第二篇 并行算法的设计

第四章 并行算法的设计基础
第五章 并行算法的一般设计方法
第六章 并行算法的基本设计技术
第七章 并行算法的一般设计过程

6.3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

第六章 并行算法的基本设计技术

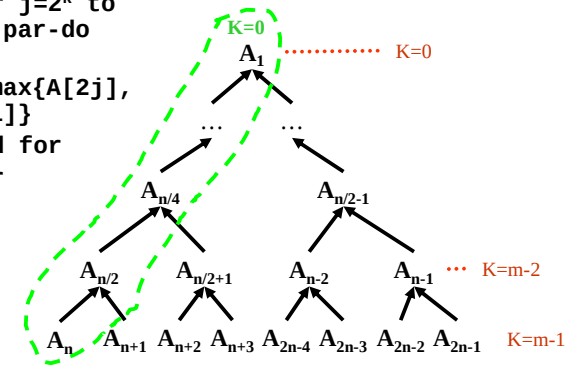
6.1 划分设计技术
6.2 分治设计技术
6.3 平衡树设计技术
6.4 倍增设计技术
6.5 流水线设计技术

算法6.8: 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
```

$t(n)=m\times O(1)=O(\log n)$
$p(n)=n/2$



计算前缀和

- 问题定义

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

  $S_i=x_1*x_2*\ldots*x_i$，$1\leqslant i\leqslant n$ 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： p154算法6.9 SIMD-TC上非递归算法

  令A[i]=x_i, i=1~n,

  B[h,j]和C[h,j]为辅助数组(h=0~logn, j=1~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)

Input: Sequence x of $n=2^k$ elements of type T, binary associative operator $\oplus: T\times T\to T$.
Output: Sequence s of $n=2^k$ elements of type T, with $s_k=\oplus_{i=1}^{k}x_i$ for $1\leqslant k\leqslant n$.

PRAM求前缀和算法

sequenceT SCAN(**sequence**T *x*, $\oplus$: $T\times T\to T$)

