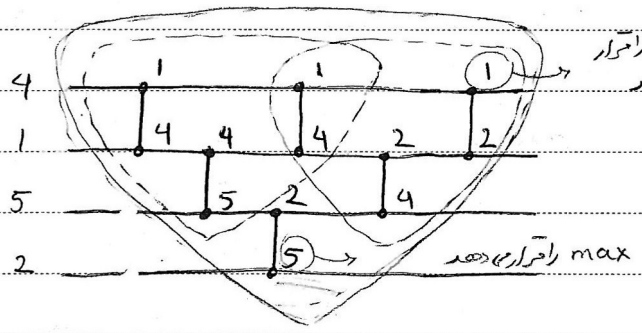
Subject 514
Date

معرفی الگوهایی که ایده‌ی آن نیست اما به صورت ساده می‌توان مدار را طراحی کرد:

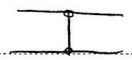
- شبکه مثلثی (insertion Sort / selection Sort)

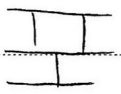
چون از ایده‌ی Selection Sort (max و min را ببرد سر جای خودش و یک مرحله کار را کم کن و بقیه کار را انجام بده) و insertion Sort (n-1 عضو اول را مرتب کن بعد یک نفر دیگه بیاد بقیه کن) استفاده کرده برآن insertion/selection Sort هم می‌گویند و چون به شکل مثلث دیده می‌شوند به آن شبکه مثلثی هم می‌گویند. در این روش اول max مشخص شد و بعد بقیه را مرتب کرد.

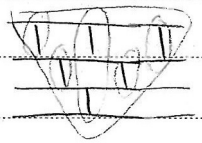


شکل) در هر مرحله باید شکل شبیه یک مثلث به

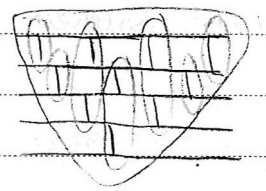
تحلیل مدار:

اگر $n=2$  $C=1$ $T=1$

اگر $n=3$  $C=3$ $T=3$

اگر $n=4$  $C=6$ $T=5$

$C(4)=1+2+3$

اگر $n=5$  $C=10$ $T=7$

$\rightarrow T(n) =$ بقیه حساب $= 2n-3 = \Theta(n)$

$C(n) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$

} با حالت ایده آل خیلی فاصله دارد پس روش خوبی نیست

n	2	3	4	5	6	...	n
T(n)	1	3	5	7	9		$T(n)=2n-3$
C(n)	1	3	6	10	15		$C(n)=\frac{n(n-1)}{2}$

$C(4)=1+2+3 \rightarrow C(n)=1+2+\cdots+n$

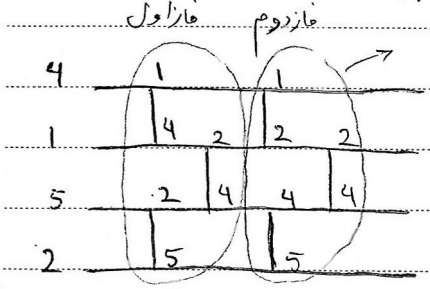
P4PCO

- مدار آجری یا زوج و فرد (odd_even Sorter)

تعداد لایه‌ها در این شبکه دقیقا n است. این مدار $\frac{n}{2}$ فاز دارد که هر فازی $n/2$ فاز که هر فاز 2 لایه ← تعداد لایه‌ها در کل = n

2 لایه متوالی (یک لایه زوج و یک لایه فرد) دارد، لایه‌ها هم حالت آجری و یکی در میون دارند.

در لایه‌ی اول هر عضو فرد با بعدیش مقایسه می‌شود (یکم با دوم، سوم با چهارم ...) و در لایه بعد هر عضو زوجی دارد با بعدیش مقایسه می‌شود (دوم با سوم، چهارم با پنجم ...) این کار در فاز بعدی تکرار می‌شود.



مثال) تحلیل مدار:

تعداد فاز = $\frac{n}{2}$ در هر فاز به تعدادی زوج و به تعدادی فرد هستند ← در مجموع تعداد مقایسه‌ها در هر فاز = n-1

$C(n) = \frac{n(n-1)}{2} = \Theta(n^2)$ ← هر ورودی داشته باشیم حداکثر با n مرحله مرتب می‌شود

نکته: C(n) متناسب (با ضریب ثابت متفاوت نشده (چون نسبت به هم ضریب ثابت از T(n) از 2 به یک تبدیل شده) اما لازم از ایده‌آل فاصله زیادی دارند

$T(n) =$ تعداد فاز $\times$ تعداد لایه در هر فاز $= \frac{n}{2} \times 2 = n$

- مدار Batcher یا Bitonic: Merge Sorter

این مدار یک پیش‌نیاز دارد. برای اینکه مدار Batcher را معرفی کنیم، ابتدا باید مدار Bitonic Sorter را معرفی کنیم، مدار کمکی merger را معرفی کنیم و بعد با داشتن این دو مدار دیگری مدار کمکی دیگر هم‌نیاز می‌توانیم merge Sorter را معرفی کنیم. در آخر مدار merge Sorter از Bitonic Sorter استفاده می‌شه. توجه شود که Bitonic S.

PAPCO

Subject 5-3
Date

یک مدار مرتب ساز در حالت کلی نیست و با شرایط خاصی مرتب سازی را انجام می‌دهد. اگر یک دنباله دونوا
یا وردی Bitonic به آن داده شود مرتب سازی را انجام می‌دهد، اگر نه معلوم نیست چه کار می‌کند!
جواب غلط می‌دهد.

دنباله Bitonic :

دنباله ای است که از یکی از خصوصیات زیر تبعیت کند:

۱- صعودی باشد (اگر هم مساوی و برابر باشند باز هم صعودی است) ۱ ۲ ۴ ۴ ۵

۲- نزولی باشد ۷ ۴ ۳ ۱ ۱

۳- صعودی بالا رود و بعد نزولی بیاید پایین مثال) ۱ ۲ ۵ ۷ ۴ ۳

۴- نزولی بیاید پایین و بعد صعودی بالا رود ۷ ۳ ۴ ۶

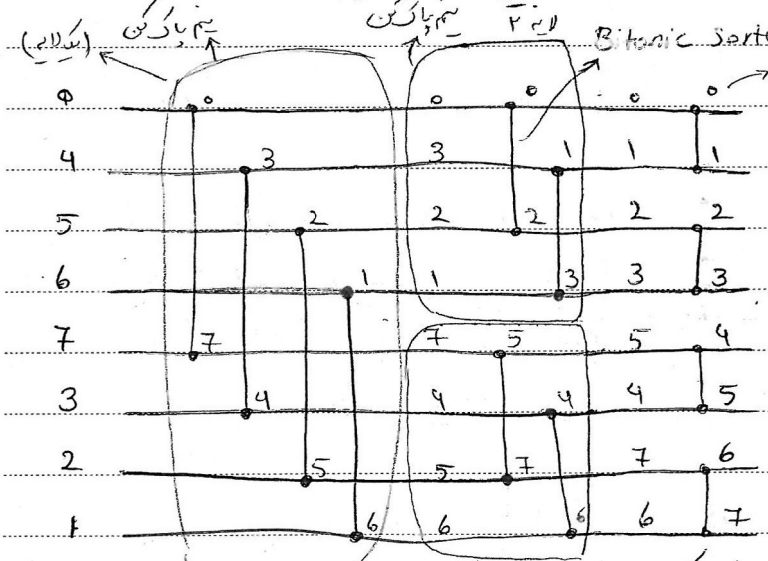
۵- اگر بایک یا چند Rotate به یکی از حالات فوق برسیم دنباله بازهم (دونوا Bitonic) محسوب می‌شود. مثال)

۱ ۲ ۳ ۴ ۶ ← rotate نقط از نوع ۵ یا ۵ ← ۱ ۲ ۵ ۷ ۴ ۳ ←

اگر دنباله (۰,۱) باشد حالت ۵ پیش میاد، ۴ حالت اول وجود دارد

مثال) مداری طراحی کنید که ورودی Bitonic بگیرد و آن را مرتب کند:

Bitonic Sorter ۸ تایی :



این مدار یک لایه نیم پاکن دارد بعلاوه
۲ تا Bitonic Sorter با ابعاد
کوچکتر (هرکدام $\frac{n}{2}$)

۵۰ درصد بالا از ۵۰ درصد پایین کوچکتر است
و هردو هم Bitonic هستند (قضیه)

نیم اول را به ۵۰ درصد دوم وصل کردیم!
خط i به خط $i+\frac{n}{2}$ وصل شد
نیم پاکن یک به هست چون سوئیچ باهم
خط‌ها را هم وصل می‌کند

تحلیل مدار : برای 8 ورودی 3 لایه داریم ← $\lg n$

یک لایه نیم پاک‌کن است که سوارزی با $n/2$ ها می‌کند

$$T(n)=\begin{cases}1+1\,T(n/2) & n>2\\ 1 & n=2\end{cases}$$

(1) موازی انجام می‌شود

$\rightarrow T(n)=\Theta(1.\lg n)$ master مسئله ۲

$=\Theta(\lg n)$

$$c(n)=\begin{cases}n/2+2\,c(n/2) & n>2\\ 1 & n=1\end{cases}$$

$n/2$ (H(n)) لایه اول مقایسه کننده دارد

$n/2$ تا Bitonic هم داریم

$\rightarrow h(n)=n^{\log_2^2}=n$

$g(n)=n/2$

$\lim_{n\to\infty}\frac{n/2}{n}=1/2$

پس $h(n)$ و $g(n)$ هم مرتبه‌اند

← پس می‌شه مسئله ۱ master

$$c(n)=\Theta(\lg n.h(n))=\Theta(n.\lg n)$$

* از لحاظ $T(n)$ و $c(n)$ ایده‌آل است

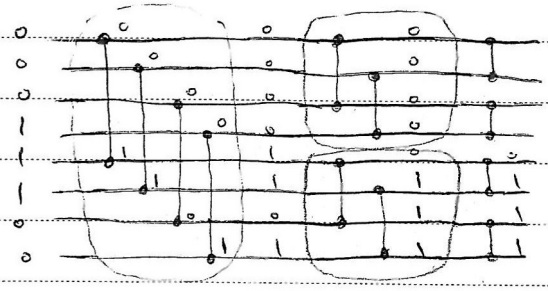
مشکل اساسی مدار Bitonic Sorter :

نقطه زمانی کار می‌کند که ورودی Bitonic باشد. بنابراین مدار کمکی است

و به صورت مستقل نمی‌تواند کار کند. پس در کنار merge Sorter استفاده می‌شود merge Sorter

ورودی را به Bitonic تبدیل می‌کند و بعد با Bitonic Sorter آن را مرتب می‌کند

اسم نیم پاک‌کن (up_cleaner) از کجا آمده؟



اگر ورودی به صورت صفر و یک باشد نیم پاک‌کن وسط مدار را به دو نصف تقسیم می‌کند.

اگر تعداد یک‌ها بیشتر باشد، ۵۰ درصد دوم کلا یک می‌شود و بعد باید نیمه اول را مرتب کنیم.

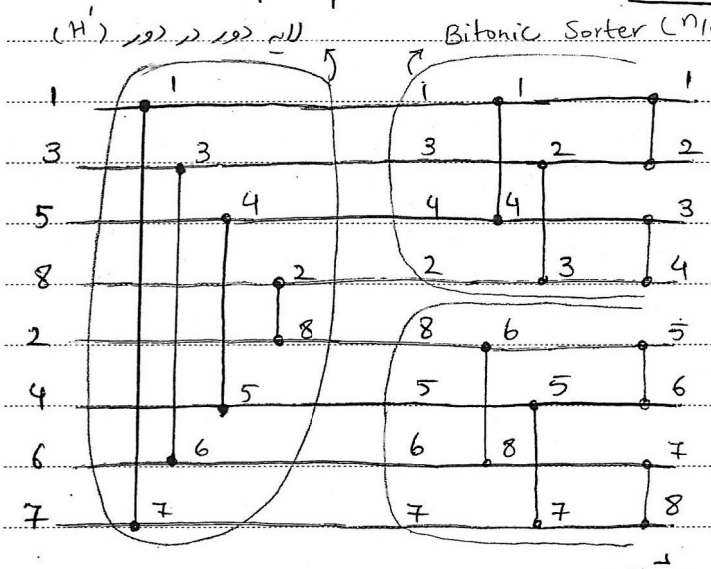
اگر تعداد صفرها بیشتر باشد، (مانند این مثال)، ۵۰ درصد اول کلا صفر می‌شود (پاک می‌شود) و بعد باید نیمه دوم را مرتب کنیم.

Subject 55

Date

مدار Merger:

دو دنباله (دنباله) مرتب شده را دریافت و آنها را با هم ادغام می‌کند.



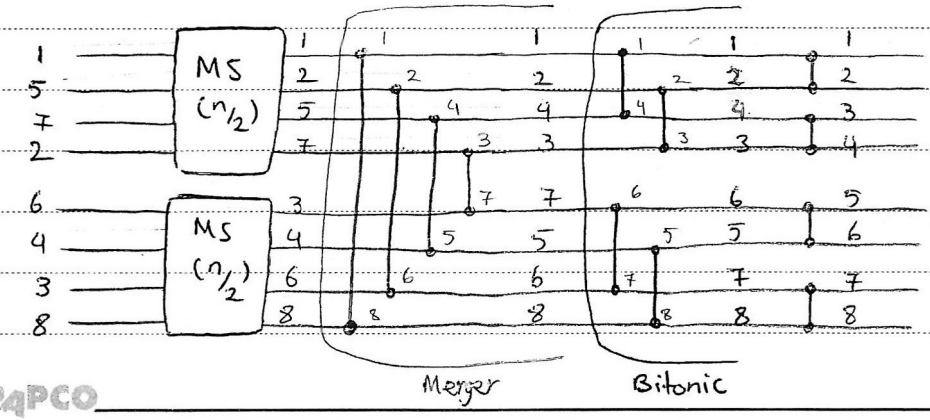
* در این مدار بجای یک کننده، یک لایه اولیه تحت عنوان "دور در دور" در نظر می‌گیریم. نتیجه مدار دور در دور این است که 50 درصد بالای کوچکتر از 50 درصد پایین می‌شود و همچنین هر دو Bitonic هشت تایی ورودی را به یک Bitonic چهارتایی مرتب می‌کنند و عمل مرتب‌سازی و merge انجام می‌شود.

$T(n) = \Theta(\lg n)$ $C(n) = \Theta(n \lg n)$

- در کل Merger n تایی از یک لایه "دور در دور" n تایی بعلاوه دو تا Bitonic $n/2$ تشکیل شده.

- تفاوت مدار Merger با مدار Bitonic این است که فقط بجای نیم پاک کننده اول، لایه دور در دور اضافه شده (جدیدا) فرق دارد اما در تعداد گروه‌ها هم فرقی ندارند) پس $T(n)$ و $C(n)$ دقیقا مثل مدار Bitonic Sorter می‌شود.

