به نام پروردگار توانا

پردازش موازی

ترجمه فصل دوم

از کتاب

INTRODUCTION TO PARALLEL PROCESSING

ALGORITHMS AND ARCHITECTURES

BY:

BEHROOZ PARHAMI

استاد: دکتر آرش قربان نیا

ترجمه و ویرایش : سمانه سادات علوی فر

پائیز 1390

طعم الگوريتم‌هاى موازي

در اين بخش 5 عمل اصلي ساده كه به عنوان پايه‌هاى سازنده معماري موازي هستند و در بخش 2.1 تعريف شد را بررسي مى‌كنيم و الگوريتم‌هاى متناظر با 4 معماري موازي را نيز بررسي خواهيم كرد:آرايه خطي، درخت باينري، مش دو بعدي و اشتراك گذاري ساده متغيرهاي كامپيوتري. (بخش 2.2 را ببينيد) اين تمرين‌ها ما را به محاسبات موازي معرفي مي كندو همچنين اثر متقابل بين الگوريتم و معماري و پيچيدگي محاسبات موازي (تحليل و حدود) را بررسي مى‌كند. همچنين پايه‌هاى اصلي محاسباتي كه براي ما مهم است ودر طول كتاب استفاده شده‌اند را خواهيم ديد.

بعضي از معمارى‌ها را به طور عميق‌تر در فصل‌هاى بعدي خواهيم خواند.

موضوعات فصول عبارتند از :

2.1 برخي از محاسبات ساده

2.2 برخي از معمارى‌هاى ساده

2.3 الگوريتم‌هاى آرايه خطي

2.4 الگوريتم‌هاى درخت باينري

2.5 الگوريتم‌هاى مش دو بعدي

2.6 الگوريتم‌هاى متغيرهاي اشتراكي

2.1 چند محاسبه ساده

در این بخش 5 محاسبه ابتدايي و زيربنايي را تعريف می‌کنيم:

1- محاسبات زنجيره‌ای (کاهش، گنجايش ورودي)
2- محاسبات پيشوندي موازي (محاسبات موازي)
3- مسيريابي بسته‌ای
4- همه پخشي و نسخه عمومی‌تر آن چند پخشي (انتشاري)
5- مرتب کردن رکوردها بر مبناي کليدشان (صعودي يا نزولي)

Semi group computing محاسبات زنجيره‌ای (کاهش، گنجايش ورودي):

* به عنوان عملگر باينري شرکت پذير:

مثال :

$$(x * y) * z = x * (y * z)$$

به ازی هر x , y, z ϵ S

A زنجيره‌ای ساده از جفت (S,*) است که S مجموعه‌ای از عناصر است که * روي آن تعريف می‌شود.

محاسبات زنجيره‌ای (که همچنين کاهش يا گنجايش ورودي نيز خوانده می‌شود.) به صورت زير تعريف می‌شود:

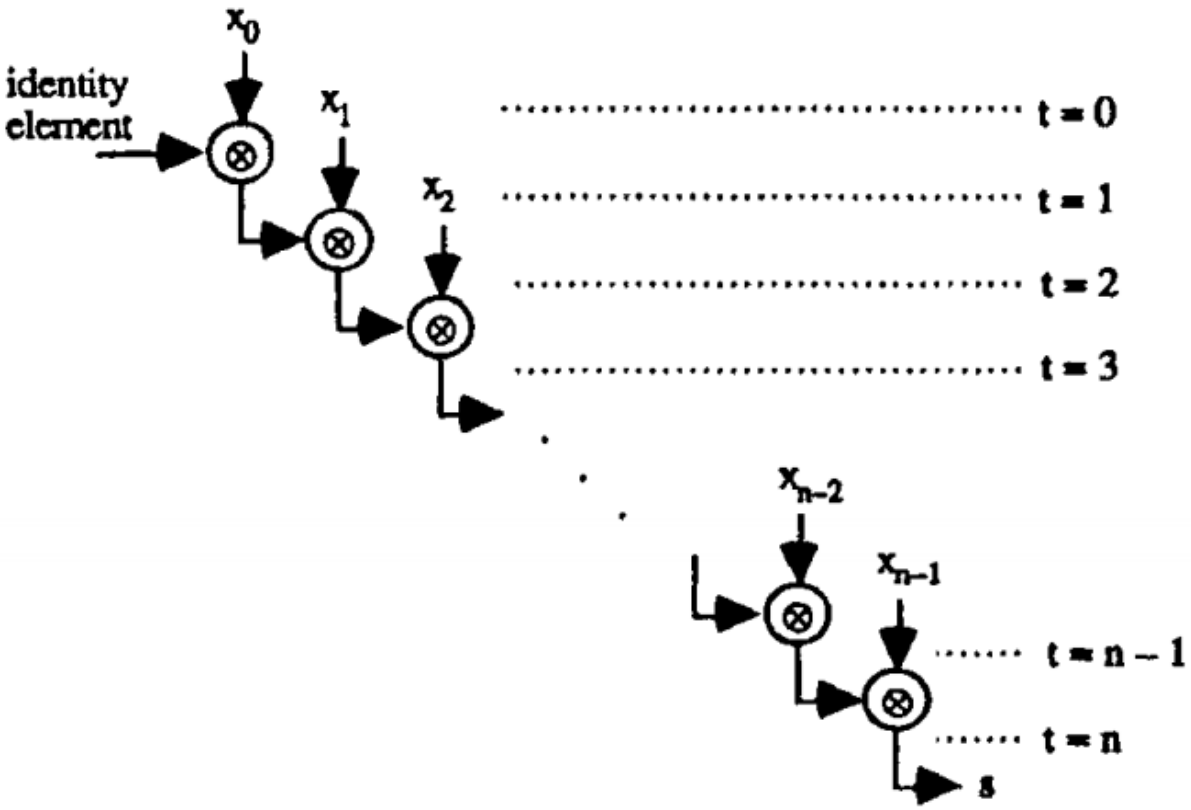
ليست n مقداري: $x_0 , x_1, ..., x_{n-1}$ را می‌گيرد و محاسبه زير را انجام می‌دهد:

$$x_0 * x_1 * ....* x_{n-1}$$

اين عملگر می‌تواند يکي از عمليات +,-,×,,∨۸,∩,∪ باشد. اپراتور * ممکن است جا به جايي پذير باشد يا خير. لذا ممکن است x*y = y*x درست باشد يا نه. (همه مثال‌های

بالا در اين رابطه صدق می‌کنند ولي به عنوان مثال محاسبه کري صدق نمی‌کند.)

اين نکته آخر بسيار مهم است؛ تا زماني که الگوريتم موازي بتواند مقادير قابل توجهي از عبارات را با استفاده از بخش بندي طرح‌ها محاسبه کند، اين مقادير بايد در نهايت به صورت چپ به راست ترکيب شوند. شکل 2.1 محاسبات زنجيره‌ای را در تك پردازنده نشان می‌دهد.



شکل 2. 1 محاسبات زنجيره‌ای در تك پردازنده

محاسبات پيشوندي موازي:

با فرض مشابه پاراگراف قبلي، محاسبات پيشوندي موازي به عنوان ارزيابي همزمان همه پيشوندهاي عبارت $x_0 * x_1 * \ldots * x_{n-1}$ است که $(x_0, x_0*x_1, x_0*x_1*x_2, \ldots, x_0*x_1*\ldots*x_n)$.

توجه کنید که i امین پیشوند، عبارت برابر است با $S_i = x_0 * x_1 * x_2 * \ldots * x_i$.

توضيح راجع به شدت جابه جايي پذيري، يا فقدان آن از عملگر بايناري * در اين جا به خوبي كاربرد دارد. گراف نمايش دهنده محاسبات موازي در تك پردازنده شبيه شكل 2.1 است اما مقادير مياني و همچنين خروجي را نيز دارد.

مسيريابي بسته‌ای:

بسته اطلاعات در پردازنده i ام مستقر است و باید به پردازنده j ام فرستاده شود. مشكل، مسيريابي كردن بسته طي پردازنده‌های مياني است كه لازم است مقصد را تا حد امكان به سرعت پيدا كند. در واقع مشكل وقتي است كه چندين رقابت پيش می‌آيد. وقتي كه چندين بسته هر كدام با مقصد خودشان در پردازنده‌های جداگانه مستقر هستند، در اين حالت مسيرهاي بسته ممكن است با بسته ديگري تداخل پيدا كند كه به پردازنده‌های مياني بروند. (در مسيريابي تداخل پيش بيايد).

وقتي هر پردازنده حداكثر يك بسته براي فرستادن و دريافت كردن داشته باشد، مسئله مسيريابي بسته‌ها ارتباط يك به يك يا مسيريابي 1-1 ناميده می‌شود.