```
if n = 1 then
    s_1 ← x_1
    return s
endif
forall i ∈ 1:n/2 do
    y_i ← x_{2i-1} ⊕ x_{2i}
enddo
⟨z_1, ..., z_{n/2}⟩ ← SCAN(⟨y_1, ..., y_{n/2}⟩, ⊕)
forall i ∈ 1:n do
    if even(i) then
        s_i ← z_{i/2}
    elsif i = 1 then
        s_1 ← x_1
    else
        s_i ← z_{(i-1)/2} ⊕ x_i
    endif
enddo
return s
```

x为B树第k层

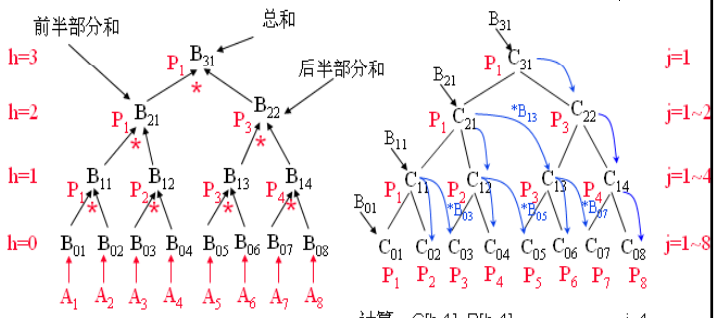
Scan返回为C树第k层

y为B树第k-1层

z为C树第k-1层

由z构造出s，也就是C树第k层

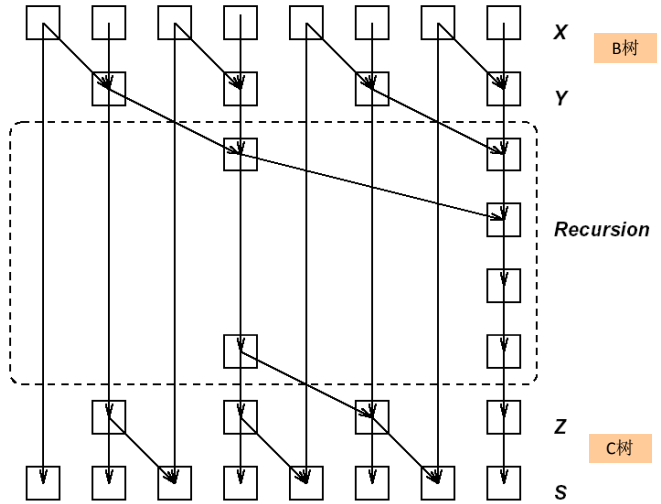
例：n=8, p=8, C01~C08为前缀和



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)

8个元素的数据流分析



6.4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

表序问题

- 算法：P155算法6.10

(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

表序问题

- 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

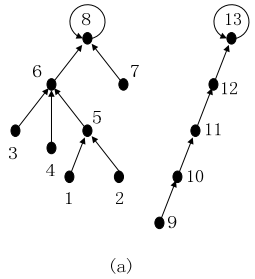
求森林的根

- 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

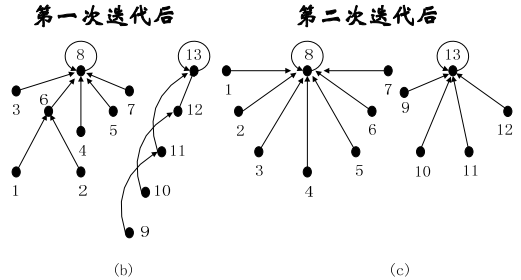
初始时
P[1]=p[2]=5 p[3]=p[4]=p[5]=6
P[6]=p[7]=8 p[8]=8 P[9]=10
p[10]=11 p[11]=12 p[12]=13 p[13]=13
s[i]=p[i]



(a)

求森林的根

- 示例



- 算法：P157算法6.11
- 运行时间：t(n)=O(logn) W(n)=O(nlogn)

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。

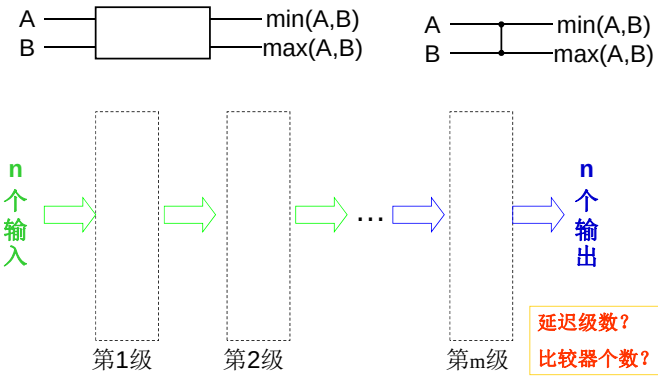
6.2 分治设计技术

6.2.1 并行分治设计步骤
6.2.2 比较器排序网络
6.2.3 凸壳问题

Batcher归并、排序网络

- 比较器操作和[0,1]原理
- 奇偶归并网络
- 双调归并网络
- Batcher排序网络
- 2D-Mesh上的排序算法

比较器操作和[0,1]原理



延迟级数？

比较器个数？

[0,1]原理：如果一个n输入的网络能够排序所有2^n种0,1序列，那么它也能排序n个数的任意序列。

0, 1, 0, 0, 1, 1, 0 => 0, 0, 0, 0, 1, 1, 1

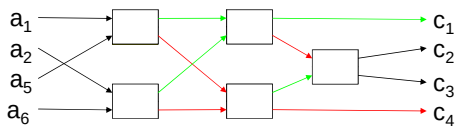
反证法：设f是单调函数，且网络对于序列$(a_1,\ldots,a_n)$排序的结果$(b_1,\ldots,b_n)$中存在b_i有$b_i>b_{i+1}$,即b序列不是有序的.

那么，该网络对于序列$(f(a_1),\ldots,f(a_n))$排序成$(f(b_1),\ldots,f(b_n))$且$f(b_i)!=f(b_{i+1})$时有$f(b_i)>f(b_{i+1})$.

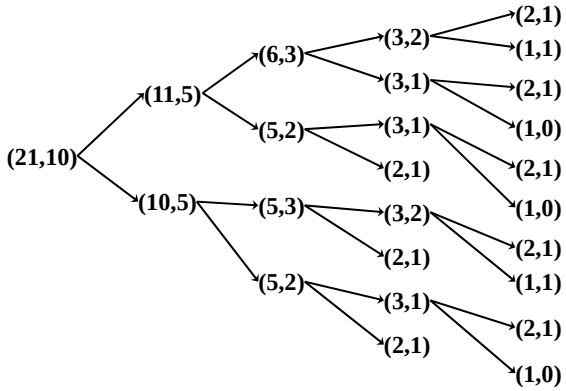
令$b_j<b_i$:$f(b_j)=0$; $b_j>=b_i$:$f(b_j)=1$,则有

$f(b_i)=1, f(b_{i+1})=0$, 所以$(f(b_1),\ldots,f(b_n))$不是有序的，即不能够对输入的(0,1)序列排序。