مدار Merge Sorter یا مدار Batcher: $(MS(n))$



PAPCO

تحلیل مقدار:

$$T_{MS}(n) = \begin{cases} 1\,T_{MS}(n/2) + \underbrace{T_M(n)}_{\lg n} & n>2 \\ 1 & n=2 \end{cases}$$

(1: سوارتی؛ T_M: زمان merge؛ MS: Bitonic M. Sorter)

$$T_{MS}(n) \xrightarrow{n>2 \text{ برای}} T_{MS}(n/2) + \lg n \rightarrow h(n) = n^0$$

$g(n) = \lg n \rightarrow$ برتر

$T(n) = \lg n \times g(n) = \lg(n) \times \lg(n) \leftarrow \lim = 1 \rightarrow$ master(>) مساوی

$$\rightarrow T_{MS}(n) = \boxed{\Theta(\lg^2 n)}$$

$$C_{MS}(n) = \begin{cases} 2\,C_{MS}(n/2) + \underbrace{C_M(n)}_{\Theta(n\lg n)} & n>2 \\ 1 & n=2 \end{cases}$$

$$C_{MS}(n) \xrightarrow{n>2 \text{ برای}} 2C_{MS}(n/2) + n\lg n \rightarrow h(n) = n^{\log_2^2} = n$$

$g(n) = n\lg n \rightarrow$ برتر ✓

$$\lim_{n\to\infty} \frac{2g(n/2)}{g(n)} = \lim \frac{2(\not{1/2}\lg n/2)}{\not{\lg n}}$$

$$= \lim \frac{\lg n/2}{\lg n} = 1 \rightarrow \text{master(>) مساوی}$$

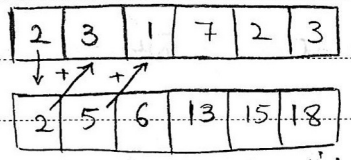
$$\rightarrow C_{MS}(n) = \Theta(\lg n) \times g(n) = \Theta(n\cdot\lg^2 n)$$

P4PCO

Subject 57
Date

محاسبه پیشوندی (Prefix computation) :

مثال) جمع پیشوندی



اعمال پیشوندی را روی (⊕) Semi group ها نیز می‌توانیم انجام دهیم.

- نمونه‌های از Semi group (⊕) های معروف :

+ , * , and , or , min , max , رشته‌ها Concat

خصوصیات ⊕ ها :

۱- خاصیت بسته بودن : $x, y \in D \rightarrow x \oplus y \in D$

یعنی اگر دو عضو از یک مجموعه باشند با عمل ⊕ هم باز از همان مجموعه هستند. مثلاً جمع دو عدد حقیقی، حاصل آن نیز یک عدد حقیقی است.

۲- خاصیت ترکیب پذیری :

$x \oplus y \oplus z = (x \oplus y) \oplus z = x \oplus (y \oplus z)$

۳- خاصیت عضو خنثی : $I \oplus x = x \oplus I = x$

عضو خنثی عمل + : ۰

* : ۱

AND : True

OR : False

min : $+\infty$ هر عددی با $+\infty$ برابر min یک عدد، همان عدد min رشته

max : $-\infty$

Concat : رشته ε، (رشته خالی)

PaPCO

۴- اگر $x \oplus y = y \oplus x$ زیر گروه ها جابجایی می نامیم ← اگر خاصیت جابجایی را داشته باشد

نکته: Concat رشته ها خاصیت جابجایی ندارد.

multi Thread راجب عنوان تابع scan در مرتبه ای prefix Computation -

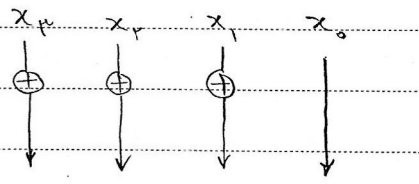
داشتیم، محاسبه به صورت نرم افزاری در زمان $\Theta(\lg^2 n)$ انجام می شود. جمع

پیشوندی کل محاسبات پیشوندی را به صورت لیست (آرایه) برمی گرداند.

محاسبه پیشوندی به صورت نخت افزاری

مدار محاسبه پیشوندی ← مدار Sequential
مدار موازی

مدار Sequential :



$$x_{0-2} = x_0 \oplus x_1 \oplus x_2$$

اگر n ورودی داشته باشیم:

(تعداد $\oplus$) تعداد گره ها

$$C(n) = n-1$$

و

(تعداد لایه ها) زمانی که صرف محاسبه می شود

$$T(n) = n-1$$

هر + در یک پالس انجام می شود ← خطی هست پس خوب نیست

مدار موازی: روش های مختلف وجود دارد، از جمله: روش تقسیم و غلبه،

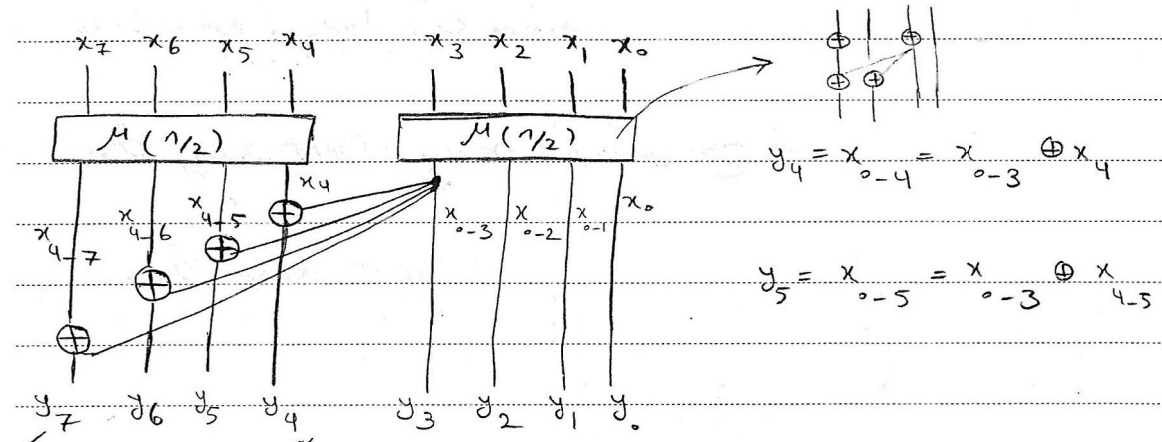
روش Brent_kong، روش kogg_stone، روش تقسیم و غلبه زوج و فرد

PAPCO

Subject 59.
Date

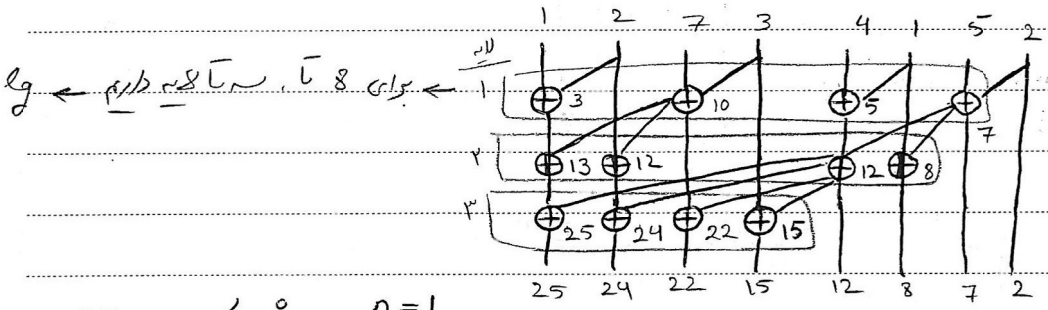
روش تقسیم و غلبه:

ایده ی این روش در تابع scan 2 (در مرتب ها) می باشد.



گروه های جدید همزمان در یک لایه انجام می شوند

اگر بخواهیم برای x_8 مدار را باز کنیم خواهیم داشت:



← برای 8 تا، سه تا لایه داریم ← lg

$$T(n) = \begin{cases} 0 & n=1 \\ 1\,T(n/2)+1 & n \geq 2 \end{cases} \longrightarrow T(n) = \lg n$$

تعداد لایه ها

$n \geq 2$ → مشابه master

↳ لایه آخر جمع زنی و همزمان انجام می شود

$$C(n) = \begin{cases} 0 & n=1 \\ 2\,C(n/2)+n/2 & n \geq 2 \end{cases} \longrightarrow C(n) = \Theta(n \lg n)$$

تعداد ⊕ ها

در لایه آخر $n/2$ ⊕ داریم

↳ $h(n) = n^{\log_2^2} = n \rightarrow g(n)$ هم مرتبه با $(n$ و $n/2$ هردو $\Theta)$ → مشابه master →

$C(n) = \Theta(\lg n \cdot n) = \Theta(n \lg$

$T_{\infty}C$ قبل Bitonic هست

مقدار $C(n)$ جزو بیشتر و هزینه بیت (اجرا) بالا هست

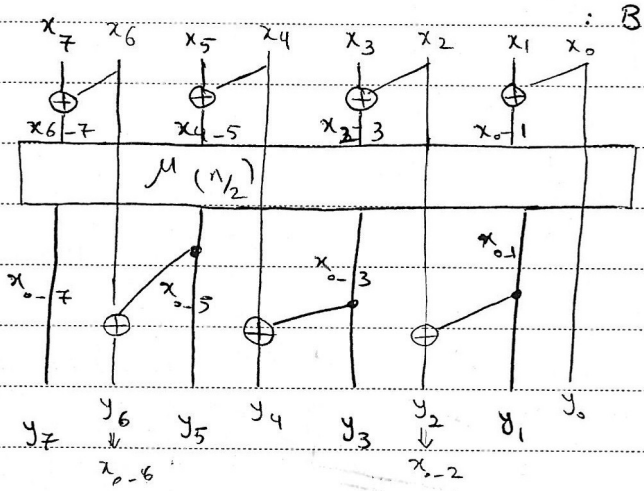
مشکل روش تقسیم و غلبه انتشار نرمن در مدار است (fan_out) که یک مشکل بیت افزار

Subject 60
Date

هست. (در مدار تخت افزاری) اگر بخواهیم از یک سیم استفاده بکنیم به چندین سیم دیگر، با افت ولتاژ مواجه می‌شویم و مدارها روشن نمی‌شوند (لذا در روش تقسیم و غلبه اگر مدار ۱۶ تایی باشد باید از یک سیم به ۸ جا استفاده بکنیم!)

روش Brent_kong برای حل این مشکل مطرح شد.

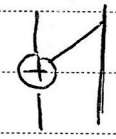
روش Brent_kong :



$y_2 = x_{0-1} \oplus x_2$

$y_4 = x_{0-3} + x_4$

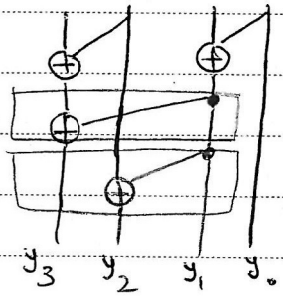
مدار دوتایی :



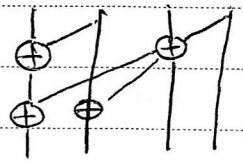
تعداد گیت $T(2) = 1$

تعداد گره $C(2) = 1$

مدار چهارتایی :



چون مدارهای ۲ تایی واسطه ندارند می‌توانیم موازی وهمزمان انجام دهیم ←



$T(4) =$

$C(4) =$

در کل :

$$T(n) = \begin{cases} 0 & n=1 \\ 1 & n=2 \\ 2 & n=4 \\ 1 + T(n/2) + 1 & n>4 \end{cases}$$

$T(n/2) + 2$

$$C(n) = \begin{cases} 0 & n=1 \\ 1 & n=2 \\ 4 & n=4 \\ n/2 + C(n/2) + (n/2 - 1) & n \end{cases}$$

$C(n/2) + n - 1$

Subject 6/8

Date

محاسبه مقدار $T(n)$ به صورت دقیق :

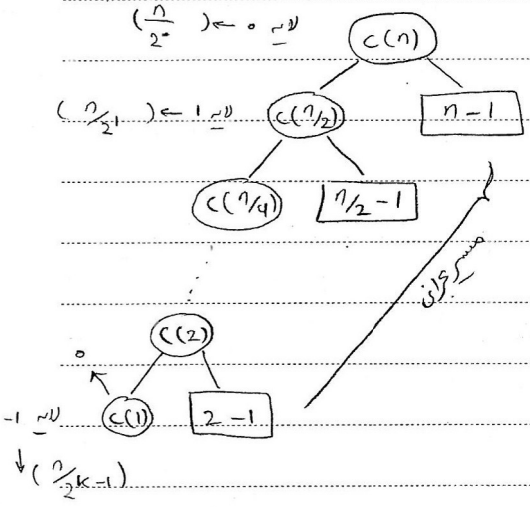
$h(n) = n^0 = 1 \qquad g(n) = 2 \rightarrow$ هم مرتبه اند $\rightarrow$ master مشمول ب

$\rightarrow T(n) = \Theta(h(n) \cdot \lg(n)) = \Theta(\lg n)$

$n \geq 4$ دقیق برای $\rightarrow 2\lg n - 2$ نسبت به مثل ضریب ثابت بیشتری دارد.

محاسبه مقدار $C(n)$ به صورت دقیق :

$C(n/2) + (n-1) \rightarrow$ master مشمول ج



$n = 2^k \quad , \quad k = \lg n$

جمع مستقیل ؛ $\quad C(n) = (n-1) + (n/2 - 1) + \cdots + (2-1)$ (2^k تا 2^1)

$= [n + n/2 + n/4 + \cdots + 2] - [k]$

(k جمله تصاعد هندسی)

تصاعد هندسی : $2 \times \frac{2^k - 1}{2 - 1} - k$

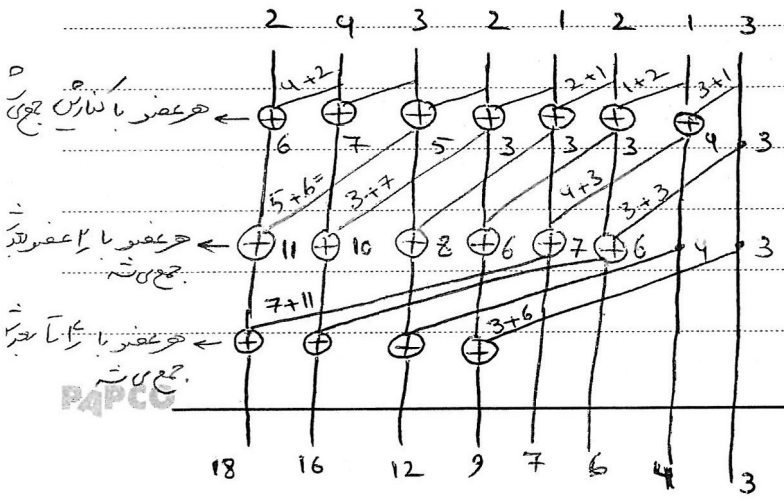
$= 2 \times (n-1) - \lg n$

خوبترین مدارهای ممکن افزار هست $\Rightarrow \boxed{C(n) = 2n - 2 - \lg n}$

در این روش در $C(n)$ پیچیدگی مدار کمتر شده ، fan_out زیاد می شود اما لایه ها بیشتر شده

و $c(n)$ هم بهبود داشته ایم و از مرتبه $\Theta(n)$ خواهد بود.

روش kogg-stone :



این روش از ایده بازگشتی استفاده می کند

$T(n) = \lg n$

تعداد لایه

$T(n) = \lg n$ — برای 8 تا ورودی، 3 لایه دارد

$c(n) = n\lg n - [\underbrace{1+2+\cdots+n/2}_{n-1}] = n\lg n - (n-1)$

توسط ۱، ۲ تا توسط اول یک ورودی را نمی‌بیند

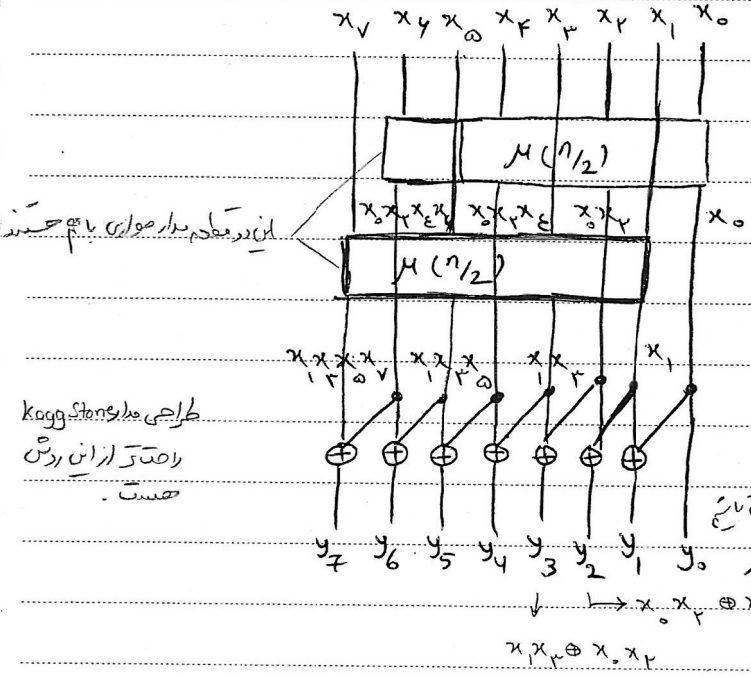
← اگر همه سطح‌ها را یک در نظر بگیریم

$\Rightarrow c(n) = \Theta(n\lg n)$

در این روش سیم بندی شلوغ، گره زیاد و هزینه ثابت افزاری بالا است اما تعداد لایه‌ها کم است و مشکل fan_out اصلا ندارد.