همه پخشي : انتشاري

گرفتن مقدار مشخص a از پردازنده معين i و انتشار آن به همه پردازنده‌ها به سرعت ممكن. كه در پايان همه پردازنده‌ها به آن دست می‌يابند يا مقدار را می‌شناسند. اين روش گاهي به ارتباط يك به چند منسوب است.

عمومی‌ترين مورد اين عمليات كه ارتباط يك به چند است، به عنوان چند پخشي يا Multicasting شناخته می‌شود.

از نقطه نظر برنامه نويسي داريم:

$X_{j:} = a$ for $1 \le j \le p$ (Broadcasting)

For jєG (Multicasting)

وقتي G يك گروه چند پخشي باشد و X_j يك متغير محلي در پردازنده j.

مرتب سازي[1] :

به جاي مرتب سازي يك مجموعه از ركوردها، هر كدام با كليد و عناصر داده، براي ساده سازي ما روي مرتب كردن مجموعه‌ای از كليدها تمركز می‌كنيم. مساله مرتب سازي ما به اين صورت تعريف می‌شود:

گرفتن ليستي از nكليد $x_{i0}, x_{i1}, ..., x_{in-1}$ به طوري كه $x_{i0} \le x_{i1} \le .... \le x_{in-1}$.

و لذا تنها مرتب سازي كليدها برحسب رتبه غيرنزولي شان را رسيدگي می‌كنيم.

هر الگوريتمي براي مرتب كردن مقادير، بر حسب رتبه غيرنزوليشان می‌تواند به روشي ساده و آسان براي مرتب سازي كليدها برحسب رتبه غير صعودي يا سورت كردن ركوردها تبديل شود.

چند معماري ساده:

در اين بخش 4 معماري موازي ساده را تعريف می‌كنيم:

1. پردازنده‌هايی به صورت آرايه خطي
2. پردازنده‌هايی به صورت درخت باينري
3. پردازنده‌هايی به صورت مش دو بعدي
4. چندين پردازنده با متغيرهاي اشتراكي

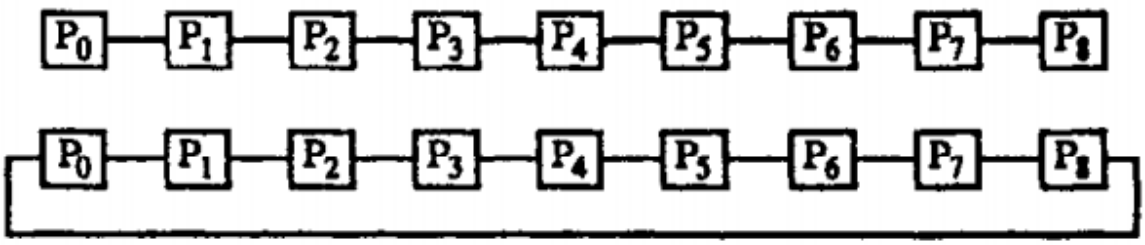
آرايه خطي:

[1] Sorting

شكل 2.2 آرايه خطي از 9 پردازنده كه از 0 تا 8 شماره گذاري شده‌اند را نشان می‌دهد. قطر آرايه خطي p پردازنده، به عنوان بلندترين كوتاه‌ترين مسافت[2] بين جفت پردازنده‌ها تعريف می‌شود كه D = P-1.

بزرگ‌ترين درجه هر گره به عنوان بيشترين تعداد پيوندها يا كانال‌های ارتباطي كه به يك پردازنده مرتبط شده‌اند، تعريف می‌شود كه : d = 2.

حلقه متغير در شكل 2.2 نيز نمايش داده می‌شود. درجه هر گره 2 است اما كم‌ترين قطر برابر است با : $D=\lfloor l/2 \rfloor$.

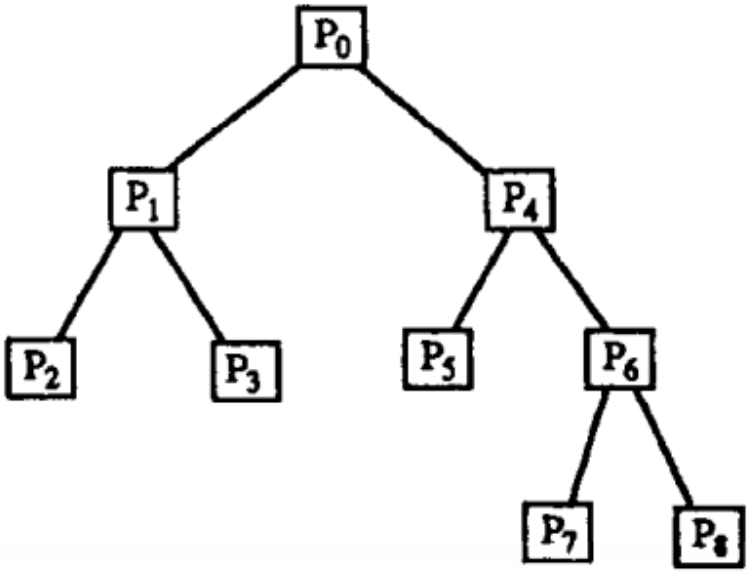


شكل 2. 1 آرايه خطي از 9 پردازنده و متغير حلقه‌اش

درخت باينري :

شكل 2.3 يك درخت باينري از 9 پردازنده را نشان می‌دهد. اين درخت باينري به گونه‌ای بالانس شده كه سطح برگ‌ها حداكثر يك واحد اختلاف دارند.

[2] longest of the shortest distances



شکل 2. 2 درخت باينري بالانس شده (غير کامل) از 9 پردازنده

اگر همه سطوح برگ‌ها یکسان و برابر باشند و همه پردازنده‌های غیربرگ 2 فرزند داشته باشند، درخت باينري،"کامل" خوانده می‌شود. قطر درخت باينري کامل برابر است با :

$$2\log_2(p+1)-2$$

به صورت کلی‌تر، قطر معماري درخت باينري بالانس شده برابر است با :

$$2\lfloor \log_2 p \rfloor$$

یا

$$2\lfloor \log_2 p \rfloor - 1$$

بر اساس قرار دادن گره‌های برگ در آخرین سطح.

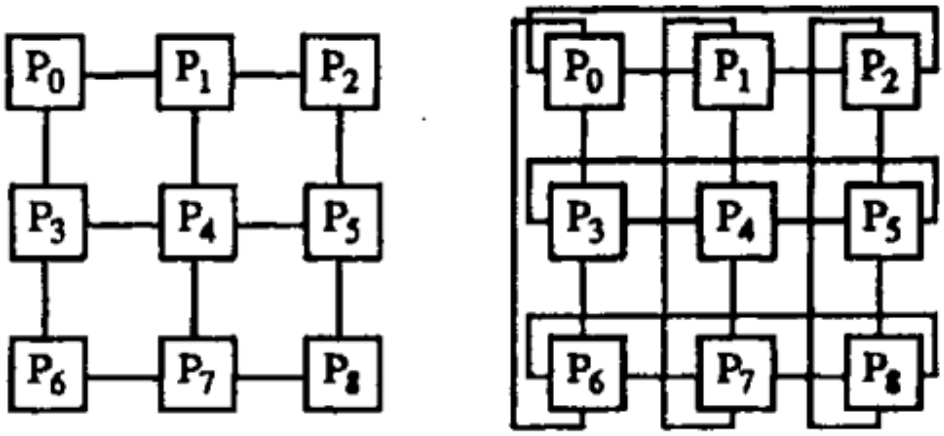
متاسفانه آرايه خطي، با معماري درخت باينري p پردازنده ممکن است چندين تفاوت داشته باشد. معمولا اين مشکلي نيست که هميشه در درخت‌های باينري کامل با آن

سرو کار داریم.

بزرگ‌ترین (max) درجه هر گره در درخت باینري برابر است با d = 3.

مش دو بعدي:

شکل 2.4 يك مش دو بعدي از 9 پردازنده را نشان می‌دهد. قطر پردازنده در مش دوبعدي مربعي برابر است با : $2\sqrt{p} - 2$



شکل 2. 3 مش دوبعدي از 9 پردازنده

به طور کلی‌تر مش لزوما نباید مربعي باشد.

. $\frac{p}{r} * r$ = پردازنده p

قطر مش دوبعدي pپردازنده = $\left(r * \left(\frac{p}{r}\right)\right)$ و لذا :

$D = r + \frac{p}{r} - 2$

مجدداً، مش‌های دو بعدي چندگانه ممکن است براي همان مقدار pپردازنده موجود باشد. به عنوان مثال : 8*2 یا 4*4.