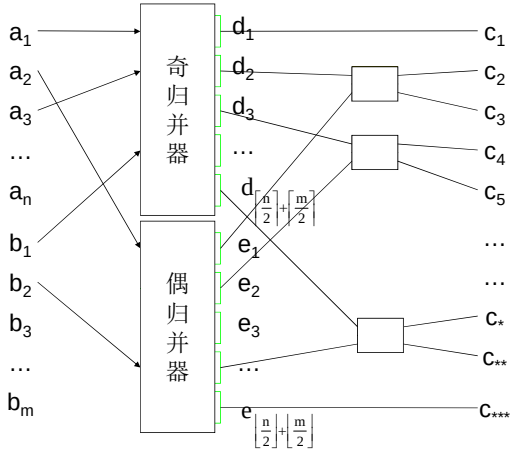
四个数的排序网络



(m,n)奇偶归并网络



(m,n)奇偶归并网络



结果序列（c序列）

$$d_1,\quad e_1:d_2,\quad \ldots\quad e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor-1}:d_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor},\quad c_*,\quad c_{**},\quad c_{***}$$

$\left\lceil \frac{n}{2}\right\rceil+\left\lceil \frac{m}{2}\right\rceil$与$\left\lfloor \frac{n}{2}\right\rfloor+\left\lfloor \frac{m}{2}\right\rfloor$之差：

0	$c_* = e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor}$,	c_{**}不存在	c_{***}不存在
1	$c_*:c_{**} = e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor}:d_{\lceil \frac{n}{2}\rceil+\lceil \frac{m}{2}\rceil}$		c_{***}不存在
2	$c_*:c_{**} = e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor}:d_{\lceil \frac{n}{2}\rceil+\lceil \frac{m}{2}\rceil-1}$		$c_{***} = d_{\lceil \frac{n}{2}\rceil+\lceil \frac{m}{2}\rceil}$

- 两个序列中的奇数位元素拿出来组成两个有序序列，由奇归并器归并；
- 两个序列中的偶数位元素拿出来组成两个有序序列，由偶归并器归并；

$$(a_1,a_3,\ldots,a_{2\lceil \frac{n}{2}\rceil-1}),(b_1,b_3,\ldots,b_{2\lceil \frac{m}{2}\rceil-1})\Rightarrow(d_1,d_2,\ldots,d_{\lceil \frac{n}{2}\rceil+\lceil \frac{m}{2}\rceil})$$

$$(a_2,a_4,\ldots,a_{2\lfloor \frac{n}{2}\rfloor}),(b_2,b_4,\ldots,b_{2\lfloor \frac{m}{2}\rfloor})\Rightarrow(e_1,e_2,\ldots,e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor})$$

- 奇归并器输出的第一个数做为结果，第i个数与偶归并器输出的第i+1个两两比较，依次作为结果。

$$d_1,\quad e_1:d_2,\quad \ldots\quad e_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor-1}:d_{\lfloor \frac{n}{2}\rfloor+\lfloor \frac{m}{2}\rfloor},\quad c_*,\quad c_{**},\quad c_{***}$$

定理：Batcher奇偶归并网络能够产生正确的归并。

采用[0,1]原理证明如下：

假设： a: k个0, $n-k$个1 d: $\left\lceil \frac{k}{2}\right\rceil+\left\lceil \frac{l}{2}\right\rceil$个0

b: l个0, $m-l$个1 e: $\left\lfloor \frac{k}{2}\right\rfloor+\left\lfloor \frac{l}{2}\right\rfloor$个0

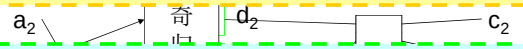