سرعت در این روش بالا هست اما هزینه ثابت افزاری بالا است.

- روش تقسیم و غلبه زوج و فرد:



این دو قطعه مدار صوری با هم هستند

طراحی مدار kogge stone راحت‌تر از این روش هست.

$x_1 \oplus (x_0 \oplus x_2) \stackrel{?}{=} (x_0 \oplus x_1) \oplus x_2$

← در صورتی که رشته باشد مورد قبول نیست

* بین این روش فقط برای نیم گره جا جایی مناسب است.

نقطه‌زنی جا جایی را نشان می‌دهد

$x_0 x_2 x_1 \equiv x_{0-2}$

$\mapsto x_0 x_2 \oplus x_1$ ← اگر رشته باشد مورد قبول نیست

$x_1 x_3 \oplus x_0 x_2$

چون خاصیت جا جایی ندارد

$$T(n) = \begin{cases} 0 & n=1 \\ 1T(n/2)+1 & n \geq 2 \end{cases}$$

← ۲ تا مدار هست اما صوری هستند، پس ۱ در نظر گرفتم

← لایه آخر صورت صوری است

$$c(n) = \begin{cases} 0 & n=1 \\ 2c(n/2)+(n-1) & n \geq 2 \end{cases}$$

← ۲ تا مدار نصف داریم

← لایه آخر

$\Rightarrow T(n) = \lg n$

$h(n) = n' = n$ — قسمت ب، master $h(n) \stackrel{\Theta}{\equiv} g(n)$

$c(n) = \Theta(n\lg n)$

P4PCO

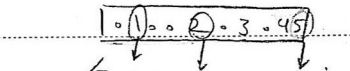
در این روش هم سیم بندی زیاد است اما مشکل fan_out نداریم.

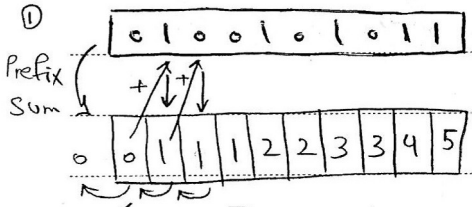
کاربردهای Prefix_computation :

prefix computation در زمینه های مختلف کاربرد دارد، از جمله در زمینه نرم افزار برای مثال در Complete Sort ، در آمار در توزیع تجمعی ، در پایگاه داده ها به صورت بازگشتی و با multi Thread انجام می شود ، در شبکه در آیند Routing و ...

مثال) پیدا کردن اولویت بیت های یک :

می خواهیم به این شکل دربیاید :





② هر عضوی را با بیت قبلی مقایسه می کنیم

اگر افزایش داشتیم عدد نوشته می شود

اگر نه صفر نوشته می شود

→ 0 1 0 0 2 0 3 0 4 5

$T(n) = \Theta(\lg n) + \Theta(1) = \Theta(\lg n)$

کپی کردن هر عضو با قبلی (موازی) کل در آیند prefix

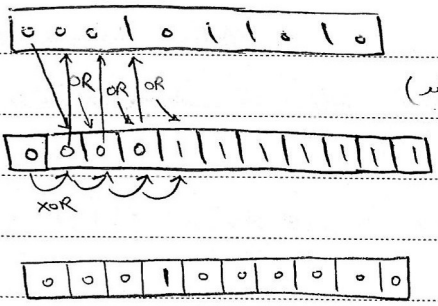
مثال) mask کردن بیت های بی ارزش (یا کم ارزش)

0 0 0 1 0 1 1 0 1 0 نقطه اولین یک از بیت چپ بماند و بقیه صفر شوند

نفی 0 0 0 1 . . . . . ←

از روش Diminished Prefix OR استفاده می کنیم.

Diminished یعنی با یک بیت فاصله یا تاخیر. پس در اینجا OR را با یک فاصله انجام می دهیم.



① در خروجی ۰ می‌گذاریم (که بایک تاخیر باشد) و بعد prefix OR می‌کنیم.

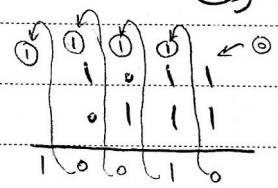
② هر خانه با یک بیت سمت راستش XOR می‌کنیم.

کاربرد دیگر: prefix computation

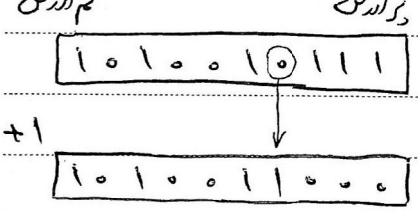
افزایش سرعت عمل جمع کردن

ADD کردن معمولی

full ADDER →



تمرین) عمل increment را بهبود دهید:



$\Theta(\lg n)$

FFT (Fast Fourier Transform) سوازی می‌باشد.

مقدمات ریاضی FFT:

۱- مفهوم چند جمله‌ای‌ها و اعمال روی آنها

۲- اعداد مختلط و توابع روی آنها

چند جمله‌ای‌ها: $4 + 3x + 5x^3 + 7x^4 + 3x^7$

هشت جمله‌ای هست اما بعضی از جملات (مثل ضریب x^5) صفر است.

برای ذخیره سازی به صورت لیست (آرایه) ذخیره می‌شود:

$\langle 4, 3, 0, 5, 7, 0, 0, 3 \rangle$

PAPCO

Subject 6-5-14
Date

اعمال روی چند جمله‌ای‌ها:

۱- ارزیابی ← جواب چند جمله‌ای به ازای یک x خاص

۲- ضرب داخلی، ضرب اسکالر، جمع و تفریق

۳- ضرب دو چند جمله‌ای

مثال برای ضرب داخلی)

$\langle 1, 2, 3 \rangle$

$\odot$

$\langle 1, 0, 4 \rangle$

$(1\times1) + (2\times0) + (3\times4) = 13$

$T(n) = O(n)$

* زمان تمام اعمال روی چند جمله‌ای‌ها به صورت sequential ، $O(n)$ ، $\Theta(n)$ است

مثال برای ارزیابی) چند جمله‌ای زیر را به ازای $x=2$ بنویسید:

$$4 + 3x + 5x^3 + 7x^4 + 3x^7$$

$\rightarrow \langle x^0, x^1, x^2, \dots, x^7 \rangle \rightarrow \langle 4, 3, 0, 5, 7, 0, 0, 3 \rangle$

به ازای x $\rightarrow \langle 2^0, 2^1, 2^2, \dots, 2^7 \rangle \xrightarrow{\odot} \langle 1, 2, 4, 8, 16, 32, 64, 128 \rangle$

ضرب داخلی $\rightarrow (4\times1) + (3\times2) + (0\times4) + (5\times8) + (7\times16) + (0\times32) + (0\times64) + (3\times128) =$

$T(n) = \Theta(n)$

به صورت بخت آزمایی می‌توان این کار را در زمان $\lg n$ انجام داد. به صورت زیر: اولین خانه یک و بقیه x می‌گذاریم $\rightarrow \lg(n)$

| 1 | x | x | ... | x |

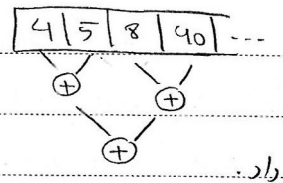
↓ prefix_multiply (prefix_mult)

| 1 | x | x^2 | x^3 | ... | x^{n-1} |

$\rightarrow \lg(n)$

PAPCO

جمع اعضای یک آرایه



$\lg(n)$

پس به صورت تخت افزاری تمام اعمال را در $\lg(n)$ به طور همزمان انجام داد.

بهترین روش sequential ارزیابی: روش Horner

ضرب چندجمله‌ای مثال مثل برای $x=2$ به روش Horner به شکل زیر محاسبه می‌شود:

$$4+x\Big[3+x\Big[0+x\Big[5+x\Big[7+x\Big[0+x\Big[0+\underbrace{x}_{2}\Big[3\Big]\Big]\Big]\Big]\Big]\Big]\Big] = 546$$

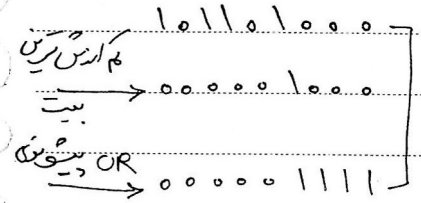
546 ← 542 ← 271 ← 268 ← 134 ← 134 ← 67 ← 62 ← 31 ← 24 ← 12 ← 12 ← 6 ← 6

ضرب دو چند جمله‌ای:

$$\begin{array}{l} (2+3x) \\ \times\ (3+4x) \\ \hline 6+9x \\ \quad 8x+12x^2 \\ \hline 6+17x+12x^2 \end{array}$$

$T(n) = O(n^2)$

جواب تمرین INC:



XOR = 1011001111 $\xrightarrow{\text{NOT}}$ 0100110000

P4PCO

Subject 67
Date

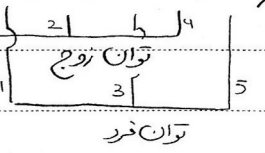
چند جمله‌ای‌ها

تفکیک یک چند جمله‌ای به دو چند جمله‌ای کوچکتر:

مثال) $f(x) = 4 + 3x + 5x^2 + 7x^3 + 2x^4 + 6x^5$

$= \langle 4, 3, 5, 7, 2, 6 \rangle$

چند جمله‌ای را به دو بخش تقسیم می‌کنیم: توان زوج و توان فرد



توان زوج $f_0(x) = \langle 4, 5, 2 \rangle = 4 + 5x + 2x^2$

توان فرد $f_1(x) = \langle 3, 7, 6 \rangle = 3 + 7x + 6x^2$

دقت از لیست میاریم بیرون به ترتیب از درجه صفر در x ضرب می‌شه

سؤال: $f(x)$ با $f_0(x)$ و $f_1(x)$ چه رابطه‌ای دارد؟

پاسخ: قابل اثبات است که:

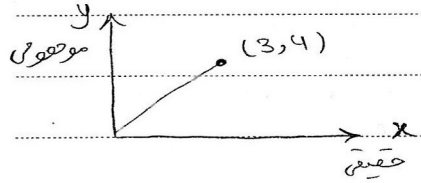
$$f(x) = f_0(\boxed{x^2}) + \boxed{x} \times f_1(\boxed{x^2})$$

f_0: توان‌های زوج

اعداد مختلط و موهومی:

اعداد موهومی: $i = \sqrt{-1} \rightarrow i^2 = -1$

$3 + 4i = (3, 4)$



به صورت زوج مرتب روی صفحه:

برای اعداد موهومی قدر مطلق هم می‌توان تعیین کرد:

$$|3 + 4i| = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

PAPCO

مثال) ضرب دو عدد موهومی :

$$(3+4i)(2+3i) = 6 + 9i + 8i + 12\,(i^2 \to -1)$$
$$= -6 + 17i$$

حالت مزدوج :

$$(3+4i)(3-4i) = \overbrace{3\times 3}^{9} + \overbrace{4\times 4}^{16} = 25$$

مثال) تقسیم دو عدد موهومی :

$$\frac{2+3i}{3-4i} =$$ کافی است دو طرف را در مزدوج مخرج ضرب کنیم

$$= \frac{(2+3i)(3+4i)}{(3-4i)(3+4i)} = \frac{-6+17i}{25} = \frac{-6}{25} + \frac{17}{25}i$$

تابع Exponential :

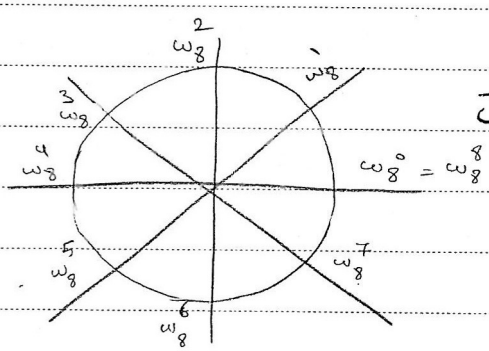
$$e^{a+bi} = e^a\left[e^{bi}\right]$$
$$= e^a\left[\cos(b) + i\sin(b)\right] = \overbrace{e^a\cos(b)}^{\text{قسمت حقیقی}} + \overbrace{e^a i\sin(b)}^{\text{قسمت موهومی}}$$

مثال) $e^{2\pi/n\,ki} = \cos\left(\frac{2\pi k}{n}\right) + i\sin\left(\frac{2\pi k}{n}\right)$ [قسمت حقیقی ندارد]

- معادله $x^n = 1$ دارای n ریشه است که این n ریشه ها به ترتیب عبارتند از :

$$x \in \left\{\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}\right\},\quad \omega_n = e^{2\pi/n\,i}$$

مثال) $x^8 = 1$

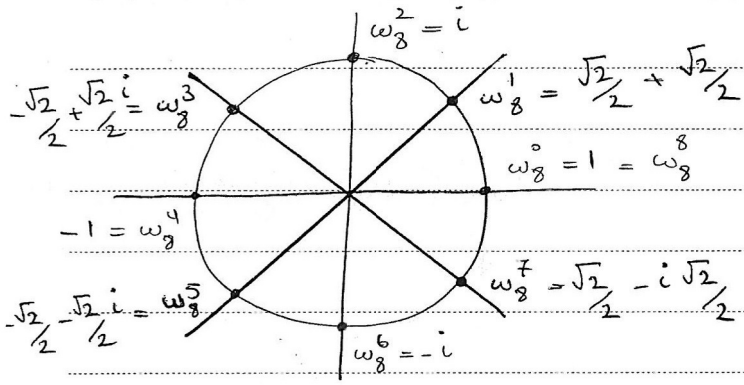


ابتدا روی صفحه مختصات دایره را به 8 قسمت تقسیم می کنیم

PAPCO

Subject 69

Date



$\omega_8^1 = e^{2\pi/8 i}$ $\pi/4 = 45^\circ$

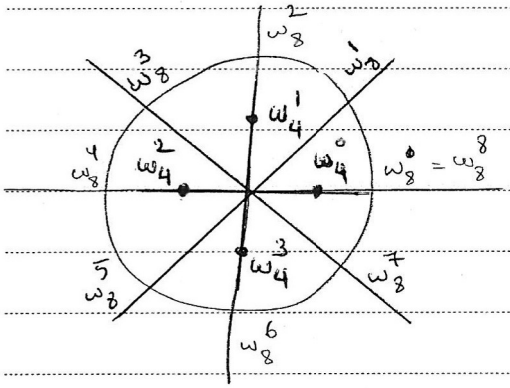
$= \cos\left(\frac{2\pi}{8}\right) + i \sin\left(\frac{2\pi}{8}\right)$

$= \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}$

$\omega_8^2 = \left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)\left(\frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}\right)$

$= \frac{1}{2} + \frac{1}{2} i + \frac{1}{2} i - \frac{1}{2} = i$

- اگر دایره را به چهار قسمت تقسیم کنیم ω_4^* یا $x^4 = 1$ هم به دست می‌آید:



$\omega_4^0 = \omega_8^0 = 1$

$\omega_4^1 = \omega_8^2 = i$

$\omega_4^2 = \omega_8^4 = -1$

$\omega_4^3 = \omega_8^6 = -i$

DFT (تبدیل فوریه گسسته): (Discrete Fourier Transform)

DFT سیستمی از اعداد حقیقی یا به اعداد مختلط و موهومی تبدیل می‌کند

مثال) $A(x) = \langle 1, 2, 3, 4 \rangle = 1 + 2x + 3x^2 + 4x^3$

$DFT(A) = B(x) = \langle b_0, b_1, b_2, b_3 \rangle$

* n مجموعه توانی از 2 است، در اینجا $x^4 = 1, n = 4$ در نظر گرفته شده، پس داریم:

PAPCO

مرحله اول) مجموعه جواب $n=4$ را به دست می‌آوریم: $\{1, i, -1, -i\} \leftarrow x^4 = 1$

یعنی ریشه‌ها را به دست می‌آوریم

مرحله ۲) ریشه‌ها را در تابع ازا گذاری می‌کنیم:

$b_0 = A(1) = 1+2+3+4 = \boxed{10}$

$b_1 = A(i) = 1 + 2i + 3(i^2) + 4(i^3) = 1 + 2i - 3 - 4i = \boxed{-2-2i}$ ($i^2 = -1$, $i^3 = -i$)

$b_2 = A(-1) = 1 - 2 + 3 - 4 = \boxed{-2}$

$b_3 = A(-i) = 1 + 2(-i) + 3(-i)^2 + 4(-i)^3 = 1 - 2i - 3 + 4i$ ($(-i)^2 = -1$, $(-i)^3 = i$)

$= \boxed{-2+2i}$

مرحله ۳) ضابطه جواب‌های به دست آمده در بردار جواب مقابل DFT(A) :

$DFT(A) = \langle 10, -2-2i, -2, -2+2i \rangle$

$= 10 - 2 - 2x - 2x^2 - 2 + 2x^3$

نکته: جمع ریشه‌ها صفر می‌شود. $\sum_{i=0}^{n-1} \omega_n^i = 0$

اثبات: راه اول) هر عضو یک قرینه دارد پس مجموعش صفر می‌شود.