مش‌های مربعي معمولا ترجیح داده می‌شوند زیرا قطر مینیم شده دارند(min بهینه).

متغیرهاي torus همانطور که در شکل 2.4 نشان داده شده‌اند، لینک‌هایی براي توسعه و بسط سطرها و ستون‌ها یا دورگردي در مش هستند. (اشاره به لینک‌های گردشي دور مش)

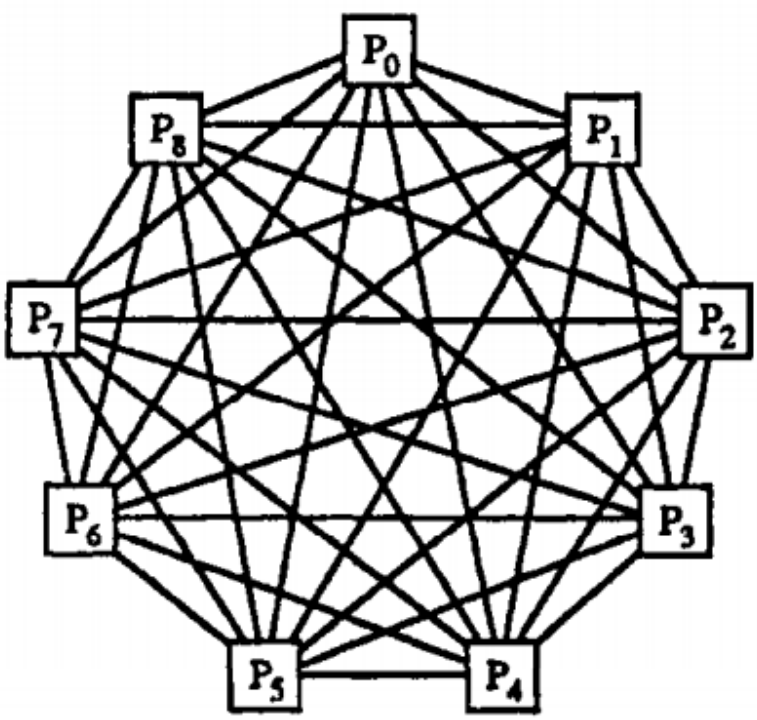
درجه نودها براي هر مش دو بعدي و (مجموع torus) برابر است با : d= 4.

اما یك p پردازنده $r * \frac{p}{r}$ توروسي قطر کمتري دارد:

$$D= \left\lfloor \frac{r}{2} \right\rfloor + \left\lfloor \frac{p}{2r} \right\rfloor$$

حافظه اشتراکي:

یك پردازنده چندگانه با حافظه اشتراکي می‌تواند به صورت یك گراف کامل مدل شودکه هر نود آن با همه نودهاي دیگر ارتباط دارد و در شکل 2.5 نمایش داده شده است.



شكل 2. 4 معماري متغير اشتراكي مدل شده روي يك گراف كامل

براي p = 9، در مش دو بعدي شكل 2.4، پردازنده 0، می‌تواند مستقيما به P_1 و P_3 داده بفرستند يا از آن‌ها بگيرد. اگرچه با واسطه نيز می‌تواند با P_4 داده تبادل كند.

در پردازنده‌های چند گانه با حافظه اشتراكي، هر قطعه داده می‌تواند مستقيما قابل دسترسي به همه پردازنده‌ها باشد. (ما فرض می‌كنيم كه هر پردازنده می‌تواند داده را به طور همزمان در همه اين P-1 پردازنده بدهد يا بگيرد.) قطر D=1 يك گراف كامل بيانگر اين دسترسي مستقيم است.

درجه نود d = P -1،از طرف ديگر بيانگر اين نكته است كه معماری‌هايی از اين دست، اگر هيچ محدوديت مكاني براي دسترسي به داده‌ها نداشته باشند، بايد پياده سازي كاملا گراني داشته باشند.

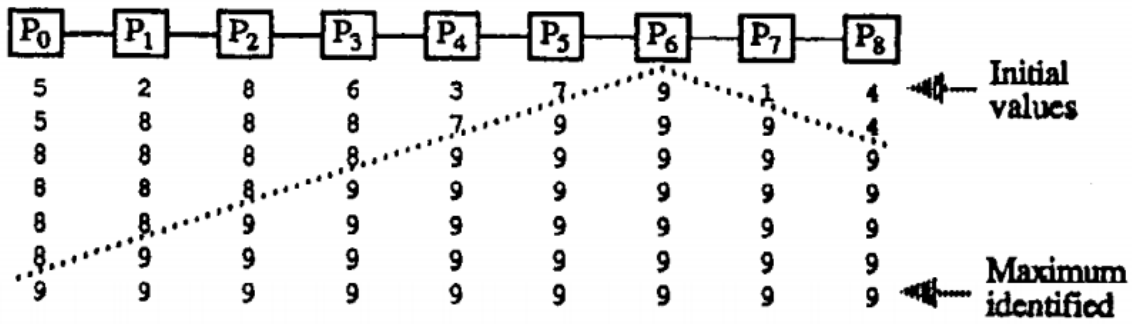
2.3 الگوريتم‌هايی براي آرايه خطي:

محاسبات زنجيره‌ای:

اجازه دهيد ابتدا نوع خاصي از محاسبات زنجيره‌ای را بررسي كنيم. براي مثال پيدا كردن بيشترين مقدار.

هركدام از p پردازنده، مقدار ابتدايي را نگهداري می‌كنند و هدف ما براي همه پردازنده‌ها دانستن بزرگ‌ترين اين مقادير است. يك متغير محلي (بزگترين تا اينجا) می‌تواند با مقدار داده خود پردازنده، مقداردهي شود.

در هر گام، پردازنده بيشترين مقدار تا اين جا[3] را به دو همسايه‌اش می‌فرستد. هر پردازنده، زمان دريافت مقادير از همسايه‌های راست و چپش، مقدار فعلي را به بزرگ‌ترين اين 3 مقدار تنظيم (set) می‌كند. يعني Max(left ,own , right). شكل 2.6 اجراي الگوريتم را براي 9 پردازنده نشان می‌دهد.



شكل 2. 6 پيدا كردن بيشترين مقدار در آرايه خطي از 9 پردازنده

خطوط نقطه چين در شكل 2.6 نشان می‌دهد كه چگونه بزرگ‌ترين مقدار از P_6 به ديگر پردازنده‌ها انتشار می‌يابد. مثلا در P_2 و P_6 دو مقدار Max وجود داشت،انتشار بايد سريع‌تر باشد.

[3] max-thus-far

در بدترین حالت، P-1 گام ارتباط (هر کدام شامل فرستادن مقدار پردازنده به دو همسایه‌اش) و همان شماره از گام‌های مقایسه‌ای سه مرحله‌ای نیاز است. این بهترین راهي است که می‌توان براي گرفتن قطر آرایه خطي pپردازنده که D=p-1 است، امیدوار بود. (قطر بر مبناي کم‌ترین محدوده است.)

در محاسبات زنجیره‌ای در حالت کلي، پردازنده‌ای که در انتهاي سمت چپ آرایه (پردازنده سمت چپي که همسایه ندارد) فعال است و مقادیر داده‌هایش را به راست می‌فرستد.(ابتدا همه پردازنده‌ها در حالت خوابیده و غیر فعال هستند.)

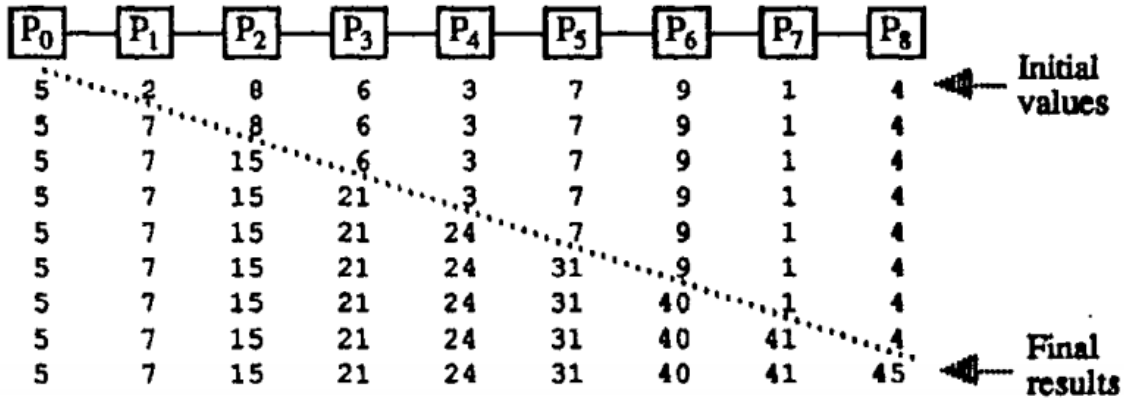
هنگام دریافت مقدار از همسایه سمت چپ، پردازنده فعال می‌شود، عملگر زنجیره‌ای * را اجرا می‌کند. براي مقدار گرفته شده از چپ و مقدار داده خودش، نتیجه را به سمت راست می‌فرستد و مجددا غیر فعال می‌شود. این موج عملیات انتشار به راست، تا زماني که سمت راست‌ترین پردازنده، نتیجه مطلوب را بدست آورد ادامه دارد. نتیجه محاسبه شده سپس به همه پردازنده‌ها به سمت چپ منتشر می‌شود.