$\left\lceil \frac{k}{2}\right\rceil+\left\lceil \frac{l}{2}\right\rceil$与$\left\lfloor \frac{k}{2}\right\rfloor+\left\lfloor \frac{l}{2}\right\rfloor$之差：

0 $c_*=0$,且为第$2(\left\lfloor \frac{k}{2}\right\rfloor+\left\lfloor \frac{l}{2}\right\rfloor)=(\left\lfloor \frac{k}{2}\right\rfloor+\left\lfloor \frac{l}{2}\right\rfloor)+(\left\lceil \frac{k}{2}\right\rceil+\left\lceil \frac{l}{2}\right\rceil)=k+l$项

1 $c_*:c_{**}=e_{\lfloor \frac{k}{2}\rfloor+\lfloor \frac{l}{2}\rfloor}:d_{\lceil \frac{k}{2}\rceil+\lceil \frac{l}{2}\rceil}=0:0$且分别是第$k+l$-1和$k+l$项

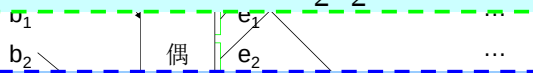
2 $c_{***}=d_{\lceil \frac{k}{2}\rceil+\lceil \frac{l}{2}\rceil}=0$且为第$k+l$项

$$C_{OE}^{M}(m,n)=\begin{cases} mn, & mn\le 1 \\ C_{OE}^{M}(\left\lceil \frac{m}{2}\right\rceil,\left\lceil \frac{n}{2}\right\rceil)+C_{OE}^{M}(\left\lfloor \frac{m}{2}\right\rfloor,\left\lfloor \frac{n}{2}\right\rfloor)+\left\lceil \frac{m+n-2}{2}\right\rceil, & mn>1 \end{cases}$$



$$\lim_{n\to\infty} C_{OE}^{M}(n,n)=O(n\log n)$$

$$m=n=2^t:\quad C_{OE}^{M}(m,n)=2C_{OE}^{M}(\frac{n}{2},\frac{n}{2})+n-1=n\log n+1$$

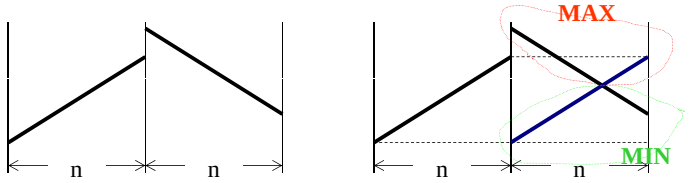


$$D_{OE}^{M}(m,n)=1+\max(D_{OE}^{M}(\left\lceil \frac{m}{2}\right\rceil,\left\lceil \frac{n}{2}\right\rceil),D_{OE}^{M}(\left\lfloor \frac{m}{2}\right\rfloor,\left\lfloor \frac{n}{2}\right\rfloor))$$

$$m=n=2^t:\quad D_{OE}^{M}(m,n)=1+\log n=1+t$$

定理(Bathcer定理)：给定一个双调序列$a_1, a_2, \ldots, a_{2n}$，对于每一$1<=i<=n$执行a_i和a_{i+n}比较交换得到$b_i=\min(a_i, a_{i+n})$，$c_i=\max(a_i, a_{i+n})$所形成的两个子序列$MIN=(b_1, b_2, \ldots, b_n)$和$MAX=(c_1, c_2, \ldots, c_n)$均是双调序列，且对于所有的$1<=i, j<=n$ 满足$b_i<=c_j$.

① 对半情况



双调归并网络

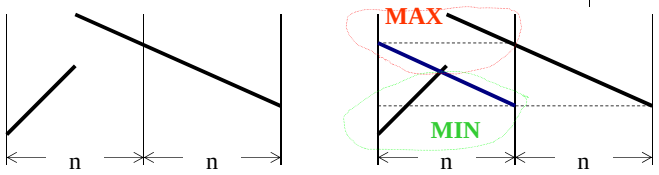
定义：一个序列$a_1,a_2,\ldots,a_n$是双调序列，如果（1）存在着一个a_k,$1<=k<=n$使得$a_1>=\ldots>=a_k<=\ldots<=a_n$成立；或者（2）此序列能够通过循环移位使得条件（1）成立。

举例：一个元素是；两个元素是。

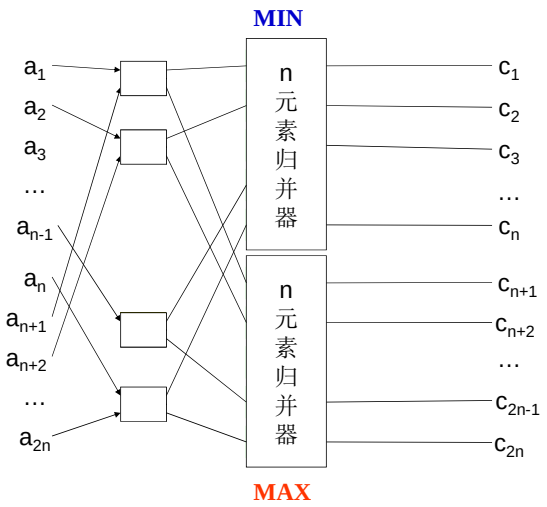