راه دوم) تقاعد هندسی: $\frac{\omega_n^n - 1}{\omega_n - 1}$, $\omega_n \neq 1$

$= \frac{1-1}{\omega_n - 1} = 0$

قرینه یابی: فرض: n زوج باشد

$-\omega_n^k = \omega_n^{n/2 + k}$

مثلاً $n=8$, $k=1$ $\rightarrow$ $-\omega_8^1 = \omega_8^5$

$\omega_8^1 = \sqrt{2}/2 + \sqrt{2}/2\, i$

$\omega_8^5 = -\sqrt{2}/2 - \sqrt{2}/2\, i$

P4PCO

Subject 71
Date

قضیه: ω_n^k در هر ضریبی که ضرب شود، روی جوابها هیچ تأثیری ندارد.

$$\omega_n^k = \omega_{nd}^{kd}$$

زمان محاسبه DFT به صورت Sequential:

زمان ارزیابی $\Theta(n)$ هست، و n تا ریشه داریم پس کلاً می‌شود $\Theta(n^2)$

زمان محاسبه به صورت موازی: باید بتوانیم به صورت تخت افزاری و موازی در زمان $\Theta(n \lg n)$ محاسبه کنیم. می‌توان در زمان $\Theta(\lg n)$ هم محاسبه را انجام داد اما خیلی هزینه بر و زمان بر است (پیچیدگی تخت افزاری (work) زیاد است چون به ازای هر ریشه یک مدار لازم داریم که ارزیابی و محاسبات را انجام دهد.)

سوال) چطور با تخت افزار کمتر این محاسبات را انجام دهیم؟ به کمک FFT

DFT^{-1} (معکوس DFT)

$$DFT^{-1}(B) = A$$

لذوری لیست نتیجه باید لیست اولیه را به دست آوریم.

مثال) $B = \langle 10, -2-2i, -2, -2+2i \rangle$

مرحله اول) از واردون ریشه‌ها استفاده می‌کنیم (در حالت اصلی واردون سیاهت ریشه‌ها را می‌خوانیم ↺
در حالت معکوس به صورت سیاهت کرد ریشه‌ها را می‌خوانیم ↻)

برای x^4 داریم 1 i -1 $-i$: $\langle 1, i, -1, -i \rangle$

P4PCO

Subject 72
Date

مرحله دوم) معکوس بردار را ارزانگذاری می‌کنیم:

$B = 10 + (-2-2i)x - 2x^2 + (-2+2i)x^3$

↓ ارزانگذاری

$B(1) = 10 + (-2-2i) - 2 + (-2+2i) = 10 - 6 = \boxed{4}$

$B(-i) = 10 + (-2-2i)(-i) - 2(-i)^2 + (-2+2i)(-i)^3$

$10 + 2i + 2i^2 + 2 - 2i + 2i^2 = 10 + 2i - 2 + 2 - 2i - 2 =$

$B(-1) = 10 + (-2-2i)(-1) - 2(-1)^2 + (-2+2i)(-1)^3$

$10 + 2 + 2i - 2 + 2 - 2i = \boxed{12}$

$B(i) = 10 + (-2-2i)(i) - 2(i)^2 + (-2+2i)(i)^3$

$10 - 2i + 2 + 2 + 2i + 2 = \boxed{16}$

مرحله سوم) جواب‌ها را در $\frac{1}{n}$ ضرب می‌کنیم (اینجا $n=4$ یعنی $*\frac{1}{4}$)

$A(x) = \langle \frac{4}{4}, \frac{8}{4}, \frac{12}{4}, \frac{16}{4} \rangle = \langle 1, 2, 3, 4 \rangle$

الگوریتم FFT:

تبدیلی که به صورت Sequential در زمان $n \lg n$ عملیات DFT را انجام می‌دهد.

از ایده‌ی merge sort و روش تقسیم و غلبه استفاده کرده.

مثال)

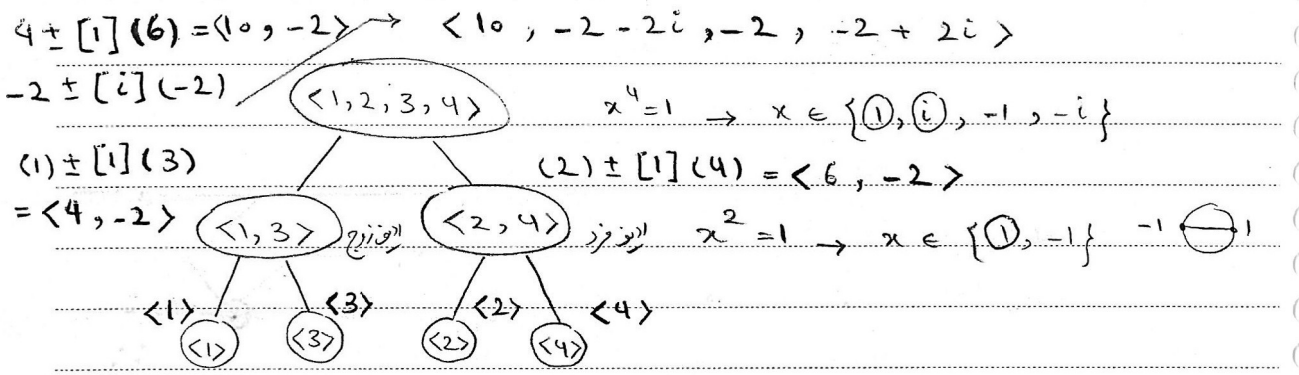
$B = \langle 10, -2-2i, -2, -2+2i \rangle$

$\langle 1, 2, 3, 4 \rangle$ $n=4$

P4PCO

Subject 7B

Date



نحوه محاسبه:

$u_i \pm \omega_n^i \cdot v_i$ ← قطعه‌پردازنای (هزدوج)

i: زیست زوج، ω_n^i: ریشه، v_i: زیست فرد

مثال) ⟨1, 1, 2, 0, 3, 0, 4, 0⟩

مرحله اول) اول صوابها را می‌نویسیم

$x^8 = 1 \rightarrow \{1, \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i, i, -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i, -1, -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i, -i, \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i\}$

$10 \pm [1](1) = \langle 11, 9\rangle$

$-2-2i \pm [\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i](1) = \langle -2 + \frac{\sqrt{2}}{2} + i(-2 + \frac{\sqrt{2}}{2})\rangle$

$-2 \pm [i](1)$

$-2+2i \pm [-\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i](1)$

$\langle 10, -2-2i, -2, -2+2i\rangle$

⟨1,1,2,0,3,0,4,0⟩ → جواب‌ها: $\langle 11, (-2+\frac{\sqrt{2}}{2} + i(-2+\frac{\sqrt{2}}{2}), -2+i, -2-\frac{\sqrt{2}}{2} + i(2+\frac{\sqrt{2}}{2}), 9, \dots\rangle$

$(1) \pm [1](0) = \langle 1, 1\rangle$

$(1) \pm [i](0) = \langle 1, 1\rangle$ } ⟨1,1,1,1⟩

⟨1,2,3,4⟩

⟨1,0,0,0⟩

$x^4 = 1 \rightarrow \{1, i, -1, -i\}$

$(1) \pm [1](0) = \langle 1, 1\rangle$

قبلا حساب کردیم

$(0) \pm [1](0) = \langle 0, 0\rangle$

⟨1,0⟩ ⟨0,0⟩

$x^2 = 1 \rightarrow \{1, -1\}$

⟨1⟩ ⟨0⟩ ⟨0⟩ ⟨0⟩

⟨1⟩ ⟨0⟩ ⟨0⟩ ⟨0⟩

* تا ضعف ریشه‌ها را پیشتر نیاز نداریم.

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ \Theta(n) + 2T(n/2) + \Theta(n/2) & n \geq 2 \end{cases}$$

seq

تقسیم — FFT تا — غلبه

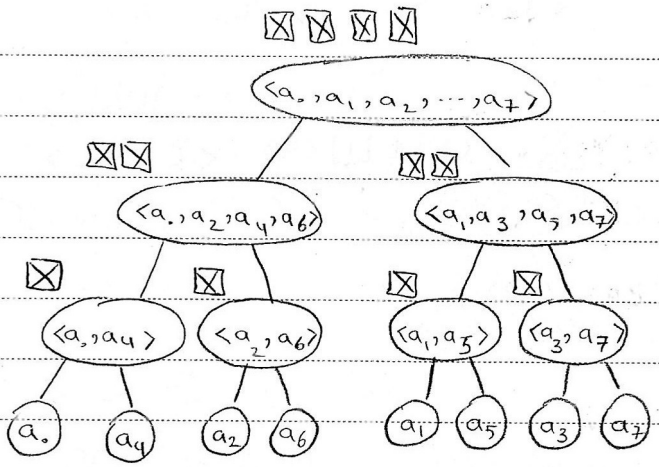
PqPCO

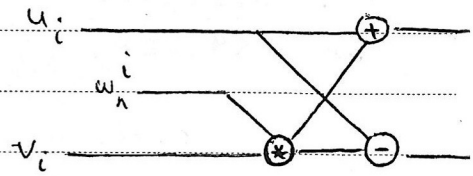
$T(n) = 2T(n/2) + \Theta(n) \xrightarrow{\text{master}} \Theta(n \lg n)$

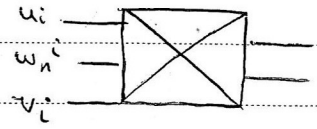
Subject #4

Date

محاسبه پروانه ای :







مختصار:

تعداد کل محاسبات پروانه ای $= \lg n \times n/2$

- در روش FFT می توانیم محاسبات را طوری انجام دهیم که داده ها اصلاً با شند و فقط یک آرایه ورودی و یک آرایه خروجی داشته باشیم. یک آرایه از ورودی ها می سازیم (ورودی ها همان داده های لایه آخر هستند) و خروجی را با استفاده از فرمول $u_i + w_n^i v_i$ محاسبه کرده و جواب ها را در لیست متناظری ذخیره می کنیم. پس کلا با ۲ آرایه می توان کل محاسبات شکل بالا را انجام داد.

- در هر ردیف (دقیقا n ورودی و n خروجی داریم. مشکل پیدا کردن و درست کردن آرایه ورودی (لایه آخر لیست) است. چون اعداد به هم ریخته هستند ($a_0, a_4, a_2, \ldots$) پس باید بیاییم این اعداد از کجا آورده اند. برای این کار از Bit Reverse استفاده می کنیم. به این صورت که اعداد 0 تا $n-1$ را به صورت بیتی می نویسیم و بعد بیت ها را معکوس می کنیم.

PAPCO

Subject ۹۷۵

Date .

مثال)

(0) 000 —Rev→ 000 (0) a_0

(1) 001 → 100 (4) a_4

(2) 010 → 010 (2) a_2

(3) 011 → 110 (6) a_6

(4) 100 → 001 (1) a_1

(5) 101 → 101 (5) a_5

(6) 110 → 011 (3) a_3

(7) 111 → 111 (7) a_7

با توجه به خروجی Bit Reverse می‌فهمیم که در آرایه ورودی باید اول a_0، بعد a_4 و ... را قرار دهیم.

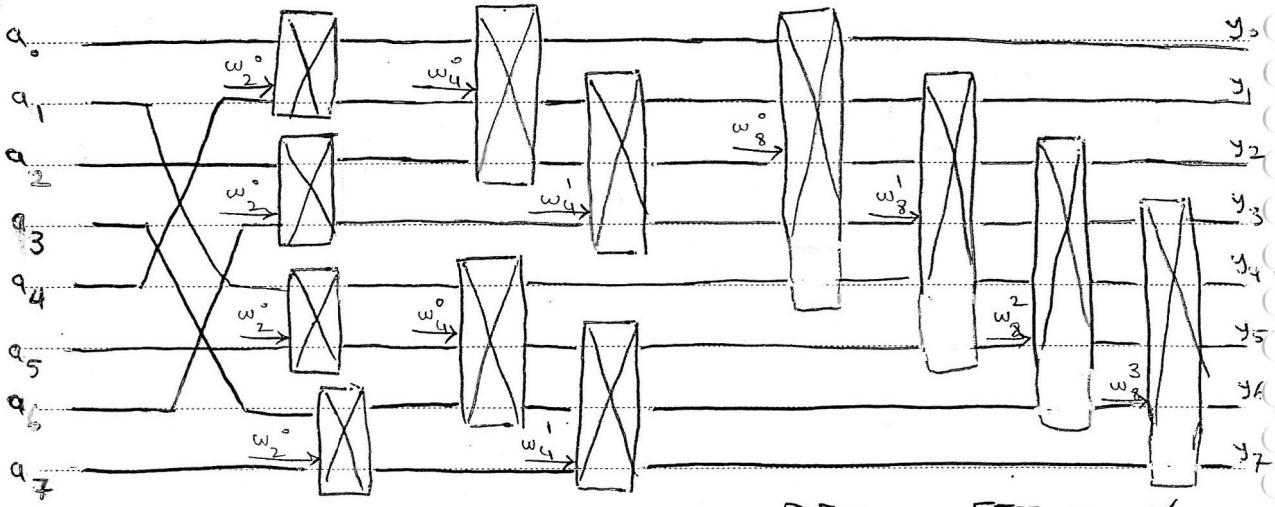
مرتبه زمانی محاسبه پروانه‌ای: (مدار نخ‌قرار) $W(n) = \frac{n}{2} \lg n = \Theta(n \lg n)$

تعداد لایه‌های که پروانه دارند ← | → تعداد پروانه

مدار نخ‌قرار برای محاسبه مولفه‌ها بدون نیاز به ضرب مولفه‌ها $T(n) = \lg n$ ← تعداد دفعاتی که پروانه‌ها به صورت موازی محاسبه می‌شوند = تعداد لایه‌های که پروانه دارند

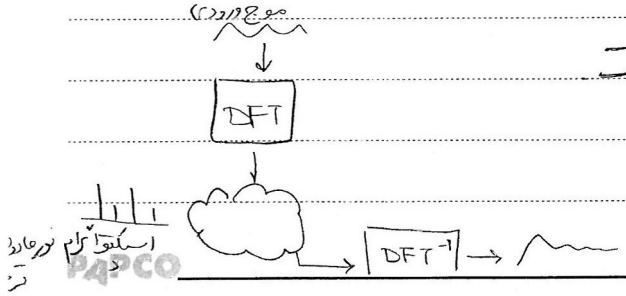
* محاسبات پروانه‌ای از لحاظ داده به هم وابستگی ندارند و به صورت موازی محاسبه می‌شوند.