محاسبات موازي :

اجازه دهید فرض کنیم که می‌خواهیم i امین نتیجه پیشوند در i امین پردازنده را بدست آوریم : o≤i≤p-1 .

الگوریتم عمومي زنجیره‌ای در پاراگراف قبلي شرح داده شد. در واقع ابتدا محاسبه زنجیره‌ای را اجرا می‌کند و سپس نتیجه نهایي را براي همه پردازنده‌ها پخش می‌کند.

پس ما الگوریتمي براي محاسبات پیشوندي موازي داریم که p-1 گام ارتباطي ترکیبي دارد.



شكل 2. 7 محاسبات مجموع پيشوندها روي آرايه خطي با 9 پردازنده

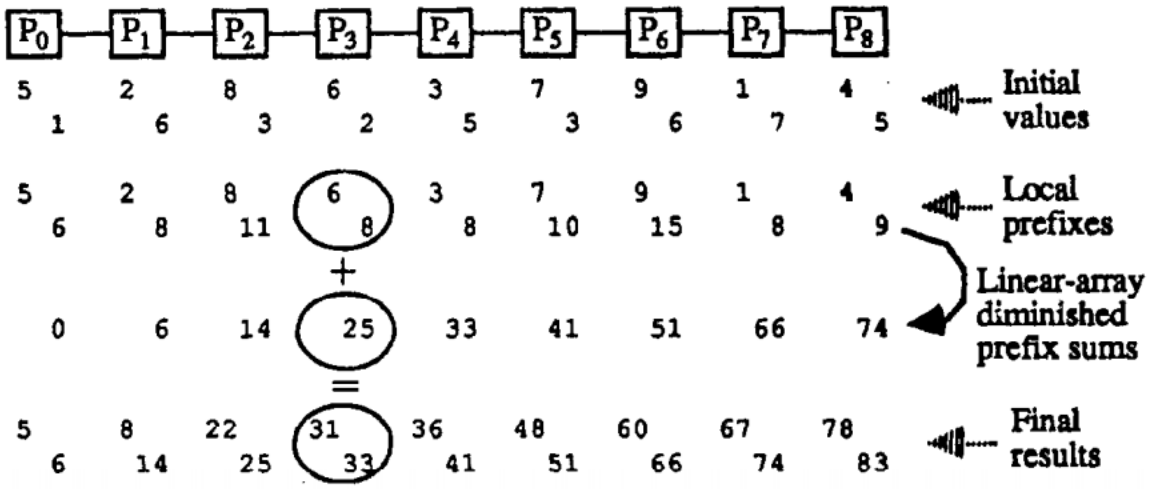
محاسبات پيشوندي موازي مختلف، كه در هر پردازنده i با نتيجه پيشوندي مقدار (i-1) ام تمام می‌شود، اغلب سودمند است. اين محاسبات پيشوندي كاهش يافته به آساني می‌تواند اجرا شود، اگر هر پردازنده به جاي اينكه مقدار دريافت شده از چپ را به راست بفرستد، در خود نگه دارد.نتايج مجموع پيشوندي كاهش يافته به عنوان مثال براي شكل 2.7 مقادير زير است:

0,5,7,15,21,24,31,40,41

تا اين جا ما فرض كرديم كه هر پردازنده يك مقدار منفرد در خود نگهداري می‌كند.

تعميم الگوريتم‌های زنجيره‌ای و پيشوندي موازي به حالتي كه هر پردازنده ابتدا چندين مقدار داده را به آساني نگه می‌دارد. شكل 2.8 نشان مي دهدكه محاسبات مجموع پيشوندي موازي با هر پردازنده در ابتدا 2 مقدار نگهداري می‌كند. الگوريتم عبارتند از : هر پردازنده محاسبه پيشوند را در مجموعه داده خودش از سايز $\frac{n}{p}$ (كه از تركيب گامهاي $\frac{n}{p-1}$ گرفته شده است) انجام مي دهد، سپس محاسبه كاهش يافته پيشوندي موازي را در آرايه خطي مثل بالا (p-1 گام ارتباط - تركيب) انجام دهد، سپس در نهايت، نتيجه پيشوندي خودش (local) را از تركيب آخرين محاسبات با پيشوندهاي محاسبه شده محلي بدست مي آورد. ($\frac{n}{p}$ گام تركيب) . در همه، $2\frac{n}{p}+p-2$ گام تركيب و p-1 گام

ارتباط مورد نياز است.



شكل 2. 8 محاسبه مجموع پيشوندها روي آرايه خطي با دو آيتم براي هر پردازنده

مسير يابي بسته‌اى:

براي فرستادن بسته اطلاعاتي از پردازنده i به پردازندهj در آرايه خطي، ما به سادگي به آن يك برچسب مسيريابي با مقدار j-i متصل می‌كنيم. علامت برچسب مسيريابي معين می‌كند كه بسته در كدام سو حركت می‌كند. (+: راست و - : چپ)

و مقدار آن نيز مشخص می‌كند كه چه عملي بايد انجام شود (0: پاك كردن بسته، غير صفر: هدايت بسته). در هر هدايت، مقدار برچسب يك واحد كاهش می‌يابد. بسته‌های چندگانه كه از پردازنده‌های مختلف سرچشمه گرفته‌اند می‌توانند پشت سر هم به سمت چپ يا راست، بدون دخالت در يكديگرجريان يابند.

همه پخشي: (Broadcasting)

اگر پردازنده i بخواهد مقدار i را به همه پردازنده‌ها همه پخشي كند، آن را به عنوان پيام

rbcast(a) (بخوانيد: r - broadcast) به همسايه راستش می‌فرستد و پيام lbcast(a) را به همسايه چپش می‌فرستد. هر پردازنده که پيام rbcast(a) رادريافت کرد، به سادگي مقدار a را کپي می‌کند و پيام را براي همسايه بعدش (اگر وجود داشت) می‌فرستد.

به طور مشابه پيام(lbcast(a موجب می‌شود تا a به طور محلي کپي شود و پيام براي همسايه چپ فوروارد شود.

در بدترين حالت، تعداد گام‌های ارتباطي براي همه پخشي برابر است با p-1.

مرتب سازي:

ما به دو نسخه مرتب سازي روي آرايه خطي توجه می‌کنيم. با I/O و بدون آن.

شکل 2.9 يك الگوريتم مرتب سازي آرايه خطي را نشان می‌دهد که p کليد وارد شده‌اند، مقادير دريافت شده با مقادير ثبت شده محلي مقايسه می‌شوند(درابتدا همه رجيستر هاي محلي مقدار ∞+ را نگهداري می‌کنند.) مقدار کوچک‌تر از دو مقدار در رجيستر محلي نگه داشته می‌شود و مقدار بزرگتر به راست منتقل می‌شود. فقط يك بار همه p ورودی‌ها گرفته شده، بايد اجازه دهيم p-1 چرخه ارتباطي اضافي براي مقادير کليدها که در جايگاه‌های مربوطه در آرايه خطي مستقر شده‌اند، عبور کنند. اگر ليست مرتب شده از چپ خارج شود، فاز خروج به سرعت می‌تواند بعد از آخرين مقدار کليد دريافت شده شروع شود. در اين مورد آرايه‌ای با نصف سايزليست ورودي، کافي است و ما به طور موثري زمان - 0 براي مرتب سازي داريم. يعني مجموع زمان مرتب سازي برابراست با زمان I/O.

اگر مقدارکليد ها براي هر پردازنده، از قبل در جايشان بود،، پس الگوريتم جابه جايي زوج – فرد می‌تواند براي مرتب سازي استفاده شود. مجموع p گام نياز است.

