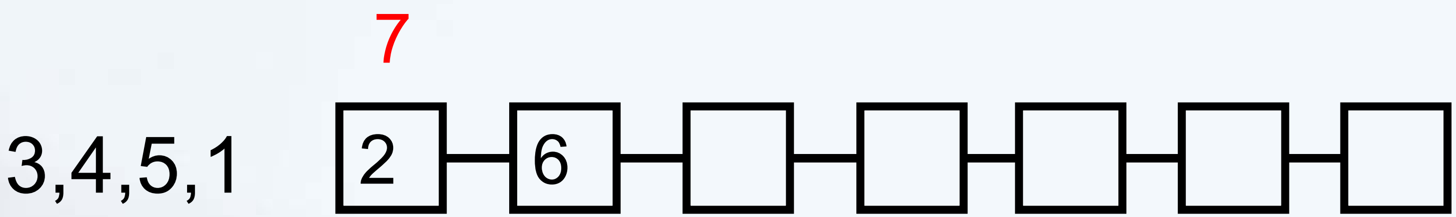
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 2



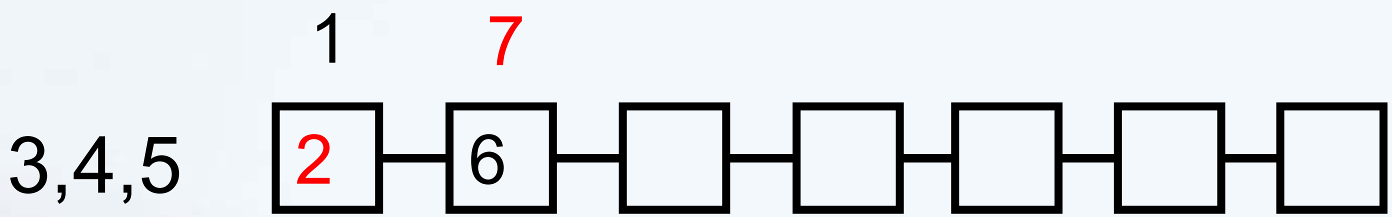
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 3



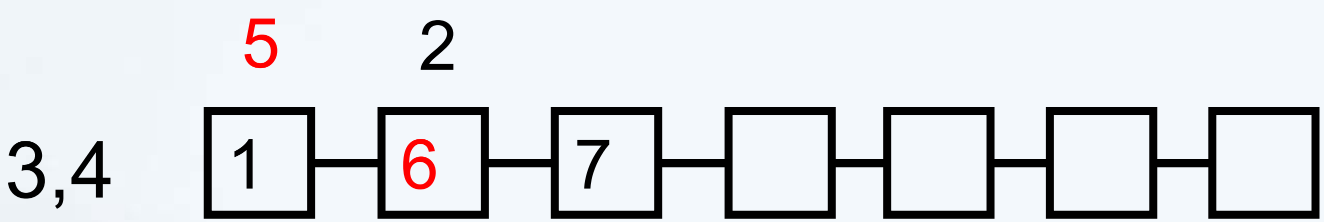
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 4



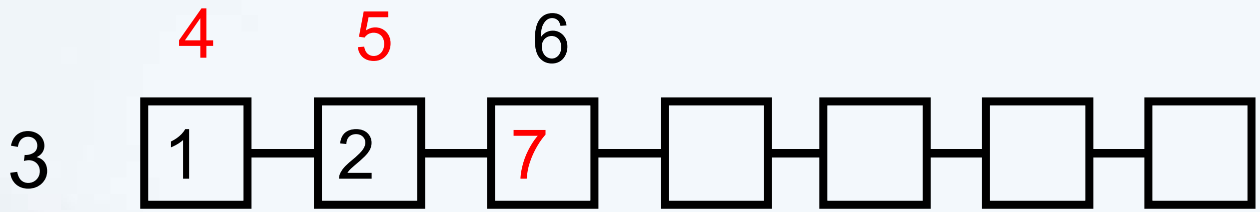
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 5



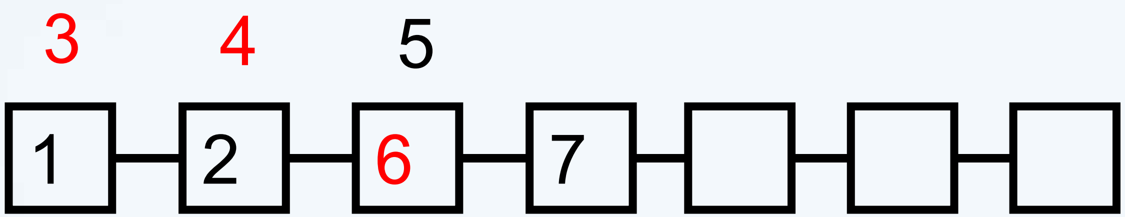
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 6