对半情况： 1，3，5，7，9，8，4，2

不对半情况： 6，4，2，0，1，7，9

② 不对半情况



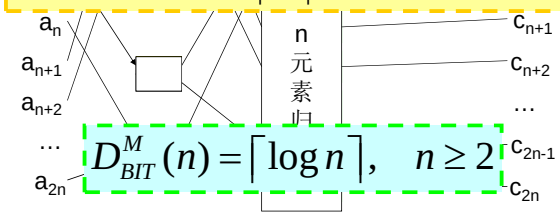
③ 其他情况：如果a序列不是上述两种情况，则根据双调序列定义，通过循环移位可变成上述情况之一。再按①、②得到MIN和MAX，对于二者进行逆向循环移位，则仍是双调的，则结论成立。



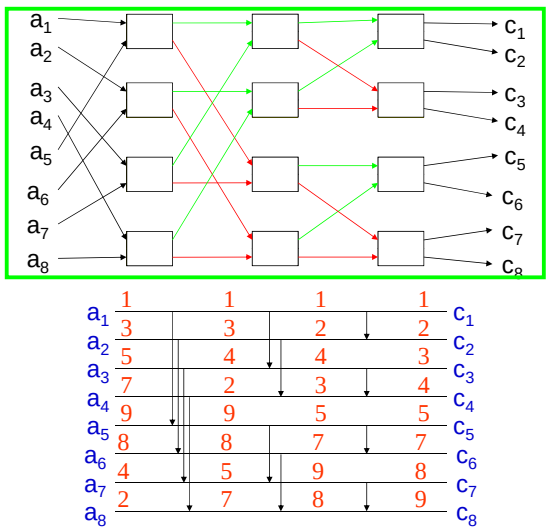
$$C_{BIT}^{M}(n) = C_{BIT}^{M}(\left\lceil \frac{n}{2} \right\rceil) + C_{BIT}^{M}(\left\lfloor \frac{n}{2} \right\rfloor) + \left\lfloor \frac{n}{2} \right\rfloor, \quad n \geq 2$$

$$n = 2^t \quad C_{BIT}^{M}(2^t) = 2C_{BIT}^{M}(2^{t-1}) + \frac{n}{2} = \frac{n}{2}\log n$$

$$D_{BIT}^{M}(n) = 1 + D_{BIT}^{M}(\left\lceil \frac{n}{2} \right\rceil)$$

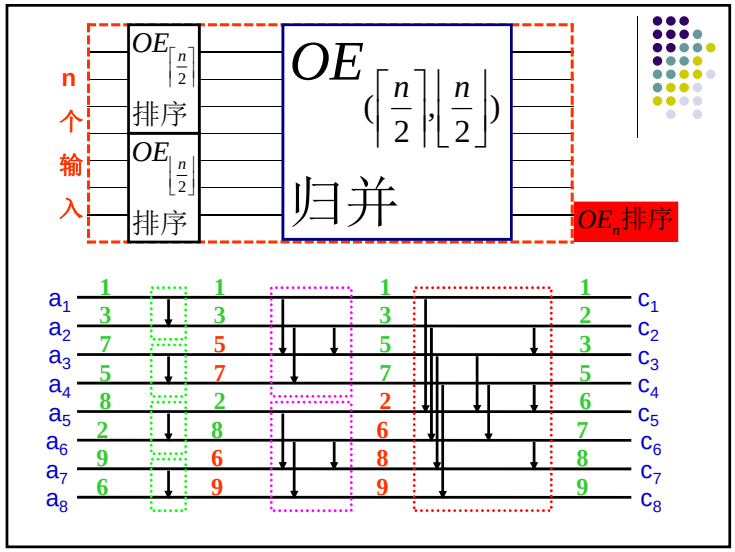


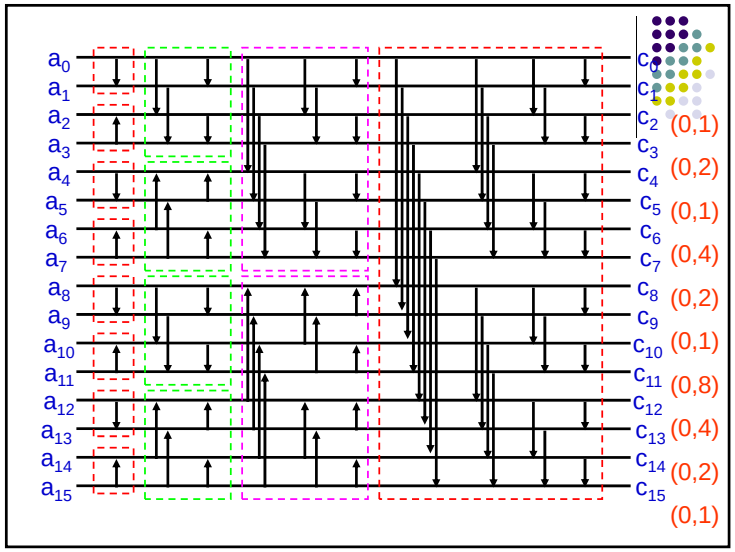
$$D_{BIT}^{M}(n) = \lceil \log n \rceil, \quad n \geq 2$$

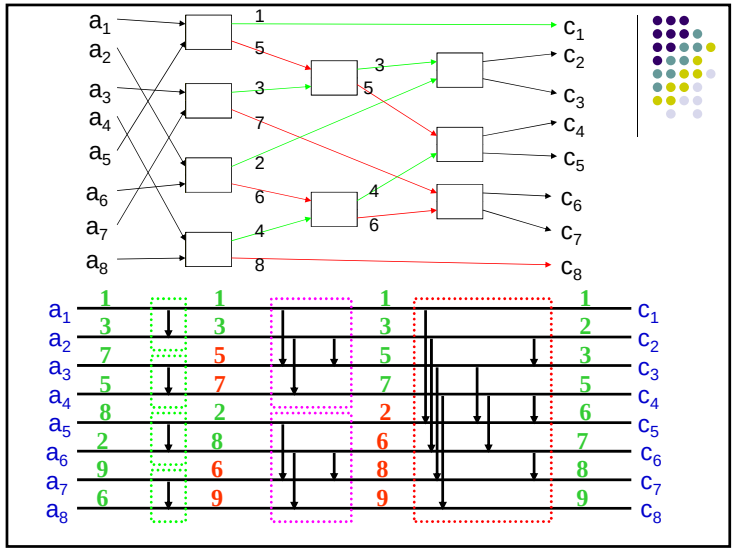


Batcher排序网络

- 对输入数进行两两比较，以形成长度为2的诸有序序列；
- 使用奇偶归并网络和双调归并网络，对两两长度各为2的有序序列施行归并，以形成一些长度为4的有序序列；
- 重复上述步骤，直到形成两个长度各为n/2的有序序列；
- 对这两个序列进行归并最终形成结果。







二维网孔上的排序算法

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

行主编号：

1,2,1,1,2,1,2,1,2,1

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

洗牌编号：

1,1,1,2,1,1,2,2,1,1

$$tr(2^t) = S\left(\left\lceil 2^{\frac{t}{2}-1} \right\rceil\right) + tr(2^{t-1})$$

$$\sum_{k=1}^{t} tr(2^k) = \sum_{k=1}^{t} (t-k+1) S\left(\left\lceil 2^{\frac{k}{2}-1} \right\rceil\right)$$

$k = 2l$与$k = 2l-1$时的$S\left(\left\lceil 2^{\frac{k}{2}-1} \right\rceil\right)$相同：

洗牌编号排序总选路数

$$(t-(2l)+1)S(2^{l-1})$$

tr归并中的总选路数

$$(t-(2l-1)+1)S(2^{\frac{2l-1}{2}-1})$$

$S(j)$交换相距j的元素的选路数

$$= \sum_{l=1}^{\frac{t}{2}} (2t-4l+3) S(2^{l-1}) = \sum_{l=1}^{\frac{t}{2}} (2t-4l+3) 2^{l-1}$$

算法描述

- 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{xi,xi+n/2}
     (1.2) l_i=max{xi,xi+n/2}
     end for
  (2)Recursive Call:
     (2.1)BITONIC_MERG(MIN=(s_0,...,s_n/2-1))
     (2.2)BITONIC_MERG(MAX=(l_0,..., l_n/2-1))
  (3)output sequence MIN followed by sequence MAX
End
```