مدار نخ‌قرار: $W(n) = 2W(n/2) + n/2$ $T(n) = T(n/2) + 1$



کاربردهای FFT و DFT:

در حوزه برق و سیگنال بسیار کاربرد است.



Subject 76
Date

کاربرد FFT در محاسبات سنگین: (مثلاً برد برنامه هزار بار ضرب دو چند جمله‌ای)

- ضرب چند جمله‌ای:

الگوریتم FFT نیز مولی برای ضرب دو چند جمله‌ای استفاده کنیم برتبه آن بجای $\Theta(n^2)$ ، $\Theta(n \lg n)$ خواهد شد.

(مثال) $1+2x$ $\xrightarrow{\text{به صورت آرایه}}$ $A = \langle 1, 2 \rangle = \langle 1, 2, 0, 0 \rangle$ توان x^3, x^2, x ۳

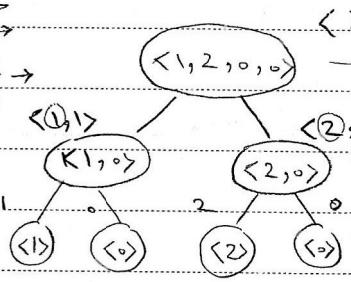
$\times$ $2+3x$ $\longrightarrow$ $B = \langle 2, 3 \rangle = \langle 2, 3, 0, 0 \rangle$ $\longrightarrow$ طول آرایه را بجای n به $2n$ تبدیل کردیم

n

چون جواب ضرب n جمله‌ای در n جمله‌ای، $2n-1$ جمله‌ای خواهد بود

حال FFT را حساب می‌کنیم

برای A $FFT_4(A)$ $\longrightarrow$ $\langle 3, 1+2i, -1, 1-2i \rangle$



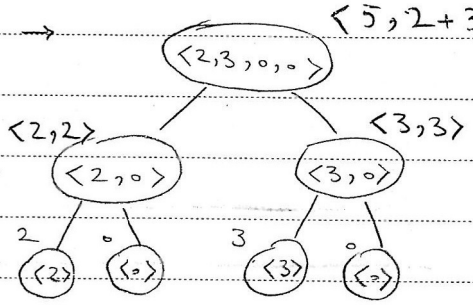
$x^4 = 1 = \{1, i, -1, -i\}$

$x^2 = 1 = \{1, -1\}$

سطح ۲: $1 \pm [1](0) = \langle 1, 1 \rangle$ ، $2 \pm [1](0) = \langle 2, 2 \rangle$

سطح x^4: $1 \pm [1](2) = \langle 3, -1 \rangle$ ، $1 \pm [i](2) = \langle 1+2i, 1-2i \rangle$

برای B $FFT_4(B)$ $\longrightarrow$ $\langle 5, 2+3i, -1, 2-3i \rangle$



سطح x^2: $2 \pm [1](0) = \langle 2, 2 \rangle$ ، $3 \pm [1](0) = \langle 3, 3 \rangle$

سطح x^4: $2 \pm [1](3) = \langle 5, -1 \rangle$ ، $2 \pm [i](3) = \langle 2+3i, 2-3i \rangle$

ضرب نقطه به نقطه $\longrightarrow$ $\langle 3, 1+2i, -1, 1-2i \rangle$ ، $\langle 5, 2+3i, -1, 2-3i \rangle$

$1+2i$ / $2+3i$ / $2+4i$ / $3i+6i^2$ / $2+7i-6 = -4+7i$

$c' = \langle 15, -4+7i, 1, -4-7i \rangle$

از حاصلضرب FFT^{-1} می‌گیریم

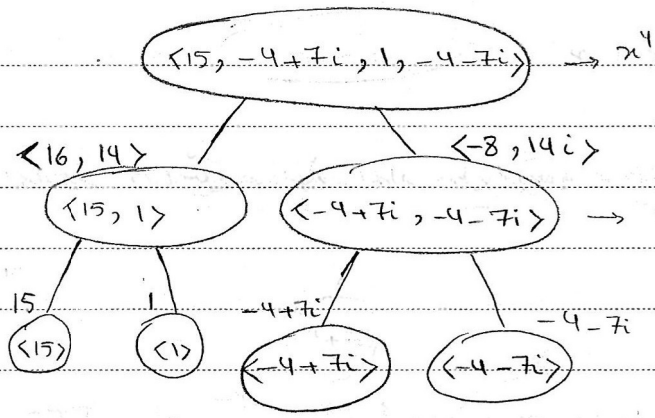
PAPCO

Subject ۶۷۷

Date

$\langle 8, 28, 24, 0 \rangle$

$FFT_4^{-1}(c')$ ←



* توجه کن inverse هست.

سطح x^2 : $15 \pm [+1](1) = \langle 16, 14 \rangle$

$-4 \pm [+1](-4-7i) = \langle -8, 14i \rangle$ $\quad (+) = -4+7i+4+7i = 14i$

$(-) = -4+7i-4-7i = -8$

سطح x^4 : $16 \pm [1](-8) = \langle 8, 24 \rangle$

$14 \pm [-i](14i) = \langle 28, 0 \rangle$ $\quad (+) = 14 - 14(-i)^2 = 0$

$(-) = 14 + 14(-i)^2 = 28$

$\rightarrow FFT_4^{-1}(c') = \frac{1}{4} * \langle 8, 28, 24, 0 \rangle = \langle 2, 7, 6, 0 \rangle$

ضرایب حواب‌های ضرب چند جمله‌ای $\longrightarrow 2 + 7x + 6x^2 + 0x^3$

تحلیل :

حرف برای گفتن ندارد

اضافه کردن صفرها $= \Theta(n)$ هست. $2n$ اما اینجا ورودی $n \lg n$ زمان FFT خود

$FFT_{2n}(B), FFT_{2n}(A) = 2\Theta(2n \cdot \lg(2n))$ ← دو تا FFT داشتیم

$= 2\Theta(2n(\lg n + 1)) = \Theta(n \lg n)$

ضرب نقطه به نقطه : $\Theta(2(n)) = \Theta(n)$ → حرف برای گفتن ندارد

$FFT_{2n}^{-1}(c') : \Theta(2n \lg(2n)) = \Theta(n \lg n)$

کل زمان الگوریتم $= \Theta(n \lg n)$ → چون $\Theta(n)$ دیدار $\Theta(n \lg n)$ حرف برای گفتن ندارد.

P4PCO

Subject ۷ 78
Date

- هر عدد بزرگی را می‌توان به صورت یک چند جمله‌ای نوشت که x آن ۱۰ است. درواقع در زمان محاسبه اعداد بزرگ در ماشین حساب، اعداد تبدیل می‌شوند به چند جمله‌ای:

مثال) $427 = \langle 4, 2, 7 \rangle = 4 + 2x + 7x^2 = 4 + (2 \times 10) + (7 \times 10^2)$

تمرین: مسأله FFT را به صورت یک برنامه multi thread بنویسید:

گام اول) پیاده‌سازی تابع Prefix Multiply به کمک ایده‌ی تابع جمع‌زنی Brent_kong برای این تابع بنویسید.

گام دوم) تابع محاسبه ریشه‌ها: Calculate Roots ()

$w^i \rightarrow Roots[i * step]$

{ P | 1 | w | w^2 | ... | w^{n-1} |
0 1 n-1

Prefix Multiply (n, p, Root)

| 1 | w | w^2 | ... | w^{n-1} |

گام سوم) تابع FFT

```
void FFT ( int n, Vector A, Vector B, step)
{ ... }

main ( ... )
{ A و n را بگیر
  w = Complex ( cos(2π/n) , sin(2π/n) )
  Prefix Computation ( );
  Calculate Roots (n, w, R);
  FFT (n, A, B, 1)
  B را چاپ کن
}
```

PAPCO

Subject A79

Date

Distributed Shared Memory یا شبکه‌های چند منظوره یا Graph Model (Distributed Shared Memory)

Graph Model ها شبکه‌هایی هستند که برای اینکه فقط یک الگوریتم خاص روی آنها کار کند، چندین الگوریتم را می‌توانیم روی این شبکه اجرا کنیم.

Distributed Shared Memory: هرکدام از این شبکه‌ها (پردازنده‌ها که هرکدام می‌توانند یک core باشد یا یک کامپیوتر باشد) تشکیل یک می‌دهند و حافظه‌شان را در اختیار یکدیگر قرار می‌دهند (از طریق خطوط ارتباطی data ها و محتویات خودشان را در اختیار دیگران قرار می‌دهند)

- Graph Model ها در دو نوع ۱- بی‌نظم ۲- منظم هستند. در این درس مورد نظر ما گراف‌های منظم هستند.

پارامترهای اندازه‌ی Graph Model:

۱- درجه (d): بیشترین تعداد یال‌هایی که از راس‌ها خارج می‌شود. مقدار درجه باید کم باشد

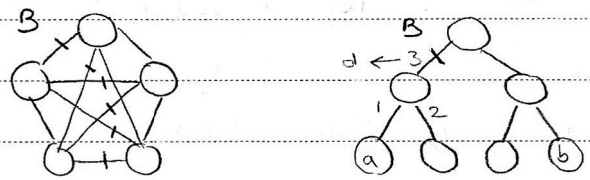
اگر درجه زیاد باشد پیچیدگی ثابت افزاری زیاد می‌شود. (هزینه ثابت بالا)

۲- قطر (D): D^{Max} طولانی‌ترین مسیر ساده بین زوج راس‌ها. مقدار قطر باید کم باشد. طولانی‌ترین مسیر بین دو راس

موقع Broadcast کردن یا Routing اگر قطر کم باشد یا زمان خیلی کمتری اطلاعات منتقل می‌شود.

۳- پهنای دوبخشی (B): حداقل تعداد یال که بین از حذف آنها شبکه به دو بخش تقریبا مساوی تقسیم می‌شود. مقدار پهنای دوبخشی باید بالا باشد. ← دوبخش باید کاملا ازهم جدا شوند!

مثال)



P4PCO

d = 4	d = 3
D = 1	D = 4 (اگر a بخواهد به b برسد ۴ تا یال)
B = 6	B = 1

Subject ق 80
Date

- مقدار پهنای دوبخشی بهتر است که زیاد باشد. اگر مقدارش کم باشد، درصورتی که یکی از خطوط قطع شود کل شبکه از کار می افتد، اما وقتی پهنای دوبخشی زیاد باشد با قطع شدن یکی از خطوط مسیرهای جایگزین برای ارتباط پیدا می شود. همچنین پهنای دوبخشی بالا اجازه می دهد که در زمان Broadcast کردن اطلاعات با بیت بیشتر و به صورت موازی انتقال یابد. هدف این است که پهنای دوبخشی بالایی داشته باشیم و همچنین گراف هم باید گراف کامل باشد.

- پارامترهای اساسی همیشه با هم برخورد میکنند، مثلاً گاهی مجبوریم برای اینکه پهنای دوبخشی را بالا ببریم، درجه راس بالا ببریم، اگر درجه بالا برود پیچیدگی سخت افزاری زیاد می شود. شبکه داریم که الگوریتم به چه توپولوژی نیاز داشته باشد و براساس آن شبکه واسط (Graph Model) را انتخاب می کنیم.

گراف مدل های معروف: (راس های ∘ : سرنخ ها را باید کامل بلد باشیم با جزئیات)

۱- Line (Ring) یک بعدی
۲- Mesh (Torus) دوبعدی
۳- Balanced Binary Tree (درخت باینری)
۴- Complete Graph (گراف کامل)
۵- Hyper Cube (ابر مکعب)
۶- CCC (حالت تغییر یافته Hyper Cube)
۷- Butterfly (پروانه ای)
۸- Hyper Tree (ابر درخت)
۹- Pyramid (هرم)
۱۰- Mesh سه بعدی زیاد در کاربرد نیست و پیاده سازی سخت افزاری آن آسان نمی باشد
۱۱- PRAM (RAM موازی) ← نوعی گراف کامل که الگوریتم های زیادی هم روی آن قابل اجرا است.

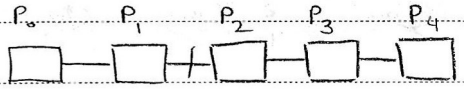
PAPCO

Subject ۸۱

Date

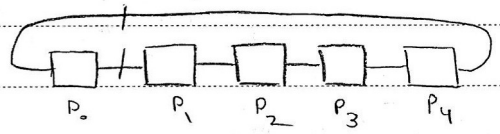
۱) Line

چند کامپیوتر را با هم متصل می‌کنیم و شکل یک خط می‌دهد:



اگر P پردازنده داشته باشیم: $D = P-1$, $B = 1$, $d = 2$

Ring: حالت تغییر یافته Line به صورتی که اولین و آخرین گره به هم وصل می‌شوند.

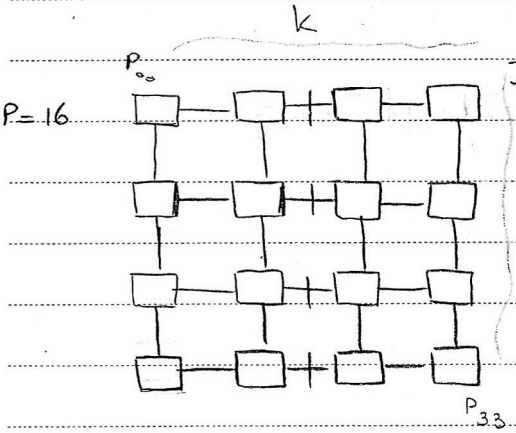


$d = 2$ $B = 2$ $D = \left\lfloor \frac{P}{2} \right\rfloor$

مثلاً از P_0 به P_3 یا P_2 ← 2

* قطر یا بیشترین زمان Routing هم می‌باشد. اگر یک Packet بخواهد از یک گره به گره دیگر ارسال شود زمان Routing چقدر خواهد بود

۲) Mesh



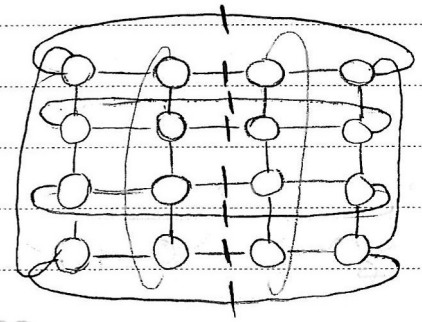
Mesh دوبعدی: سطری یکبار به صورت ستونی (و مسیر باید طی شود، یکبار به صورت

$D = 2(\sqrt{P}-1)$ $P = k^2 \rightarrow k = \sqrt{P}$

در هر سطر یا ستون تعداد یال‌هایی کمتر از تعداد رأس‌هایست k

$d = 4$

$B \simeq \sqrt{P}$ ← اگر زوج باشد دقیقاً $\sqrt{P}$ اگر فرد باشد کم‌تر است.



$D = 2\left\lfloor \frac{\sqrt{P}}{2} \right\rfloor \simeq \sqrt{P}$: Torus

$d = 4$

$B \simeq 2\sqrt{P}$

نسبت به Mesh دوبعدی پهنای دوبخشی دوبرابر و قطر نصف می‌شود.