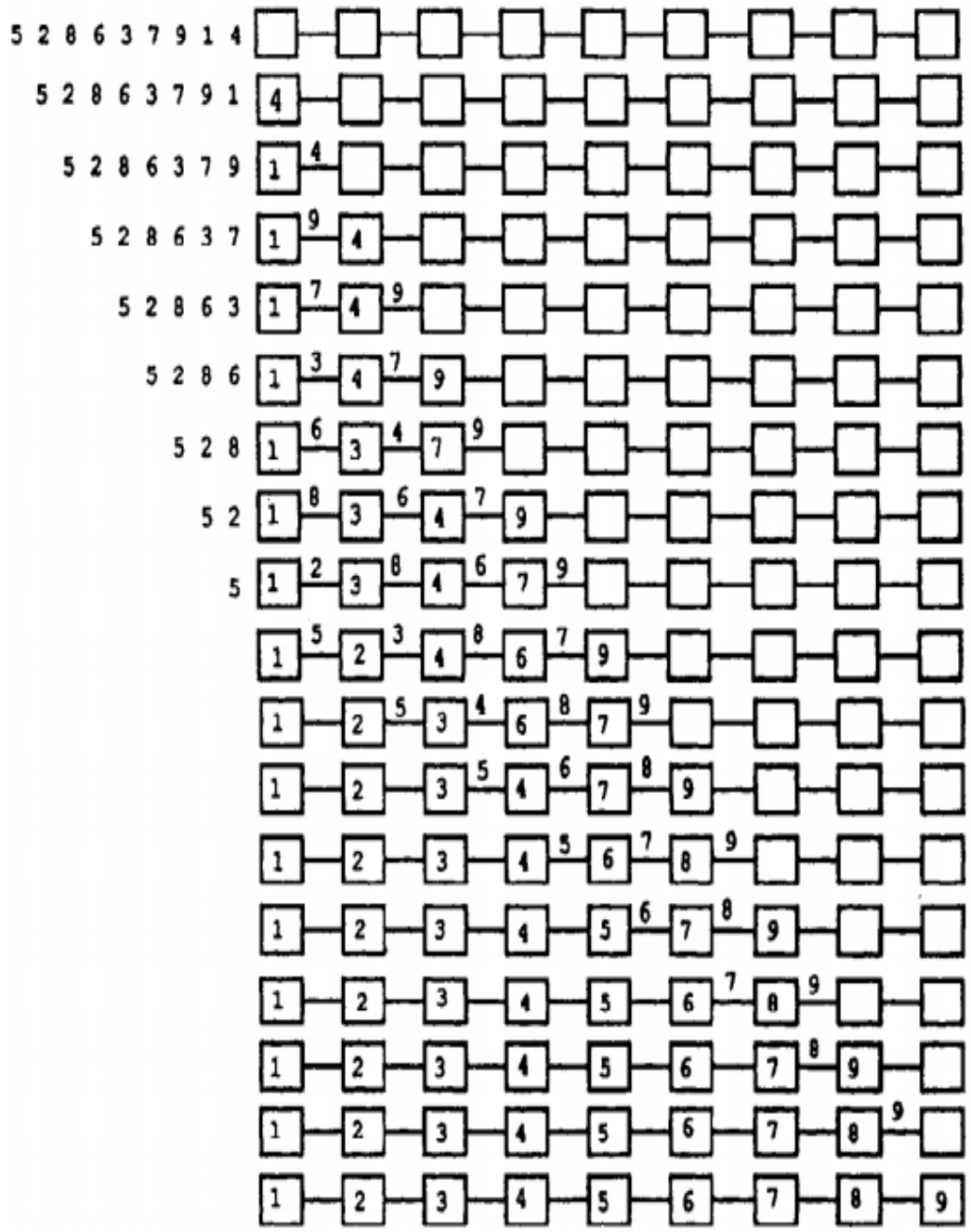
در گام شماره فرد، پردازنده‌های شماره فرد، مقادير را با همسايه‌های راست شماره زوجشان مقايسه می‌کنند. اگر مقادير خارج از ترتيب بود، 2 پردازنده مقاديرشان را جابه جا

می‌کنند.

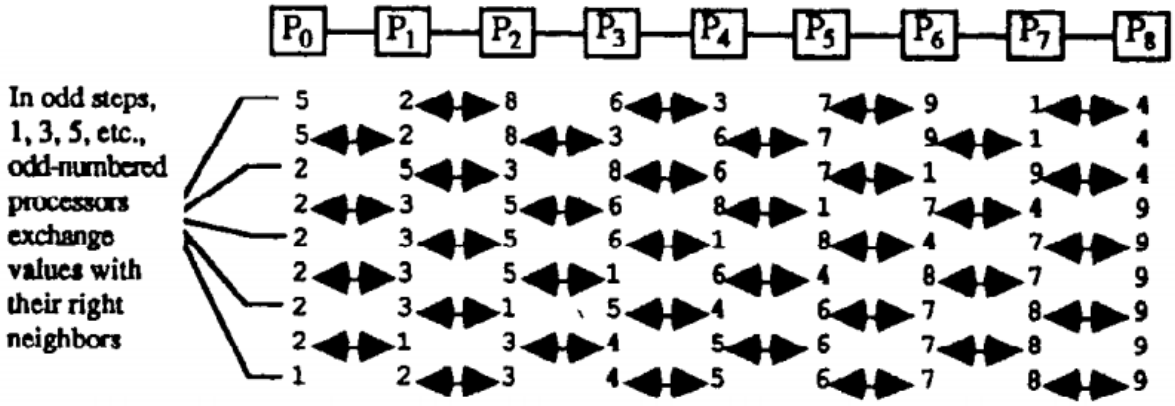
به طور مشابه در گام شماره زوج، پردازنده‌های شماره زوج مقادیر را با همساي هاي راستشان (مقایسه – جا به جا) می‌کنند. (شکل 2.10)

در بدترین حالت ممکن، بزرگ‌ترین مقدار کلید در پردازنده 0 مستقر شده است و باید همه مسیر را تا انتهاي آرایه برود. که این امر به p-1 حرکت به سمت راست نیاز دارد. يك گام باید اضافه شود زیرا هیچ حرکتي در گام اول رخ نداده است. البته يك جابه جايي زوج – فرد باید استفاده شود ولي این مسئله در پیچیدگي زماني الگوریتم ما براي بدترین حالت آرایه خطي 9 پردازنده‌ای تاثیري نخواهد داشت.

توج کنید که الگوریتم جابه جايي زوج – فرد از p پردازنده براي مرتب کردن p کلید در p گام (مقایسه – تعویض) بهره می‌گیرد.



شكل 2. 9 مرتب سازي در آرايه خطي با كليدهاي ترتيبي ورودي از چپ



شكل 2. 10 مرتب سازي جابه جايي زوج – فرد روي آرايه خطي

چه چيز اين الگوريتم خوب است؟ در واقع خواص الگوريتم ياد شده چيست؟ اجازه دهيد كه الگوريتم جابه جايي زوج – فرد را با رجوع به معيارهاي مختلف مطرح شده در بخش 1.6 ارزيابي كنيم.

بهترين الگوريتم‌هاى مرتب سازي ترتيبي، مرتبه $p \log p$ گام مقايسه – تعويض براي مرتب كردن يك ليست سايز p مي گيرند. اجازه دهيد براي ساده سازي فرض كنيم، كه دقيقا $p \log_2 p$ گام دارد.

پس داريم:

$$T(1) = W(1) = p \log_2 p$$

$$T(p) = p$$

$$W(p) = \frac{p^2}{2}$$

$$S(p) = \log_2 p$$

$$E(p) = \frac{\left(\log \frac{p}{2}\right)}{p}$$

$$R(p) = \frac{P}{2 \log \frac{p}{2}}$$

$$U(p) = \frac{1}{2}$$

$$, \; Q(p) = 2 \frac{\left(\log \frac{p}{2}\right)^3}{p^2}$$

در اغلب موقعیت‌های تمرین، تعداد n کلید که مرتب می‌شوند(مسئله سایز) از تعداد p پردازنده بزرگ‌تر هستند(سایز ماشین). الگوریتم مرتب سازي جابه جايي زوج – فرد ابتدا هر پردازنده ليست خودش را از سايز $\frac{n}{p}$ با استفاده از هر الگوريتم مرتب سازي ترتيبي كارا،مرتب مي كند. اجازه دهيد بگوئيم كه اين امر $\left(\frac{n}{p}\right) \log_2 \frac{n}{p}$ گام مقايسه – تعويض دارد.

سپس مرتب سازي جابه جايي زوج – فرد مثل قبل اجرا می‌شود به جز اينكه هر گام مقايسه – تعويض با گام ادغام – شكاف جايگزين می‌شود. كه 2 پردازنده‌های مرتبط زير ليست‌هايشان از سايز $\frac{n}{p}$ را باهم در يك ليست مننفرد از سايز $2\frac{n}{p}$ ادغام مي كنند و سپس ليست را از وسط به دو نيم تقسيم مي كنند، يك پردازنده نيمه كوچكتر را نگهداري مي كند و ديگري نيمه بزرگتر را.