2015网络与数据科学研讨会

发布时间 2015-05-07

2015网络与数据科学研讨会

联合主办：西安交通大学
华为技术有限公司
微软亚洲研究院

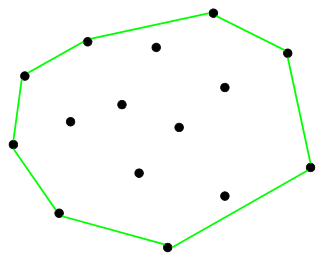
时间：2015年5月11日（周一）
地点：科学馆101
会议主要讨论大数据，计算机网络，信息计算机通信以及相关的生物，数理等学科的科学前沿问题。
会议邀请来自南华大学、中国科学院软件所、华为技术有限公司、微软亚洲研究院、百度及其他的多名相关领域知名专家作大会报告。

大会报告（下午）

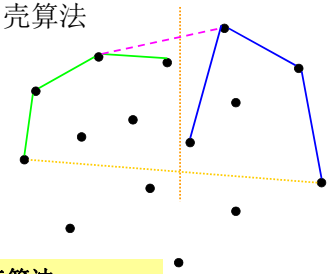
14:30-15:15	深度图像理解：从图像到人脸，从物体到像素 微软亚洲研究院-首席研究员：孙剑博士	科学馆101 报告厅
15:15-16:00	BSP Algorithms and Software for Networks and Big Data 华为法国研究院-首席专家 Bill McColl教授	
休息		
16:10-16:55	互联网的大数据实践 百度-首席专家 张潼博士	

从BSP模型看并行计算发展

凸壳问题



折半分治凸壳算法



PRAM上求上凸壳算法

- 点数小于等于4直接求；
- 按照x坐标把点集分成两半；
- 分别求各半的上凸壳；
- 求二者的上公切线；
- 得到结果.

PRAM上求凸壳算法

- 识别x坐标最大和最小两个极点；
- 按照y坐标点集分成两部分；
- 调用求上/下凸壳算法；
- 并且得到结果.

第六章 并行算法的基本设计技术

6.1 划分设计技术
6.2 分治设计技术
6.3 平衡树设计技术
6.4 倍增设计技术
6.5 流水线设计技术

并行凸壳算法的一些文献

- 周启海，黄涛，吴红玉，基于双群双域四向水平倾角最小化圈绕的凸壳并行新算法，计算机科学，2008年2期
  - 本文针对现行凸壳算法(诸如:串行类的卷包裹凸壳算法、格雷厄姆凸壳算法等,并行类的折半分治凸壳算法、快速凸壳算法等)效率不高的缺点,根据同构化凸壳构造基本定理,利用工作站机群优点,提出了效率更高的双群(即:其机群分为2个子机群)、双域(即:其数据分布域分为2个子分布域)、四向(即:其每个子分布域内凸壳顶点的寻找方向均各自为顺时针、逆时针2个寻找方向)水平倾角最小化圈绕的凸壳并行新算法.
- 刘瑶，最小有向外接矩形算法的CUDA并行实现，大连理工大学 2014（学位论文）
- 毕硕本，陈东祺，颜坚，郭忆，基于二维凸壳的平面点集Delaunay三角网算法，计算机科学，2014年10期
- 颜坚，毕硕本，汪大，郭忆，多核架构下计算凸壳的并行算法，计算机科学，2013年2期

6.1 划分设计技术

6.1.1 均匀划分技术
6.1.2 方根划分技术
6.1.3 对数划分技术
6.1.4 功能划分技术

均匀划分技术

- 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.

Time Comlexity

- 在Phase b中，由於各處理器平行進行Quicksort，故所需時間為O(k log k)，其中k=n/p。
- 在Phase d中，對p2個資料做排序需要$O(p^2 \log p^2)$的時間。在Phase f中各個處理器對本地排序好的list做p-1次binary search(list的長度不大於k)，所以總共需要的時間為$O(p^2 \log p^2 + p \log k)$。
- 在Phase h中，各處理器所需要merge的大小不會超過2k，故Phase h可在O(2k log p)時間完成。
- 將上述結果相加，所需時間為$O(k \log k + k \log p + p \log k + p^2 \log p^2)$。當$n \geqq p^3$時，近似於$O(k \log k) = O((n/p) \log n)$，顯然是cost optimal的。

林育德,A Survey on Parallel Sorting by Regular Sampling (PSRS) ,台灣大學 R87921104

均匀划分技术

- 例6.1 PSRS排序过程。N=27，p=3，PSRS排序如下：

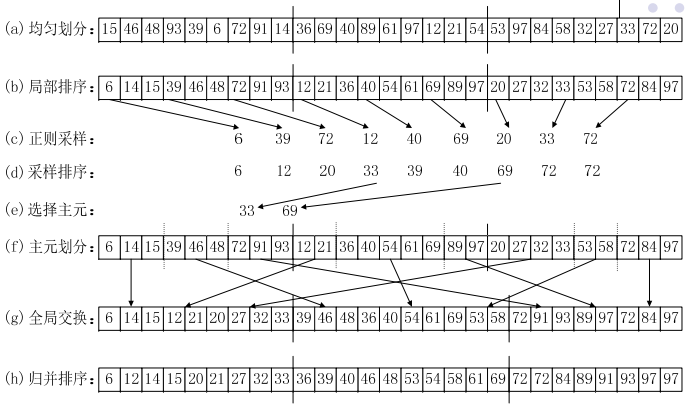


图6.1

Cost of a Parallel System

- Cost is the product of parallel runtime and the number of processing elements used ($p \times T_P$).
- Cost reflects the sum of the time that each processing element spends solving the problem.
- A parallel system is said to be *cost-optimal* if the cost of solving a problem on a parallel computer is asymptotically identical to serial cost.
- Since $E = T_S/pT_P$, for cost optimal systems, $E = O(1)$.
- Cost is sometimes referred to as *work* or *processor-time product*.

6.1.2方根划分技术

- 划分方法

  n个元素$A[1..n]$分成$A[(i-1)\times\sqrt{n}+1\cdots i\times\sqrt{n}]$, $i=1\cdots\sqrt{n}$

示例：SIMD-CREW模型上的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$i\lceil\sqrt{p}\rceil$ 和 $j\lceil\sqrt{q}\rceil$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)，直至组为空时递归结束；各组仍按$k=\lfloor\sqrt{pq}\rfloor$分配处理器

end.