PqPCO

- ساختار حلقوی خیلی ساختار مهم و کاربردی است (TokenRing با این ساختار کار می‌کند)

(روش) Torus هم تنها با حلقوی کردن روش Mesh توانستیم همسایه‌های بخشی را با لایه‌ی و قطر لازم کم کنیم، بدون آنکه درجه تغییر کند.

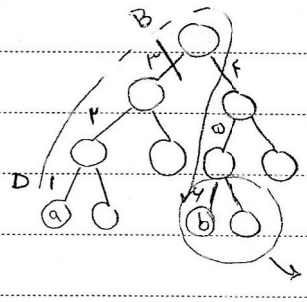
- در کل شبکه‌های line و Ring قابلیت انعطاف بالایی دارند. می‌توانیم با صرف تغییرات یک گره جدید اضافه کنیم

- اگر دو شبکه Mesh بخواهند به هم وصل شوند تعداد یال‌های زیادی باید به هم متصل شوند.

(۳ Binary Tree

معمولا Balanced Binary Tree به کار می‌برند یعنی درخت باینری متوازن

یا تقریبا متوازن = Balanced B.T اگر برگ‌ها در یک سطح با هم اختلاف داشته باشند



- حالتی که درخت کامل نباشد:

$B = 1$

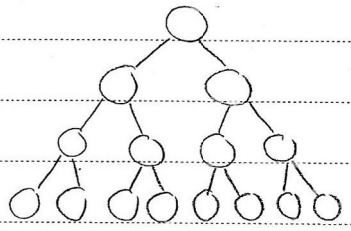
$d = 3$

$D = 2\lfloor \lg P \rfloor (-1)$

در شرایط خاص پیش می‌آید. اگر یک سمت درخت کاملا پر باشد و سمت دیگر نیم پر باشد

$D = 5$ آگر نبود

- حالتی که درخت کامل باشد:



$B = 1$

$d = 3$

$D = 2\lfloor \lg P \rfloor$

اگر $D = 2^k \longrightarrow P = 2^{k+1} - 1$

$D = 2^3 = 8 \rightarrow P = 2^4 - 1 = 15$

P4PCO

83 ✱✱ $D = 2k$

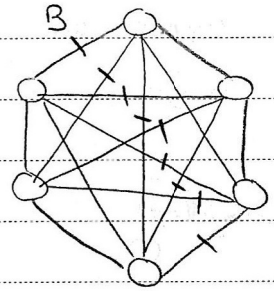
Subject $\text{رشد} = 2^k$

Date $2^{k+1} + \dots + 2^{2k} = ?$

تمرین ۱) قطر Fat-Tree = $2k$ ، تعداد برد رزنده‌ها = ؟

تمرین ۲) اگر قطر یک هرم $2k$ باشد، تعداد گره‌ها = ؟

۴- Complete Graph : (Distributed shared Memory)



$d = P - 1$ ← (ارتباطی) ← درجه ضلع زیادی بین هر گره

بهترین تحت افراز خواهد داشت و با فرزیادی باید مرتب کنیم

تنها حسن این شبکه قطر کم آن است $D = 1$ ←

Routing بسیار سریع خواهد بود

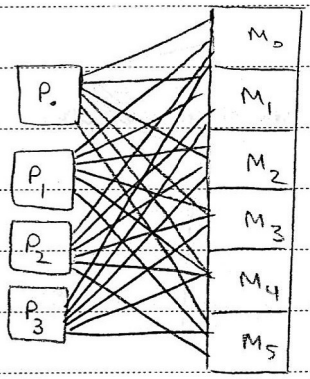
$B = \lfloor P/2 \rfloor \cdot \lceil P/2 \rceil$ ← ضلع بالاست بین خوشه

راست ← 3 ... 3 ← تعداد گره‌ی سمت چپ

۵- PRAM :

یک تعدادی دارنده داریم و یک تعداد واحد حافظه بجای اینکه حافظه‌ها توزیع شده باشند، ضرب دکارتی با واحدهای حافظه به وجود می‌آید

متمرکز هستند اما هر کدام از دارنده‌ها به صورت جزمان به واحدهای حافظه دسترسی دارند



این شبکه پیاده‌سازی نشده چون در عمل اگر بخواهیم هر آن چند پردازنده به یک واحد حافظه وصل کنیم مشکلات زیادی از جمله مشکل هماهنگ‌سازی خواهیم داشت. این ماشین حالت انتزاعی دارد (مثل ماشین تورینگ که در عمل پیاده‌سازی نشده و حالت انتزاعی دارد) اما از لحاظ تئوری PRAM سریعترین ماشین موازی است.

قابل اثبات است که هر الگوریتمی در این ماشین انجام دهیم با یک مقدار کندی بیشتر می‌توان روی ماشین‌های واقعی انجام داد.

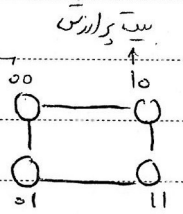
(هر کاری روی PRAM بتوان انجام داد را می‌توان با یک ضریب $\lg n$ روی ماشین‌های واقعی انجام داد) شبه مرتب سازی در این روش را می‌توان با زمان $\Theta(\lg n)$ انجام داد (در جایی که در عمل بهترین الگوریتم تحت افرازی (Batcher) با زمان $\Theta(\lg^2 n)$ انجام دادیم.

PAPCO

۶ـ Hyper Cube :

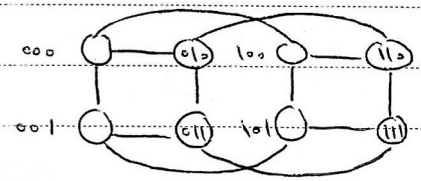
ـ ابرمکعب یک بعدی:



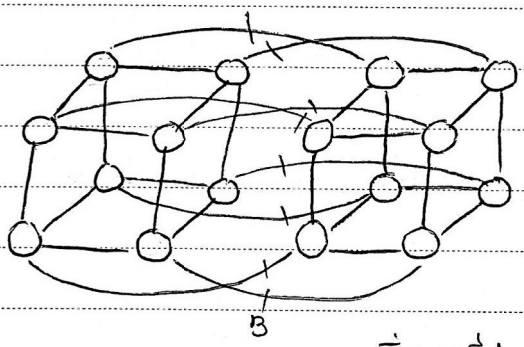


ـ ابرمکعب دوبعدی :

ـ ابرمکعب سه‌بعدی :



ـ ابرمکعب چهاربعدی :



ـ درحالت کلی اگر ابرمکعب k بعدی داشته باشیم :

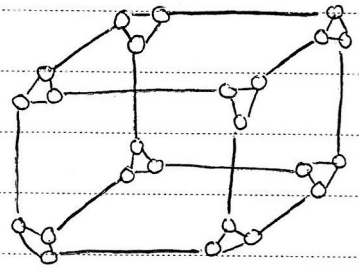
$P = 2^k \rightarrow k = \lg P$

$d = k \rightarrow$ درجه رأس‌ها برابر با تعداد بیت‌هاست $\rightarrow$ Hyper Cube تنها ابرمکعبی است که درجه رأس‌هایش با تعداد بیت‌هایشان برابر است

$B = \frac{P}{2} = 2^{k-1}$ ما به ازای این هیپرکیوب درجه رأس‌ها باید و چهار راستی هم داشته باشد

$D = k$ ← به خاطر این مشکل Hyper Cube ، گراف ccc طراحی شد

۷ـ ccc : شکل ابرمکعب است اما رأس‌ها از روش Ring تبعیت می‌کنند :



Routing پیچیده‌تر اما با جبران هزینه‌ی بخت آزمایی ضعیف کم می‌شود

$P = k \cdot 2^k$

$d = 3$

$D = 2k$ ← قابل اثبات ولی نخواسته!

$B = 2^{k-1}$ ← برای بدست آوردن P بخیر

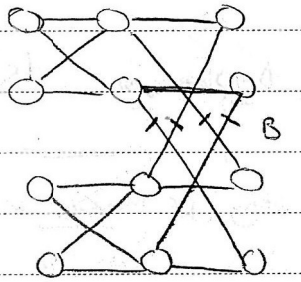
PAPCO

Subject B5

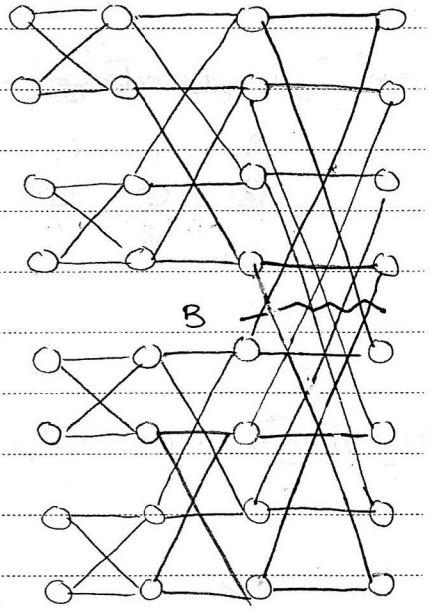
Date

۸- Butterfly : تعداد گره ها: $P = 2^k \cdot (k+1)$ در حالت کلی

$k=2$



$k=3$



$d=4$ (در کل)

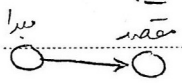
$D = 2^k \ (2 \cdot k)$

$B = 2^k \ (2^k)$

* $k+1$ ستون داریم، در هر ستون 2^k عضو داریم ← $P = 2^k \cdot (k+1)$

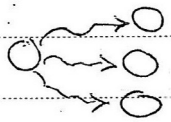
مهمترین الگوریتم های قابل اجرا روی Graph Model ها:

- Routing : می خواهیم Packet را از یک رأس به رأس دیگر بسازیم

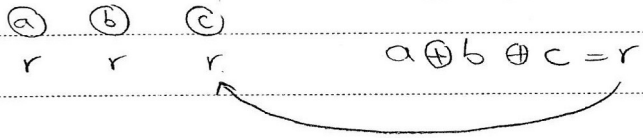


الگوریتم Routing = مقدار تقریبی order

- Broadcast : یک نود بخواهد data خودش را اصطلاحا هم مرز دهد.



- Reduce : فرض کنید روی یک $\oplus$ بخواهیم عمل Reduce را انجام دهیم به این معنی است که همه می خواهند از مقدار حاصلجمع مطلع شوند.



- کاربرد Reduce : رشته های رای گیری، گیری روی نظرات (اتفاق نظر و Max) همه از مقدار آن مطلع می شوند)

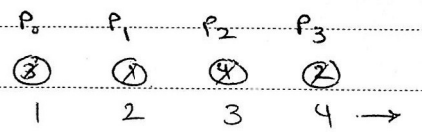
PAPCO

Subject ۸۶
Date

- Prefix Computation : (a) (b) (c)

a a⊕b a⊕b⊕c

- Sorting : یعنی مقادیر را طوری جابجا کنیم که به ترتیب شماره پردازنده مرتب شوند



ابتدا باید پردازنده‌ها عمل Routing و Broadcast انجام دهند تا مرتب شود.

پیاده‌سازی الگوریتم‌ها روی شبکه‌ی Graph Model :

همه الگوریتم‌هایی که روی Graph Model انجام می‌شود براساس ماشین‌های تپنده است.

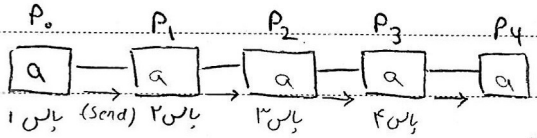
ماشین تپنده :

تعدادی پالس یا Clock اتفاق می‌افتد، در هر کلاک کارهای زیر انجام می‌شود :

۱- دریافت (خواندن) بافرهای ورودی (Recive)

۲- محاسبات محلی ۳- ارسال (پر کردن) بافرهای خروجی (Send)

- Routing شبکه‌ی line :



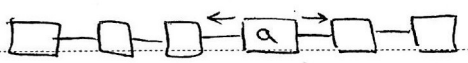
⇒ با ۴ پالس data به مقصد های مورد نظر می‌رسد. به این پالس‌ها زمان اجرای الگوریتم می‌گوییم.

زمان اجرای الگوریتم : اگر گره i بخواهد به گره j داده بفرستد به اندازه $|i-j|$ پالس طول می‌کشد.

با توجه به اینکه j بزرگتر از i هست یا نه جهت حرکت را تعیین می‌کنیم.

روی Packet هایی که ارسال می‌کنیم هم مبدأ هم مقصد، محتوا و هم جهت حرکت مشخص شده

مثل همان کار Broadcast : Routing هست. حال اگر گره میانی بخواهد پیام بفرستد :



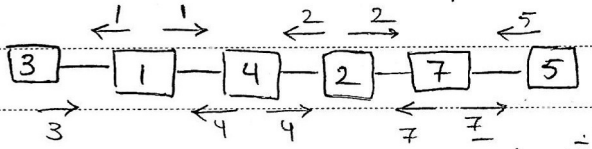
یک packet به سمت چپ باید بفرستد یک Packet به سمت راست، هرکس Packet با عنوان تحت راست دریافت کرد به سمت راست بفرستد و برعکس

زمان Routing و Broadcast در بدترین حالت : به اندازه قطر شبکه $D = P - 1$

در کل مثل تحلیل‌هایی که قبلاً کردیم جواب زمان الگوریتم هم اینجا همان است. $P/2$: اگر Ring باشد

Reduce_ : فرض کنید عمل Reduce روی max انجام می‌شود (Reduce_Max)

یعنی همه گره‌ها می‌خواهند مقدار max را بدانند :



۱) هرکسی data خودش را برای همسایه‌اش می‌فرستد

۲) همسایه‌ها درصورت بهتر شدن جواب خود را به همه‌شان می‌گویند. اگر شود مقدارش تغییر می‌کند.

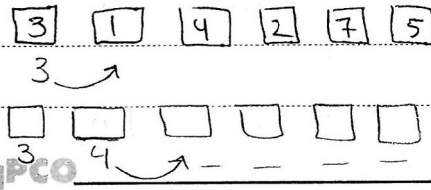
3 4 4 4 4 7 7 7 7 7

1) 3 4 4 7 7 7 — 3) 4 7 7 7 7 7

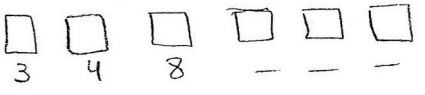
2) 4 4 7 7 7 7 — 4) 7 7 7 7 7 7

حداکثر مراحل = (همه گره‌ها) $D = P - 1$

حال فرض کنید می‌خواهیم حاصلجمع کل را به دست آورده و حاصلجمع کل را در اختیار همه گره‌ها قرار



دهیم : گره اول مقدار فعلی خودش را در اختیار همسایه قرار می‌دهد و حاصلجمع آن حساب می‌شود در گره ذخیره می‌شود، در مرحله بعد گره دوم مقدارش را در اختیار همسایه بعدی قرار می‌دهد و ...