به عنوان مثال اگر P_0 ليست (1,3,7,8) را نگه دارد و P_1 ليست (2,4,5,9) را داشته باشد، يك گام ادغام – شكاف، ليست‌ها را به ترتيب به دو ليست زير تبديل می‌كند:

(1,2,3,4) , (5,7,8,9)

از آنجايي كه ليست‌ها سورت شده‌اند، گام ادغام شكاف نياز به $\frac{n}{p}$ گام مقايسه – تعويض

دارد.

بنابراين مجموع زمان اين الگوريتم برابر است با:

$$\left(\frac{n}{p}\right)\log_2\frac{n}{p}+n$$

توجه كنيد كه اگر $P<\log_2 n$، اولين عبارت (مرتب سازي محلي) بايد قرار بگيرد، تا زماني كه عبارت دوم (ادغام آرايه) براي $P>\log_2 n$ برقرار شود.

براي $P\geq\log_2 n$، پيچيدگي زماني الگوريتم خطي است و برابر با n است. از اين رو الگوريتم بسيار كاراتر از نسخه يك – كليد به هر پردازنده است.

يك ديدگاه نهايي راجع به مرتب سازي:

مرتب سازي اصالتا بسيار مهم است، اما گاهي اوقات در مسيريابي نيز به ما كمك می‌كند،فرض كنيد مقادير داده توسط p پردازنده آرايه خطي كه با ديگر پردازنده‌ها تعيين مسير شده‌اند، نگه داشته می‌شوند. مثل اينكه مقصد هر مقدار متفاوت از بقيه است. اين مسئله "جايگشت مسيريابي" نام دارد زيرا p مقصد متمايز بايد 0,1,2,..., P-1 باشند، شكل دادن ركوردها با آدرس مقصد به عنوان كليد و مرتب سازي آن ركوردها باعث می‌شود كه هر ركورد در مسير صحيح پايان يابد.

در نتيجه، جايگشت مسيريابي در آرايه خطي احتياج به P گام مقايسه – تعويض دارد . بنابراين به طور موثر، Pبسته در مقدار زمان مشابه مسيريابي می‌شوند كه براي مسير يابي يك بسته منفرد در بدترين حالت نياز است.

2.4 الگوريتم درخت باينري:

در الگوريتم‌هايی كه براي پردازنده‌های درخت باينري است، فرض می‌كنيم كه عناصر

داده، در ابتدا فقط در پردازنده‌های برگ نگه داشته می‌شوند.

پردازنده‌های غیر برگ (داخلي) در محاسبات شرکت می‌کنند، اما عناصر داده را در خود نگه نمی‌دارند. این ساده سازي فرض موجب آرامش شده و الگوریتم را ساده‌تر می‌کند. تقریبا نیمي از نودهاي درخت باینري، نود برگ هستند، ناکارآمدي که از این فرض نتیجه می‌شود، زیاد بزرگ نیست و قابل اغماض است.

محاسبات زنجیره‌ای:

معماري درخت دودویي به طور ایده آل براي این محاسبات مناسب است . (به این دلیل که محاسبات زنجیره‌ای اغلب به محاسبات درختي منسوب است.)

هر نود داخلي 2 مقدار از فرزندانش دریافت می‌کند (اگر هر کدام از آن‌ها قبلا مقداري را محاسبه کرده باشند یا نود برگ باشند.) عملگر را روي آن‌ها اجرا می‌کند و نتیجه را رو به بالا به والدش پاس می‌دهد. بعد از $\lfloor \log_2 p \rfloor$ گام، پردازنده ریشه نتیجه محاسبات را دارد.

همه پردازنده‌ها می‌توانند از نتیجه به وسیله عملیات همه پخشي از ریشه،آگاه شوند.

زمان کل :$2\lfloor \log_2 p \rfloor$ گام.

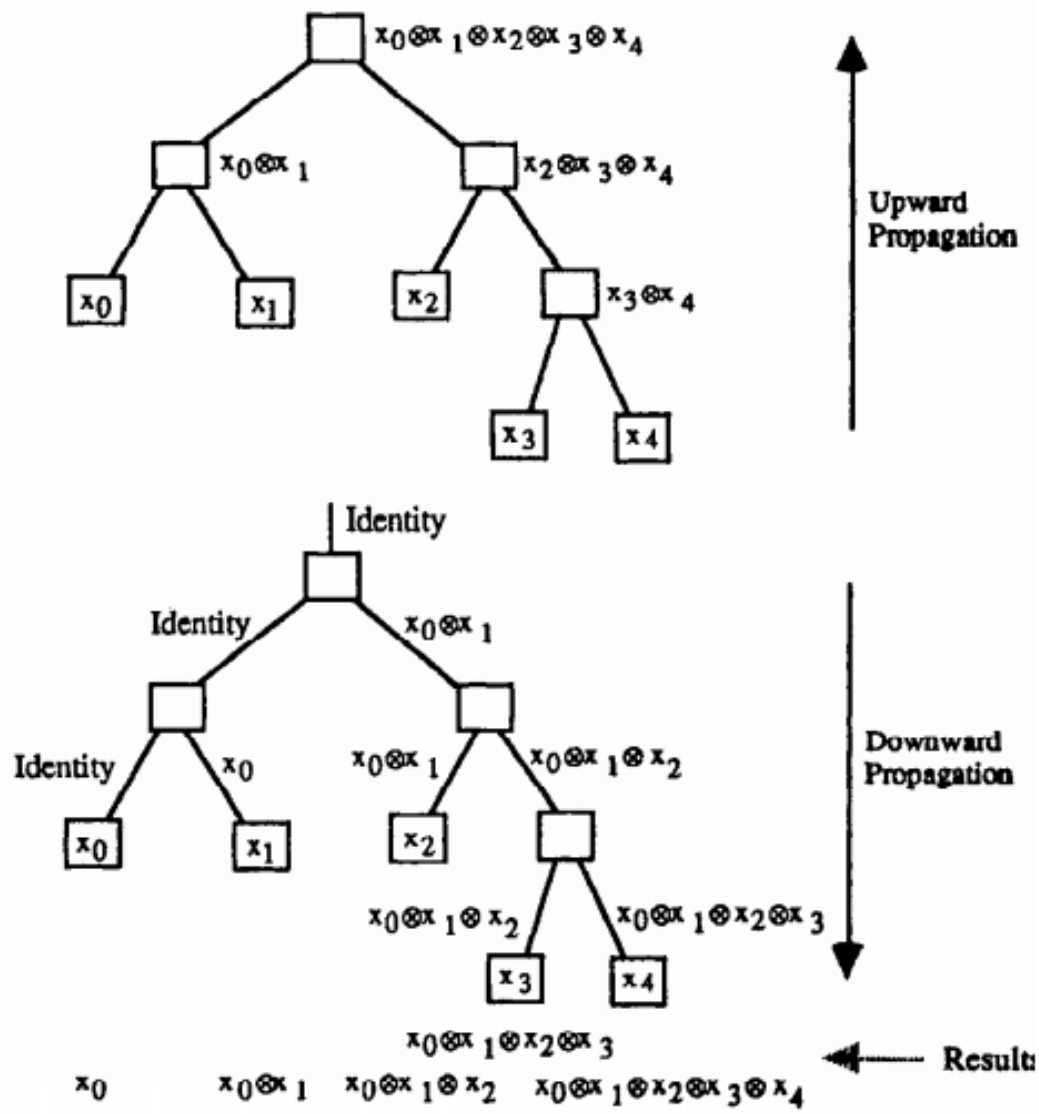
محاسبات پیشوندي موازي:

مجددا، کاملا آسان است و می‌تواند به طور بهینه در $2\lfloor \log_2 p \rfloor$ گام اجرا شود. (فراخواني که قطر درخت باینري $2\lfloor \log_2 p \rfloor$ یا $2\lfloor \log_2 p \rfloor - 1$ است.)

الگوریتم شامل فاز انتشار به سمت بالا است که از حرکت داده به سمت پائین تبعیت می‌کند. همان طور که در شکل 2.11 نشان داده شده است فاز انتشار به سمت بالا همانند حرکت داده به سمت پائین در محاسبات زنجیره‌ای است . در پایان این فاز (مرحله) هر نود نتیجه محاسبات زنجیره‌ای را براي زیر درختش خواهد دانست. مرحله به

صورت زیر است: هر پردازنده، مقداري را که از فرزند چپش دریافت کرده، به خاطر می‌سپارد. هنگام دریافت مقدار از والدش، گره مقدار دریافت شده از بالا را به فرزند چپش پاس می‌دهد و این مقدار و مقداري که از فرزند چپ به فرزند راست آمده را ترکیب می‌کند. ریشه به عنوان عنصر مشخص شده از بالا مشاهده شده و بدین سان فاز رو به پائین با فرستادن عنصر مشخص به چپ و مقدار دریافت شده از فرزند چپ به راست آغاز می‌شود.