分析：第4步递归归并时，原来的k台处理器是否够用？

设A和B中各段长度分别为p_i和q_i

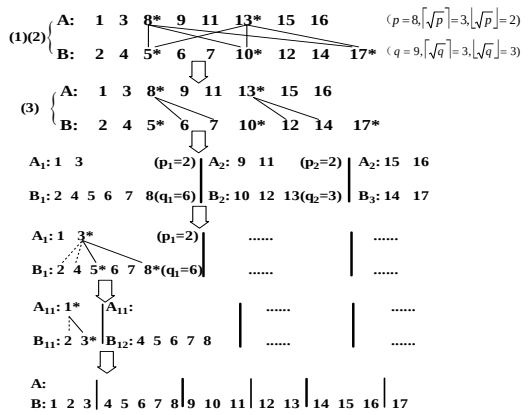
$$\sum p_i = p \quad \sum q_i = q$$

$$\sum\sqrt{p_i q_i} \le \sqrt{\sum p_i \sum q_i}$$

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

方根划分技术

- 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



时间复杂度分析

- 递归归并过程中，各段组中的有序序列均是A,B中的一段，所以从A的分段来看，每个归并中的两段中，至少有一段其长度不大于 $\lfloor\sqrt{p}\rfloor$
- 假定第i次递归归并中，某归并有一个序列长度为

$$R(i):\lambda_i$$

$$R(i+1):\lambda_{i+1}\le\lfloor\sqrt{\lambda_i}\rfloor$$

$$\lambda_0 = p \Rightarrow \lambda_i \le \lfloor p^{2^{-i}}\rfloor$$

$$i\le\log\log p + C \Rightarrow t_k(p,q) = 2\lceil\log\log p + C\rceil$$

分析

- 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≤ $\lfloor\sqrt{pq}\rfloor=k$；

注: $\lceil\sqrt{q}\rceil-1\le\lfloor\sqrt{q}\rfloor$

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$\Sigma p_i\le p$, $\Sigma q_i\le q$, 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。

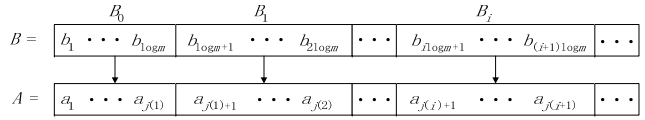
6.1.3对数划分技术

- 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：设有序序列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归并转化为(A_0、B_0)和(A_1、B_1)的归并

分析

- 算法分析

(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 $\lambda_i=$常数C 时终止递归，

即 $\lfloor p^{2^{-i}}\rfloor\ge$常数$C$ => $i\le\log\log p+$常数C_1

$p^{2^{-i}}\ge C$

$2^{-i}\log p\ge C'$

$2^iC'\le\log p$

$iC''\le\log\log p$

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

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

6.1.4功能划分技术

- 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：p148算法6.4

输入：A=(a1,…,an)；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

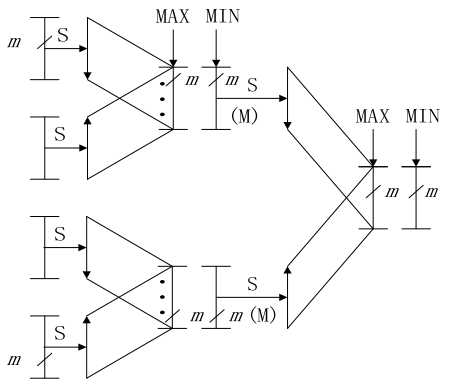


图6.3 (m-n)-选择过程

第六章 并行算法的基本设计技术

6.1 划分设计技术
6.2 分治设计技术
6.3 平衡树设计技术
6.4 倍增设计技术
6.5 流水线设计技术

- 陈剑骏，陈耀武，基于功能和数据划分的H.264并行解码算法，计算机工程，2012年12期
- 针对H264视援解码算法的并行模块选择、捌分及解码建度优化等阔燕，面向TilePt064多辕平台．提出一种可扩展的H 264并行解码算怯。对该算法的内部功雠块进行整合和捌分，根据桉问数据的依赣关系。动态分配功柏穰块爱优化算法并行教率。实验结果表明．该算法在解码效率、多核并行程度，解码时延等方面均有较好性拈，相比传统井行解码算j击，其并行加速比提高约25％。