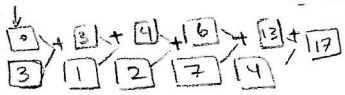


حداکثر پاس‌ها = $P - 1$

PAPCO

Subject ** 88
Date

Diminished Prefix Sum : حاصلجمع باکس بیت آخر پدرست میاد



برای اینکه همه مقدار حاصلجمع کل را بدانند ابتدا باید یک Prefix Computation انجام شود

$P-1$ $P-1$

و بعد یک Broadcast انجام شود. زمان ۲ برابر می‌شود اما از لحاظ مرتبه همان $\Theta(P)$ هست

اگر شبکه Ring باشد Prefix Comp. همان $(P-1)$ و Broadcast = $P/2$ زمان می‌برد

Prefix computation بلاک :

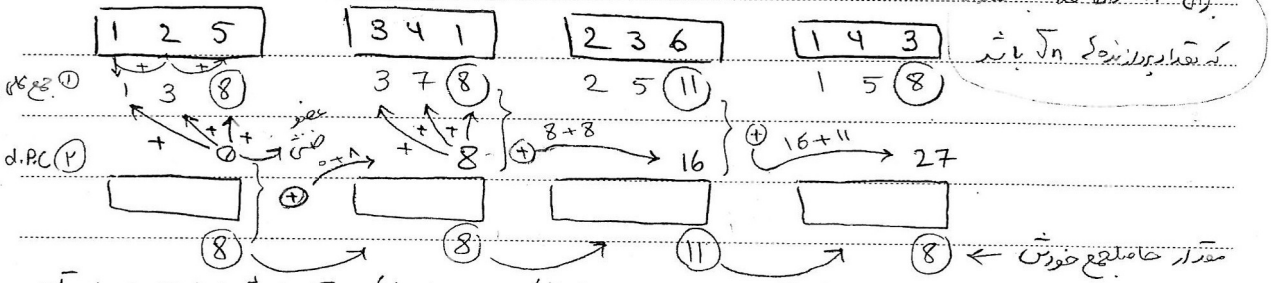
هر کدام از اعضا یک بلاک data دارند اما تعداد پردازنده‌ها کم است. می‌خواهیم

Prefix computation انجام دهیم، بلاک‌ها طولانی‌اند اما تعداد پردازنده‌ها کم است و هر پردازنده هم زیادی دارد

- در این صورت به صورت محلی Prefix-Comp. را انجام می‌دهیم (هر پردازنده جداگانه Prefix_Comp را انجام می‌دهد بلاک‌ها به صورت موازی این کار را انجام می‌دهند)

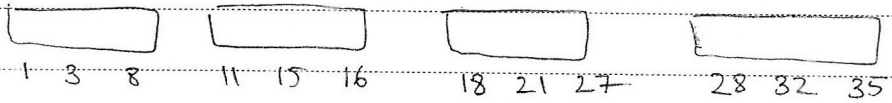
سپس روی مقادیر آخری که داریم یک diminished Prefix Computation انجام می‌دهیم (باکس آخر بلوک‌ها حساب شود یعنی مثلاً در بلاک اول مقدار خودمان را صفر می‌گذاریم و مقدار حاصلجمع را برای همه می‌فرستیم حساب مقدار خودش را بعلاوه حاصلجمع می‌فرستد برای مرحله بعد) در diminished همیشه مقدار اولی صفر (عضو خنثی) است

برای n داده مناسب هست که تعداد پردازنده‌ها $\sqrt{n}$ باشد



تعداد پاس = زمان = $D = P-1$

در مرحله آخر مقادیر قرمز رنگ را با تک تک مجموع هر بلاک که در ابتدا آوردیم جمع می‌کنیم :

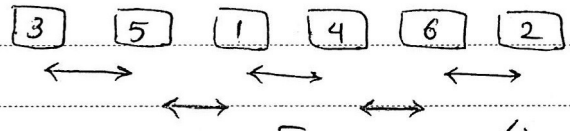


P4PCO

تعداد پاس‌ها (مراحل) $P-1$ و محاسبات محلی که در هر RAM به صورت جداگانه هزینه چندانی ندارد. پس در کل زمان همان $\Theta(n)$ هست.

- Sorting : به دو شکل هست ، یا data از بیرون می‌آید یا data را از قبل داریم.

در اینجا فرض بر این است که data را داریم ولی نامرتب است.

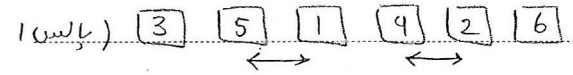


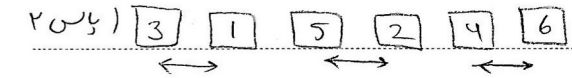
پاس ۱
پاس ۲

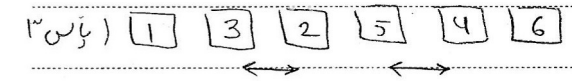
ایده Sorting در line قبل ایده شکل مرتب سازی آخری است

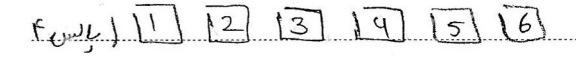
در پاس اول پردازنده های زوج با همسایه راست مقایسه می‌شود ، در پاس بعدی پردازنده‌های فرد با همسایه راست مقایسه می‌شود . این کار آنقدر تکرار می‌شود تا data مرتب شود.

زمان سعی در بدترین حالت = P پاس









اگر یک پردازنده باشد :

$$w_1(P) = T_1(P) = \Theta(P \lg P)$$

$$\rightarrow \begin{cases} T(P) = P \rightarrow P \text{ پاس} \\ w(P) = \Theta(P^2) \end{cases}$$

(پردازنده)

در هر پاس یک مقایسه بین دو عضو رخ می‌دهد (Send/Rec) ← P پاس داریم و در هر پاس P عمل (Send/Rec)

$$\text{Speed up} = \frac{T_1(P)}{T_P(P)} = \frac{\Theta(P \lg P)}{\Theta(P)} = \Theta(\lg P)$$

→ نه تنها خطی نیست بلکه اصلا خوب نیست چون متناسب با lg هست ، یعنی هر چی جلو برید ضرب دیگر پاس.

$$\text{Redundancy} = \frac{w_p(P)}{w_1} = \frac{\Theta(P^2)}{\Theta(P \lg P)} = \Theta\left(\frac{P}{\lg P}\right)$$

→ خیلی بالاست

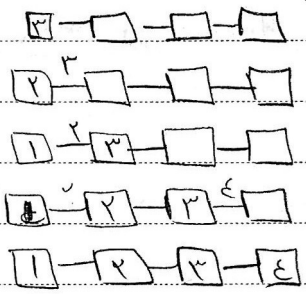
* علت اینکه Sorting در شکل line بهینه نیست (هم speedup کم و هم دوباره‌کاری زیاد) از خود شکل line هست. شکل line رهنمای دوبخشی کمی دارد (یعنی با قطع شدن یک خط کل شکل از کار می‌افتد)

PqPCO

Subject 90
Date

الگوریتم مرتب سازی در شبکه line :

حالت ۱) data از بیرون بیاید



داده از بیرون بصورت سریال وارد می‌شود.

۱) عدد اول وارد می‌شود و عدد دوم

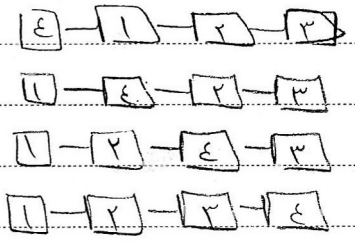
ابتدا با (۱) مقایسه می‌شود و در صورت

نزوم جابجا می‌شود و به سمت جلو حرکت می‌کند

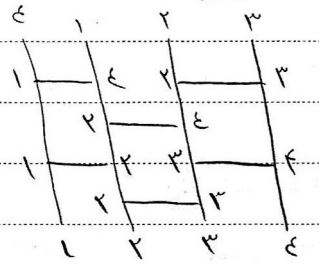
۲) این کار تا آخر انجام می‌شود.

تعداد پاس : $2p-1$

حالت ۲) پردازنده‌ها از قبل عدد داشته باشند ← از شبکه مرتب ساز آخری استفاده می‌شود



شبکه آخری را ۹۰° می‌چرخانیم تا مرتب سازی صورت گیرد.



نتیجه :

* همدتا در زمان $p-1$ پاس اما در بدترین حالت p پاس

تحلیل :

$$T(p) = p$$

$$C(p) = \frac{p(p-1)}{2} \Rightarrow \Theta\left(\frac{p^2}{2}\right)$$

$$S_p = \frac{T_1}{T_\infty} = \frac{p \lg p}{p} = \lg p \rightarrow$$

خوب نیست و این خاطر مشکل شبکه line است که همیاری به خوبی ندارد.

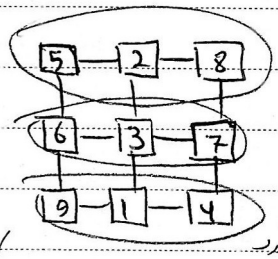
$$R_p = \frac{W_p}{W_1} = \frac{p^2}{2p \lg p} = \frac{p}{2 \lg p}$$

Subject 91
Date

پیاده سازی الگوریتم های متفاوت در شبکه Mesh :

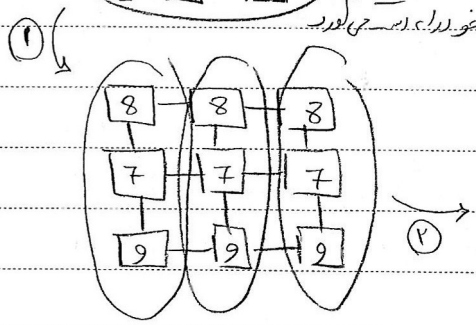
- محاسبه Max در Mesh دوبعدی :

(مثال)



۱) مرحله اول هر سطر مقدار max خودش را پیدا می کند.

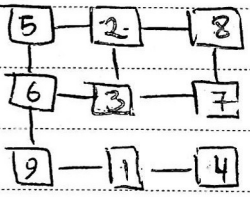
۲) مرحله دوم هر ستون مقدار max خود را پیدا می کند



زمان مصرفی = تعداد شبکه = $2\sqrt{P}-1$

- محاسبه Prefix_Sum در Mesh دوبعدی :

(مثال)



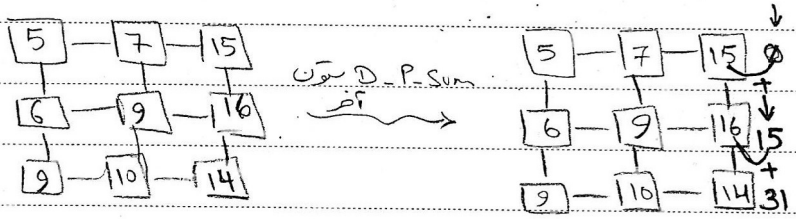
۱) مرحله اول محاسبه Prefix_Sum در هر سطر

۲) محاسبه diminished Prefix_Sum فقط در ستون آخر

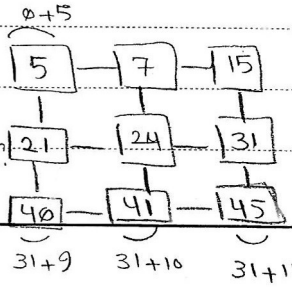
۳) جمع مقادیر با مقادیر بدست آمده در مرحله ۲

Pre_Sum هر سطر

Diminished Prefix Sum



جمع مقادیر مرحله ۲ با همه مقادیر



$T = 3\sqrt{P}-1$

۳ برابر $\sqrt{P}-1$ هربار

P4PCO

Subject 92
Date

- الگوریتم Sorting در شبکه Mesh (مرتب سازی مارپیچی)

4	16	10	7
2	8	15	12
11	14	13	5
6	1	9	3

۱) یکی در میان سطرها را به صورت صعودی و نزولی مرتب می‌کنیم
(اول صعودی سطر بعد نزولی و ...)
۲) به صورت صعودی هر ستون را مرتب می‌کنیم.
۳) دو مرحله قبل را تکرار می‌کنیم
٤) به صورت سطری و صعودی مرتب می‌کنیم.

①

4	7	10	16
15	12	8	2
5	11	13	14
9	6	3	1

مرحله ۱

②

4	6	3	1
5	7	8	2
9	11	10	14
15	12	13	16

مرحله ②

③ تکرار ۲ مرحله قبل

1	3	4	6
8	7	5	2
9	10	11	14
16	15	13	12

مرحله ۱

1	3	4	2
8	7	5	6
9	10	11	12
16	15	13	14

مرحله ۲

④

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

مرحله ۴

تعداد مراحل : $\lg\sqrt{P}$
مقایسه‌ها در هر مرحله $2\sqrt{P}$
زمان کل = $(\lg P+1)\sqrt{P}$

- پیاده‌سازی الگوریتم‌های مختلف در Binary Tree :

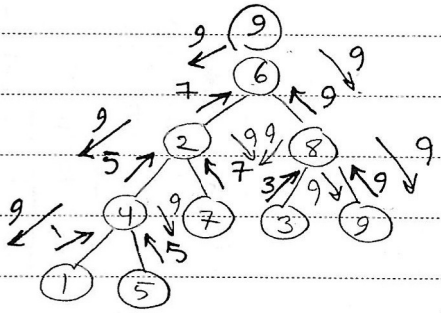
- دو مرحله دارد یکی بالا رفتن (فهماندن) یکی پایین آمدن (Broadcast)
$\lg P$ $\lg P$

PqPCO

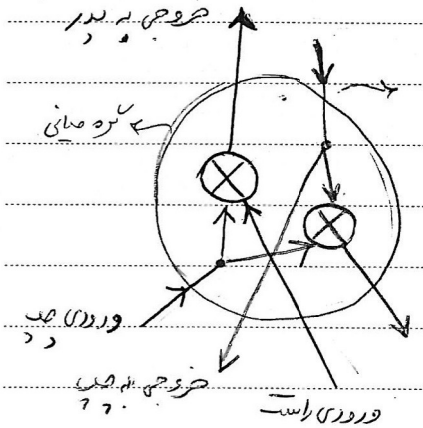
Subject 93
Date

- برای Prefix_Sum هر گره مقدار خودش را در اختیار پدرش قرار می‌دهد و پدر هم مقدار خودش را با مقادیر فرزندانش جمع می‌زند و برای پدرش می‌فرستد. در نهایت ریشه درخت مقدار های جمع را دارد و برای فرزندانش می‌فرستد.

- برای Max باز هم هر گره مقدار خودش را در اختیار پدرش قرار می‌دهد پدر مقادیر فرزندان با مقدار خودش مقایسه می‌کند و max را برای پدرش می‌فرستد. در نهایت ریشه درخت مقدار max ها را در اختیار فرزندانش قرار می‌دهد.



- الگوریتم Prefix computation در Binary Tree :

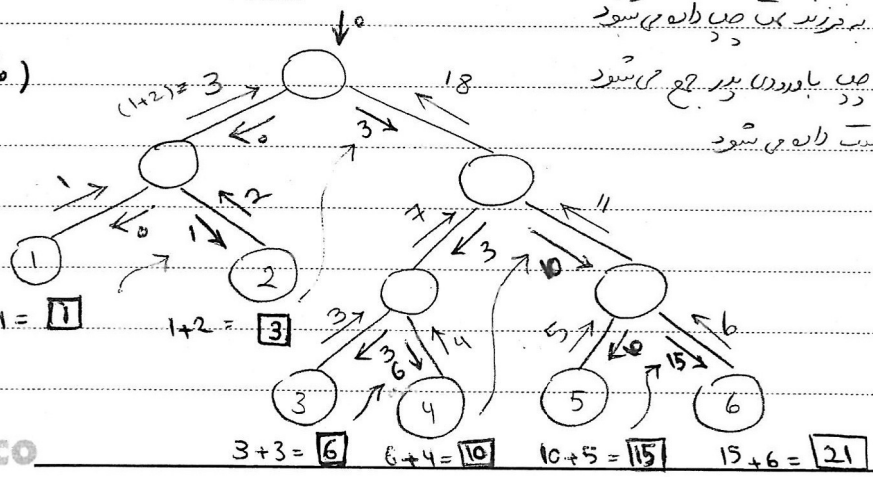


* در این الگوریتم فرض می‌کنیم فقط برگ‌ها مقدار دارند.