در انتهاي فاز رو به پائین پردازنده‌های برگ نتایج مربوط به خود را محاسبه می‌کنند.



شكل 2. 11 محاسبات پيشوندي موازي روي درخت بايناري از پردازنده‌ها

اين مورد آموزنده براي ديدن بعضي از برنامه‌های كاربردي از محاسبات پيشوندي موازي در اين نقطه آمده است.

ليست گرفته شده از 1_s ,0_s رتبه هر كدام در اين ليست (مكان مرتبط بين اولي)

می‌تواند با محاسبات مجموع پیشوندي مشخص شود.

Data:	0	0	1	0	1	0	0	1	1	1	0
Prefix sums:	0	0	1	1	2	2	2	3	4	5	5
Ranks of 1s:			1		2			3	4	5	

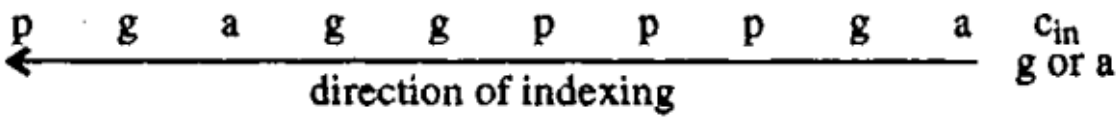
مدار اولويت، ليستي از 0 و 1 به عنوان ورودی‌ها دارد و اولين 1 را از ليست برمي دارد (بيشترين اولويت).

تابع مدار اولويت می‌تواند به صورت زير تعريف شود:

Data:	0	1	0	1	0	0	1	1	1	0
Diminished prefix logical ORs:	0	0	1	1	1	1	1	1	1	1
Complement:	1	1	0	0	0	0	0	0	0	0
AND with data:	0	1	0	0	0	0	0	0	0	0

به عنوان مثال آخر محاسبات كري در طراحي جمع كننده می‌تواند به عنوان محاسبات پيشوندي موازي به شيوه زير تنظيم شود: اجازه دهيد "g" , "p" ,"a" را به عنوان رويدادي كه موقعيت رقم خاص را در جمع كننده توليد می‌كند، منتشر می‌كند يا كري راختني می‌كند، علامت گذاري كنيم .

به عنوان مثال براي جمع كننده اعشاري، اين‌ها با مجموع ارقام كه بزرگ‌تر از 9 يا برابر با 9 يا كمتر از 9 می‌شوند، رابطه دارند لذا داده ورودي براي مدار كري شامل برداري از عناصر 3 مقداري مثل زير است:



كري نهايي در موقعيت‌های مختلف می‌تواند به وسيله محاسبات پيشوندي موازي تعريف شود. با استفاده از عملگر كري "¢"، به صورت زير تعريف می‌شود:

$(X \in \{g, p, a\})$ را به عنوان كري ورودي به موقعيت ببينيد.)

در واقع اگر هر نود در دو درخت شكل 2.11 با مقدار منطقي مربوط به عملگر كري جايگزين شوند، مدار پيش بيني كري 5- رقمي نتيجه خواهد شد.

مسيريابي بسته‌ای:

الگوريتمي كه براي مسيريابي بسته اطلاعاتي از پردازنده i به پردازنده j در درخت دودويي از پردازنده‌ها ست متكي بر طرح شماره گذاري پردازنده‌ها است.

طرح شماره گذاري پردازنده‌ها در شكل 2.3 نشان داده شده است، براي اين منظور بهترين نيست اما در اين جا براي توسعه الگوريتم مسيريابي می‌تواند استفاده شود.
طرح شاخص گذاري شكل 2.3 به عنوان "پيش مرتب كردن"[4] شاخص گذاري شناخته می‌شود و تعريف بازگشتي زير را دارد:
نودهايي كه در زير درخت هستند با اولين شماره گذاري نود ريشه شماره می‌شوند، پس ابتدا زير درخت چپ و در نهايت زير درخت راست . شاخص هر نود از همه شاخص‌های اولاد كمتر بود.
فرض می‌كنيم كه هر نود،از شاخص خود در زير درخت نيز آگاه است، كدام در زير درخت كوچك‌تر است، و می‌داند كه بزرگ‌ترين شاخص نود در زير درخت چپ و راست كدام

[4] preorder

است. بسته در این مسیر از نود i تا نود مقصد و مقیم درنود خودش در همان لحظه، طبق این الگوریتم مسیریابي می‌شوند.

```
if dest = self
then remove the packet {done}
else if dest < self or dest > maxr
   then route upward
   else if dest ≤ maxl
      then route leftward
      else route rightward
      endif
   endif
endif
```

الگوریتم هیچ فرضي راجع به درخت نمی‌سازد مگر اینکه درخت دودویي باشد. به طور خاص، درخت نیازي به کامل بودن یا حتي بالانس بودن ندارد.

همه پخشي:

پردازنده i، داده مطلوب را به سمت بالا به پردازنده ریشه می‌فرستد و سپس ریشه، داده را به سمت پائین براي همه پردازنده‌ها broadcast می‌کند .

مرتب سازي:

می‌توانیم از الگوریتم مشابه حبابي[5] استفاده کنیم که اجازه می‌دهد که کوچک‌ترین عناصر در برگها به صورت حباب به سمت اولین پردازنده ریشه بالا بیایند، لذا به ریشه اجازه می‌دهند که همه عناصر داده را به صورت (غیر نزولي) ببینند. ریشه سپس عناصر را با نظم مناسب به نودهاي برگ می‌فرستد.

قبل از شرح این قسمت الگوریتم، به حرکت رو به بالاي حبابي داده‌ها نگاهي می‌کنیم. اجازه دهید حرکت رو به پائین ساده‌تری را بررسي کنیم:

این حرکت رو به پائین در صورتي که هر نود شماره برگ‌های زیر درخت چپش را بداند به سادگي هماهنگ می‌شود. اگر مرتبه عنصر دریافت شده از بالا (که در شمارنده محلي نگهداري می‌شود) از عدد نودهاي برگ (به چپ) تجاوز نکند، در این حالت آیتم داده به چپ فرستاده می‌شود. در غیر این صورت به راست می‌رود. توجه کنید که بحث ضمني بالا فرض می‌کند که داده از چپ به راست در برگ‌ها مرتب شده است. حرکت رو به بالاي داده در الگوریتم مرتب سازي بالا می‌تواند به صورت زیر انجام شود که عمل پردازنده از نقطه نظر خودش شرح داده شده است.

در ابتدا، هر برگ یك آیتم داده منفرد دارد و همه دیگر نودها خالي هستند. هر نود خالي فضایي براي ذخیره سازي 2 مقدار دارد، مهاجرت رو به بالا از زیردرختهاي چپ و راستش صورت می‌گیرد.