功能划分的概念示意

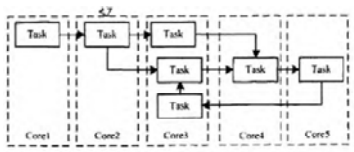


图2 宏块级并行的功能划分方法

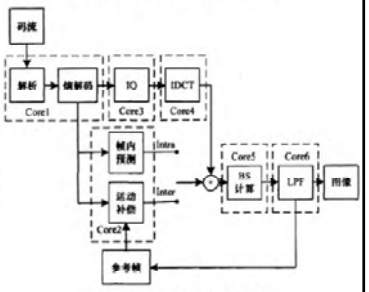


图4 解码功能的多核划分

6.5 流水线设计技术

6.5.1 设计思想
6.5.2 5-point DFT的计算

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；

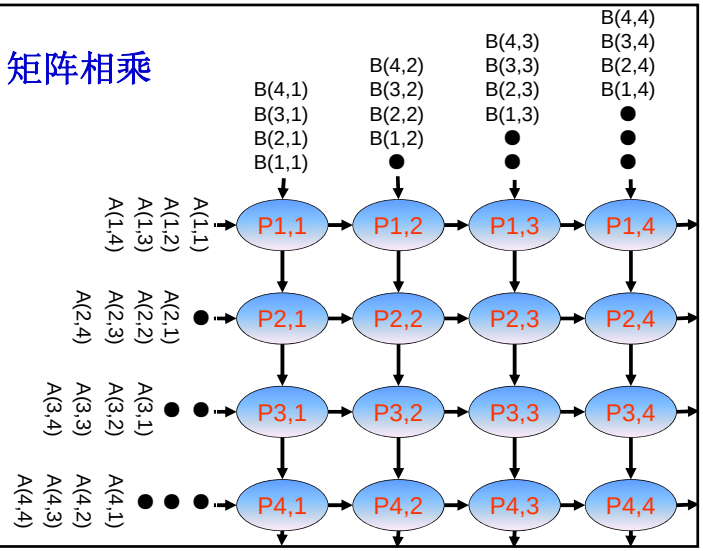
5-point DFT的计算

- 问题描述

5-point DFT的计算。应用秦九韶(Horner)法则，

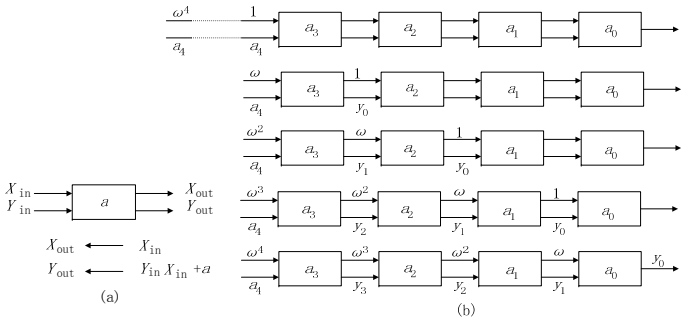
$$\begin{cases} y_0=b_0=a_4\omega^0+a_3\omega^0+a_2\omega^0+a_1\omega^0+a_0 \\ y_1=b_1=a_4\omega^4+a_3\omega^3+a_2\omega^2+a_1\omega^1+a_0 \\ y_2=b_2=a_4\omega^8+a_3\omega^6+a_2\omega^4+a_1\omega^2+a_0 \\ y_3=b_3=a_4\omega^{12}+a_3\omega^6+a_2\omega^4+a_1\omega^2+a_0 \\ y_4=b_4=a_4\omega^{16}+a_3\omega^6+a_2\omega^4+a_1\omega^2+a_0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

矩阵相乘



5-point DFT的计算

- 示例： p(n)=n-1, t(n)=2n-2=O(n)



第一次作业的题目

- 并行机模型部分
  1. 选取一种典型的多核处理器，基于并行机模型分析其体系结构的特点以及对并行计算的支持程度，要求给出具体性能参数。
  2. 众核处理器的并行机模型是什么？
  3. 除了课内介绍的互联网络拓扑以外，选取并行机中一到两种不同的典型拓扑结构，分析其特点及对并行计算的支持，要求给出具体性能参数。
  4. 研究一种通用的网络拓扑结构，使得典型拓扑都能够完美地或者接近完美地嵌入到这个网络中，并给出具体的嵌入表达式。
  5. MIMD的五种拓扑结构中，缺少量化参数，查阅资料，尽可能完整地提供这些参数值。

考试题

说明：

1. 任选一题，完成后书面提交，截至时间本学期第17周；
2. 题目中涉及的“并行计算”可能包括结构、模型、算法、性能、易用性等方面，所以思路要开阔；
3. 题目中涉及到要你进行选择的时候，一定要明确你的选择依据或者理由；
4. 答卷中需要包含分析或实验的内容，有新见解、完整性好或者论述充分者得满分。

第一次作业的题目

- 并行计算模型部分
  6. 除了PRAM、APRAM、BSP和LogP模型外，还有典型并行计算模型，选取两种进行模型描述，必要时举例说明。
  7. 用APRAM模型如何描述多核处理器，其精度如何？
  8. 选取一种并行计算模型，编写一个有一定规模的、具有实用意义的算法。给出这个算法的原理描述、具体描述以，并进行分析。
  9. 设计3D-Mesh上的排序算法，分别给出算法的原理描述和具体描述，并进行时间复杂度分析；
  10. 设计超立方体上的并行排序算法，分别给出算法的原理描述和具体描述，并进行时间复杂度分析；
  11. 编程实现PSRS算法，给出带注解代码和运行测试结果；

1. 试论述生物学系统中的并行性对并行计算的贡献。
2. 试论述工学系统中的并行性对并行计算的贡献。
3. 写出一个典型的博弈树搜索问题（或其他同等问题）的并行算法，编程实现，进行算法分析。
4. 写出一个典型的数据挖掘问题的并行算法，编程实现，进行算法分析。
5. 试从网络协议级对第八章的选路方法和开关技术进行解释。
6. 应用**PCAM**方法学设计一个并行应用（必须要是临界问题）。
7. 写出一个典型的博弈树搜索问题（或其他同等问题）的并行算法，给出基于**BSP**模型和**LogP**模型的算法描述，进行算法分析。
8. 写出一个典型的数据挖掘问题的并行算法，给出基于**BSP**模型和**LogP**模型的算法描述，进行算法分析。