۱) برگ‌ها مقادیر خود را به پدر ارسال می‌کنند و پدر با هم جمع می‌زند و برای پدرش می‌فرستد

۲) برای برگرداندن جواب از ریشه به سمت فرزندان ارسال می‌شود. ورودی که سمت پدر است عیناً و بدون هیچ تغییری به فرزند سمت چپ داده می‌شود و مقدار گره سمت چپ با ورودی پدر جمع می‌شود و به گره سمت راست داده می‌شود

(مثال)

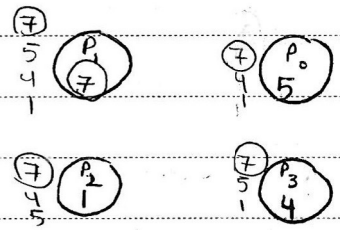


P4PCO

Subject 94
Date

- پیاده‌سازی الگوریتم‌های متفاوت در Complete Graph

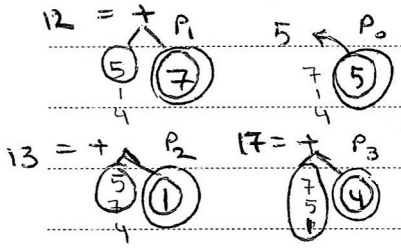
- محاسبه Max :



۱) هر مقداری خودش را برای مقدار می‌فرستد
۲) هر گره ورودی خودش را با ورودی خودش مقایسه و Max را پیدا می‌کند

زمان معرفی = $\Theta(P-1) + \Theta(1) + \Theta(P-1) = \boxed{\Theta(P)}$

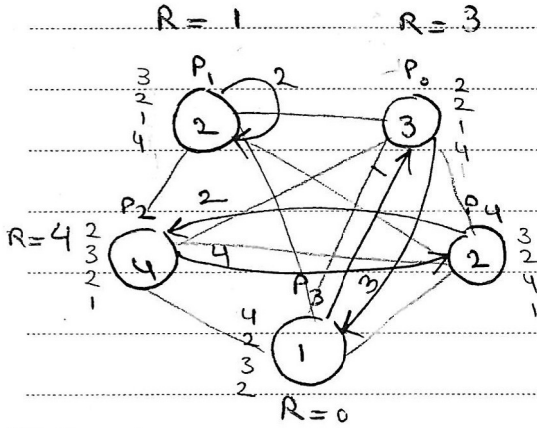
- محاسبه Prefix_Computation :



۱) هر مقداری خودش را برای مقدار می‌فرستد
۲) هر پردازنده مقدارش را با پردازنده قبل از خودش جمع می‌کند.

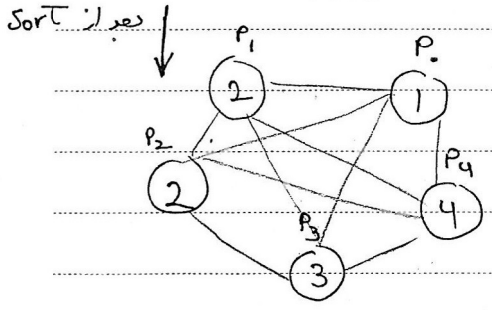
$\boxed{\text{زمان معرفی} = \Theta(P)}$

- Sorting :



۱) تمام گره‌ها مقادیر خودشان را برای مقدار می‌فرستند
۲) برای هر گره Rank را حساب می‌کنیم
۳) هر پردازنده مقدار خودش به پردازنده‌ای با شماره Rank مقدارش می‌فرستد.

Rank : تعداد پردازنده‌هایی که مقادیر آن‌ها از مقدار گره فعلی کوچکتر باشد و در صورت تساوی شماره پردازنده آن گره کوچکتر باشد.



زمان معرفی : $\Theta(P-1) + \Theta(1) + \Theta(P-1) + \Theta(1)$

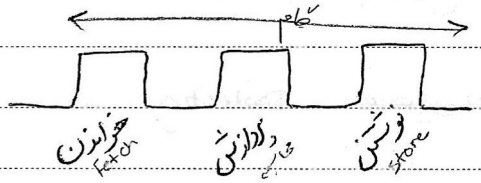
$= \boxed{\Theta(P)}$

P4PCO

Subject 95
Date

PRAM (Parallel RAM)

همان طور که گفتیم به صورت تخت اقراری مقابل پیاده سازی نیست و جالب انتزاعی دارد. فرآیند محاسبات در PRAM به صورت سینکرونیک (تپنده) است. گام ها یا پالس های بزرگ از ساعت clock تشکیل شده: ۱- خواندن (fetch کردن / load) : پردازنده ها همزمان با هم می توانند شروع کنند به خواندن data ۲- پردازش محلی: data هایی که دریافت کردند روان clock یک محاسباتی روی آن انجام می شود (calculate) ۳- نوشتن (save کردن) : همهٔ پردازنده ها باهم می توانند data را بنویسند.



- با توجه به قابلیت های خواندن و نوشتن که همزمان، می توانیم مدل های مختلفی را برای PRAM تصور کنیم. مدل های PRAM:

- به صورت پیش فرض PRAM دارای خصوصیت CREW است. یعنی خواندن به صورت عمومی و مشترک و نوشتن به صورت انحصاری باشد

- مدل کامل تر CRCW می باشد، یعنی نوشتن هم به صورت همزمان صورت بگیرد ولی تبعات آن زیاد هست، اینکه چه چیزی نوشته شود و چه اتفاقی برای آن خانه از حافظه می افتد. CRCW را به انواع مختلفی تقسیم می کند

- EREW حالت ساده‌تری از CREW است که در اینجا اجازهٔ خواندن هم‌زمان هم داده نمی‌شود

در واقعیت EREW خیلی ضعیف‌تر از CRCW نیست، کاری که در CRCW انجام دهیم

از نظر زمانی با یک ضرب در $\lg$ می‌توانیم در EREW ماشین ساده و ضعیف‌تر هم انجام دهیم.

در CRCW چه راه‌حل‌هایی برای نوشتن هم‌زمان وجود دارد؟

- Undefined : موقع نوشتن هر چیز غلطی نوشته شود (حالت پیش‌فرض که اگر ناآگاهانه نوشتن هم‌زمان رخ داد هر چیزی اتفاق بیفتد.) (CRCW-U)
- Detecting : تشخیص دهد که خطا نوشته شده و یک مقدار NULL یا پیغام خطا ثبت شود (CRCW-D)
- Common : اگر مقادیر یکسان بودند همان مقدار نوشته شود، اگر نه ERROR نوشته شود (CRCW-C)
- Random : به صورت Random یکی از دو مقدار نوشته شود (CRCW-R)
- Priority : (اولویت دار) هر کدام شماره‌ی داشته باشد نوشتن را انجام دهد (CRCW-P) قوی‌ترین حالت در CRCW - به صورت افتراقی هست نه واقعی
- Reduction : یکی از این عملیات روی را مقدار صورت گیرد: OR, and, max, min جمع یا (هر semigroup) و در نهایت ترکیب آن نوشته شود

+ از نظر قدرت EREW از همه ضعیف‌تر هست، بعد CREW بعد CRCW-D

مرتب سازی در زمان $\lg^2 n$

CRCW-C، CRCW-R، CRCW-P

مرتب سازی در زمان $\lg n$ انجام می‌شود، چیزی که ضعیف‌تر از آن هست

PqPCO

Subject 97
Date

الگوریتم هایی که می‌توانیم روی CREW و EREW بنویسیم:

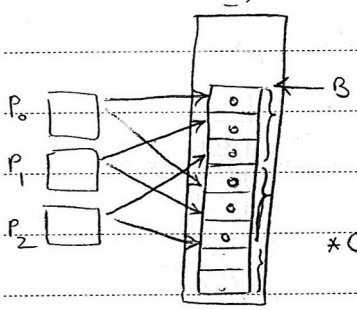
- الگوریتم Copy کردن (مقداردهی آرایه): کپی کردن مقدار صفر

وقتی می‌نویسیم B[x] درایه‌ی صفر دوم آرایه ← آدرس آرایه ← فاصله M[B+x] انجام می‌شود یک یک

آدرس جدید تولید می‌کند و دیگر نیاز مستقیم دارد.

مثال) می‌خواهیم آرایه B (که در وسط حافظه است و n خانه دارد) را پر کنیم:

آرایه



* زمان $\lceil \frac{n}{p} \rceil$

8 تا خانه با 3 تا پردازنده داریم

ترتیب نوشتن ↓

	پاس 1	پاس 2	پاس 3
P_0	0	3	6
P_1	1	4	7
P_2	2	5	8

در مرحله اول هرکدام از پردازنده‌ها یکی از خانه‌ها را پر کنند و در مرحله دوم به اندازه تعداد پردازنده‌ها (اینجا $P=3$) خانه پر شود جلو.

* زمان اجرا به تعداد بلاک‌هایی که می‌توانیم همزمان بنویسیم بستگی دارد

الگوریتم:

Initialize an n_vector (base address = B) to all 0s:

for_Proc i=0 to p-1 do → * هیچ زمانی نمی‌برد for_Proc چون همزمان باهم کار می‌کنند

$\Theta(n/p)$ ← for j= 0 to $\lceil n/p \rceil - 1$ do

if $jp + i < n$ then $M[B + jp + i] = 0$ → اگر اندیس بالاتر از n نشد استاب اکی انجام شود.

end for

endfor

تحلیل: کل زمان = n/p → اگر n بود: Speed up = $\frac{n}{n/p} = p$

Speed up خطی است و بهینه است.

* for_Proc یعنی همه پردازنده‌ها باهم شروع به کار کنند

- الگوریتم جمع دو آرایه :

از همان الگوریتم مبدی استفاده می‌کنیم فقط بجای انتساب صفر (خط if)

خط زیر را اضافه کنیم :

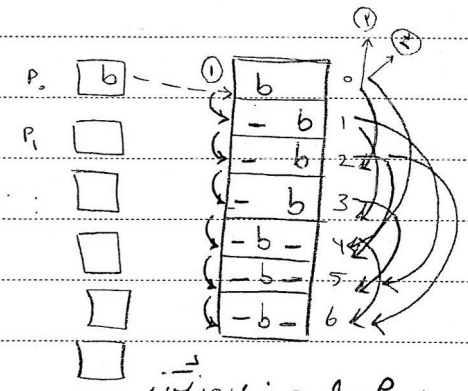
$[c + jp + i] := [A + jp + i] + [B + jp + i]$

- الگوریتم Broadcast :

در CREW اول یک نوشتن داریم و بعد یک خواندن که می‌شود ۲ کلاک بزرگ .

مشکل زمانی است که EREW باشد (یعنی خواندن و نوشتن انحصاری باشد) . در این حالت

به صورت زیر عمل می‌کنیم :



۱) همه‌ی داننده‌ها هرچه دارند در یک خانه حلوا بنویسند (هرکس یک گام حلوا بنویسد)

۲) هرکس دو گام حلوا بنویسد

۳) هرکس چهار گام حلوا بنویسد .

زمان نوشتن = $\lg P$

این الگوریتم بیازشبیه kogg stone است .

الگوریتم :

Making P copies of B[0] by recursive doubling

for $k = 0$ to $\lceil \lg_2 P \rceil - 1$ do $\longrightarrow \Theta(\lg P)$

$\Theta(1)$ {
for_Proc $j = 0$ to $P - 1$ do
Copy $B[j]$ into $B[j + 2^k]$ $\longrightarrow$ (یک گام هست (بزرگ)
}

end for

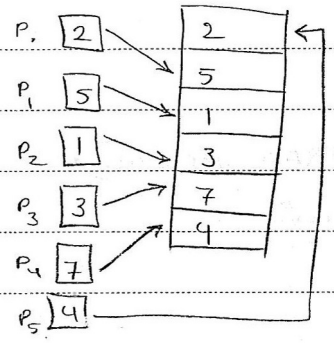
end for

کلا زمان $\Theta(\lg P)$

اگر $2^k - 1$ بود مقدار work کمتر می‌شد چون خانه‌های null کپی نمی‌شود. Redundancy

Broadcast All_To_All :

الگوریتم‌هایی که خواهند مقادیرشان را برای یکدیگر بفرستند. کاربردآن در Prefix comp. هست. (مثلاً در Complete Graph این نوع Broadcast کردن کاربرد داشت) برای مرتب سازی هم کاربرد دارد. برای این کار از تمام مقادیر پردازنده‌ها یک کپی می‌گیریم و در حافظه RAM ذخیره می‌کنیم، حال بقیه می‌توانند از آرایه استفاده کنند، در مرحله بعد P_0 باید مقادیر سایر پردازنده‌ها را بخواند، همچنین P_1 و...



$P_0 \xrightarrow{\text{بخواند}} P_1\ P_2\ P_3\ P_4\ P_5$

$P_1 \longrightarrow P_2\ P_3\ P_4\ P_5\ P_0$

$P_2 \longrightarrow P_3\ P_4\ P_5\ P_0\ P_1$

} کل عملیات هزوان است

الگوریتم:

EREW PRAM algorithm for all-to-all broadcasting

$\Theta(1)$ for Proc $j=0$ to $P-1$ do → $P-1$ خانه داریم
write own data value into $B[j]$

$\Theta(P-1)$
for Proc $j=0$ to $P-1$ do
for $k=1$ to $P-1$ do
Read the data value in $B[(j+k) \bmod P]$
end for
end for

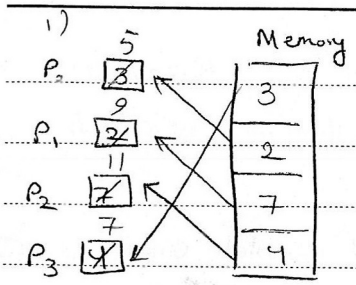
بنابر مقادیر از تعداد پردازنده‌ها بیشتر شود.

مرتبه زمانی = $\Theta(P)$

Subject 100

Date

۱)



مثال All_To_All :

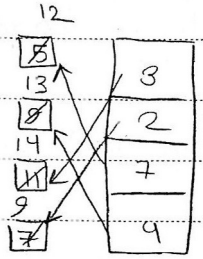
۱) هر مقدار خود را در RAM می‌نویسد

۲) هر از یک خانه حافظه بخوانند و فقط چون آخرین خانه $i+1$ نداریم از $(i+1) \bmod P$ استفاده می‌کنیم

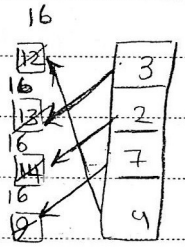
۳) هر کس از ۲ خانه حافظه بخواند

۴) هر کس از ۳ خانه حافظه بخواند

۲)



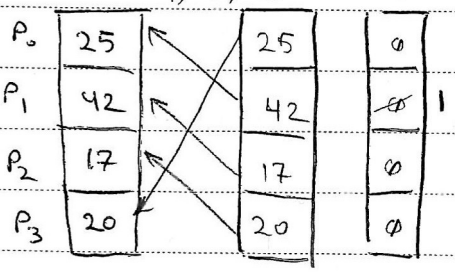
۳)



مربوط بازی در PRAM (EREW)

شبه الگوریتم Complete Graph می‌باشد.

۱) ۲)



۱) در یک پایین می‌توان S و R را پر کرد (Rank, mema)

← S با مقادیر محلی و R با ۰ پر می‌شود

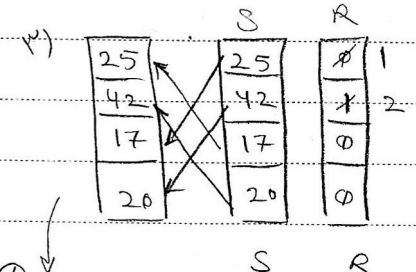
۲) هر پردازنده از یک خانه حافظه بخواند، اگر بزرگتر از مقدار محلی‌اش بود حتی اگر نه کوچکتر بود مقدار R یک واحد از این پایین باید

۳) از ۲ خانه حافظه بخواند و...

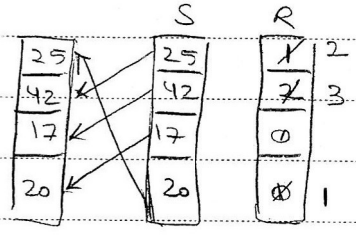
۴) از ۳ خانه حافظه بخواند و...

۵) در نهایت طبق R هر عدد بزرگ در خانه پردازنده مربوطه

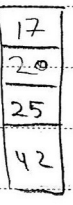
۳)



④



⑤ نوشتن در پردازنده متناظر R



زمان بدترین: $\Theta(P)$

هزینه:

P4PCO