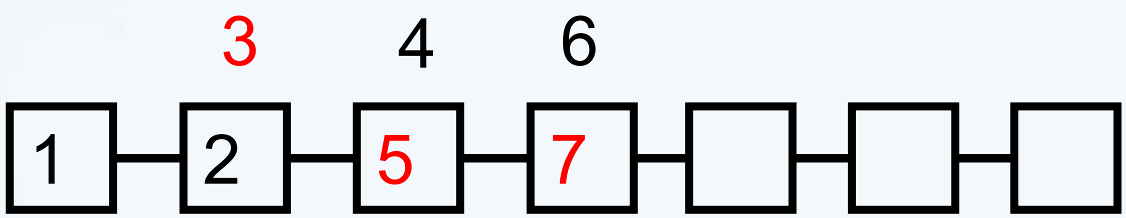
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 7



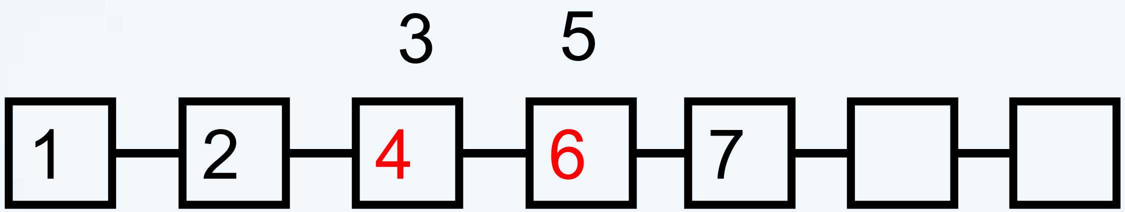
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 8



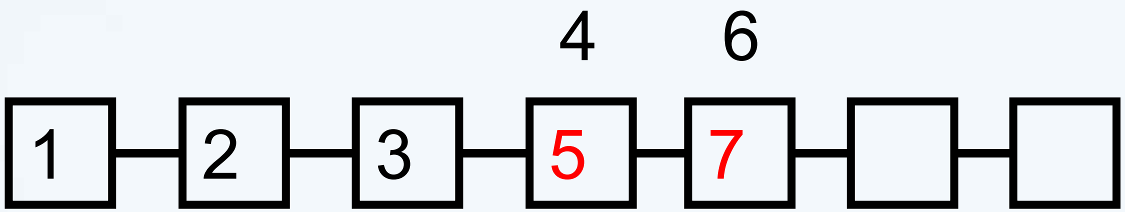
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 9



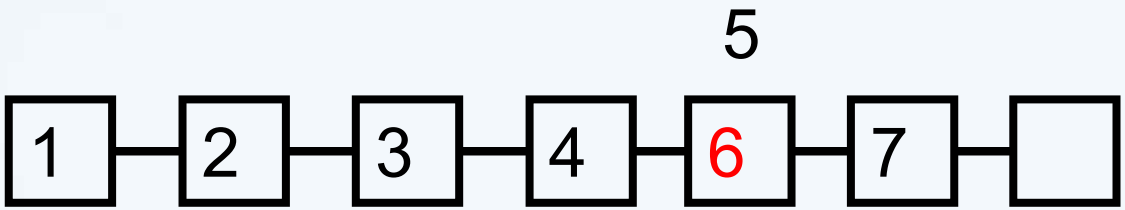
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 10



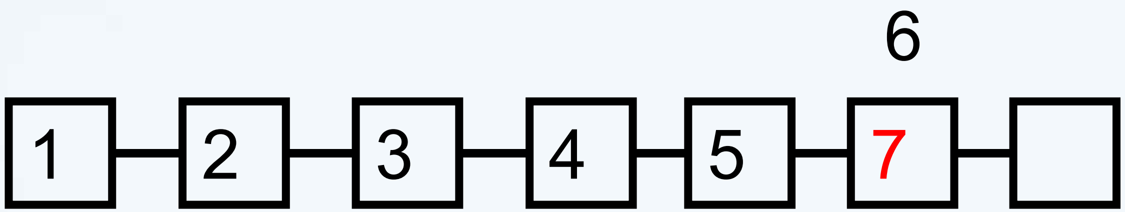
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 11



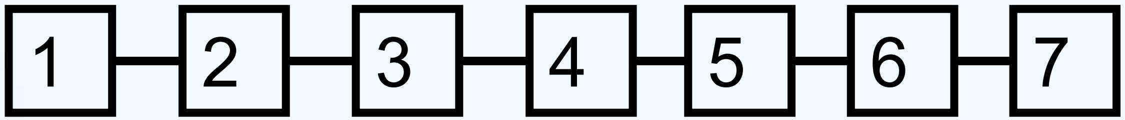
Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 12



Systolic排序算法示例: Sorting 3, 4, 5, 1, 7, 2, 6

Step 13



作 业

1.以下是上三角方程组回代解法的串行算法的形式化描述。（算法10.1）

输入： A_{n*n} $b = (b_1, \dots, b_n)^T$ 输出： $x = (x_1, \dots, x_n)^T$

```
Begin
    (1)for i=n downto 1 do
        (1.1)x_i=b_i/a_ii
        (1.2)for j=1 to i-1 do
                b_j=b_j-a_ji x_i
                a_ji=0
            endfor
        endfor
End
```

①请指出串行算法哪些部分可以并行化。②写出并行算法的形式化描述（需要注明计算模型类型），分析你的算法的时间复杂度。

2. 习题7-10（P207）