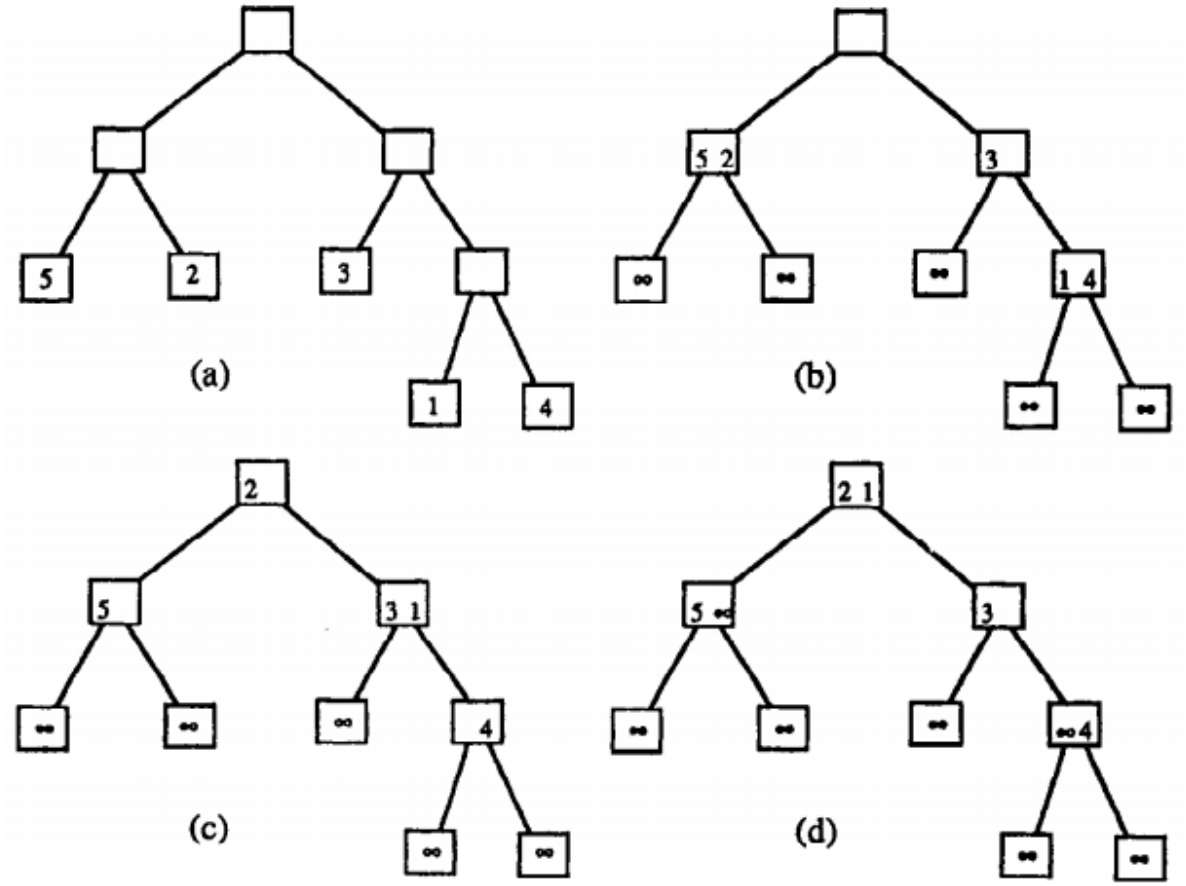
[5] bubble sort

```
if you have 2 items
then do nothing
else if you have 1 item that came from the left (right)
   then get the smaller item from the right (left) child
   else get the smaller item from each child
   endif
endif
```

شکل 2.12 نشان می‌دهد که، در ابتدا گام‌های معدودي از حرکت داده به سمت بالا است (طبق اشاره، وقتي که کوچک‌ترین عنصر در نود ریشه است، آماده شروع حرکت رو به پائین است.).

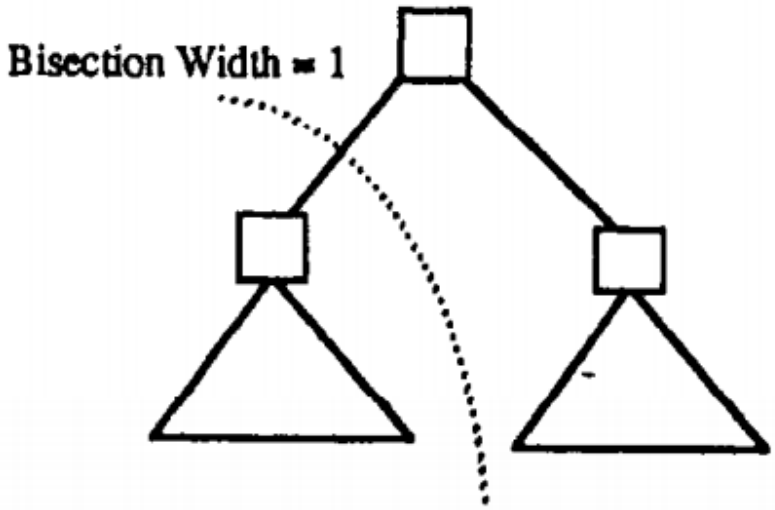
الگوریتم مرتب سازي بالا، براي تعداد عناصري که مرتب شده‌اند، زمان خطي را صرف می‌کند.

ممکن است علاقمند باشیم بدانیم که اگر الگوریتم خیلي کاراتري بخواهد توسعه یابد، گرفتن قطر معماري درخت لگاریتمي است.(یعني در بدترین حالت، عنصر داده مجبور است $2\lfloor \log_2 p \rfloor$ گام حرکت کند تا در مکان خودش سورت شود.) پاسخ متاسفانه این است که ما نمي توانیم اساسا بهتر از بالا عمل کنیم.



شكل 2. 12 گام‌هاى محدود الويه از الگوريتم مرتب سازي روي درخت باينري

دليل مبتني بر استدلال حد پائين است كه در بسياري از متون بسيار مناسب است. همه آن چيزي كه ما نياز داريم اين است كه معماري درخت را به 2 بخش مساوي يا تقريبا برابر بخش بندي كنيم. (مركب از $\lceil \frac{p}{2} \rceil$ يا $\lfloor \frac{p}{2} \rfloor$ پردازنده) كه برش ارتباط منفرد نزديك به پردازنده ريشه باشد .(شكل 2.13)



شكل 2. 13 عرض دو بخشي معماري درخت باينري

ما ميگوييم كه "عرض دو بخشي" معماري درخت باينري 1 است. حال در بدترين حالت، نحوه قرارگيري داده‌ها در ابتدا ممكن است طوري باشد كه براي مرتب شدن در مكان خودشان، همه مقادير نيمه چپ(يا راست) درخت بايد به نيمه راست (يا چپ) حركت كند . از اين رو همه عناصر داده بايد در طول يك لينك منفرد پاس شوند. اينكه چگونه ما حركت داده‌ها را سازماندهي مي‌كنيم مهم نيست در هر حال براي پاس دادن همه عناصر داده به گلوگاه زمان خطي لازم است.

اين مثالي از "دو نيم كردن بر مبناي حد پائين " بود.

2.5 الگوريتم مش دو بعدي

در همه الگوريتم‌های مش دو بعدي ارائه شده در اين فصل، ما از الگوريتم‌های آرايه خطي بخش 2.3 به عنوان زيربنا استفاده مي‌كنيم. اين امر موجب تسهيل الگوريتمها مي‌شود اما نه لزوما كارامدترين آن‌ها.

معماری‌های مبتني بر مش و الگوريتم‌های آن‌ها با جزئيات بيشتر در بخش 3 (فصل 9-

12) شرح داده خواهند شد.

محاسبات زنجیره‌ای:

براي اجراي محاسبات زنجیره‌ای در مش دو بعدي، محاسبات زنجیره‌ای را در هر رديف و سپس در هر ستون انجام دهيد . به عنوان مثال، در يافتن مقدار max مجموعه مقاديرp، سورت شده در هر پردازنده، ابتدا بيشترين سطرهاي محاسبه شده و براي همه پردازنده‌ها در سطر در دسترس قرار داده شده و سپس max ستون‌ها مشخص می‌شوند. اين امر، برطبق نتايج بخش 2.3، $4\sqrt{p}-4$ گام در يك pپردازنده مش دو بعدي در پي دارد.

پردازش مشابه می‌تواند براي محاسبه مجموع اعداد p استفاده شود.

توجه کنيد که در محاسبات زنجیره‌ای در حالت کلي با عمليات غير جابه جايي پذير (جابه جايی ناپذير)، براي اينکه الگوريتم درست کار کند، اعداد p بايد در سطر اصلي مرتب شوند .

محاسبات پيشوندي موازي:

مجددا اين امر بسيار ساده است و می‌تواند در 3 فاز انجام شود، فرض کنيد که پردازنده‌ها (و مقادير مرتب شده‌شان) در سطر اصلي شاخص گذاري شده‌اند:

1. محاسبه پيشوندي موازي را در هر سطر انجام دهيد.
2. محاسبات کاهش يافته پيشوندي موازي را در سمت راست‌ترين ستون انجام دهيد.
3. نتايج را در سمت راست‌ترين ستون به همه عناصر سطرهاي مربوطه broadcast کنيد و با مقدار ابتدايي پيشوندي سطري محاسبه شده، ترکيب نمائيد.

به عنوان مثال در انجام جمع‌های پیشوندي، اولین سطر مجموع پیشوند از چپ به راست محاسبه می‌شود. در این نقطه، پردازنده‌ها در سمت راست‌ترین ستون مجموع سطر را نگه می‌دارند. محاسبه پیشوندي کاهش یافته در آخرین ستون حاصل جمع همه سطرهاي قبلي در هر پردازنده است. ترکیب جمع همه سطرهاي قبلي با حاصل جمع پیشوندي سطر، شامل همه مجموع پیشونداست.

مسیریابي بسته‌ای:

براي مسیریابي بسته داده از پردازنده در سطر R و ستون C به پردازنده در سطر r´ و ستون c´، ابتدا آن را به سطر R و ستون c´ هدایت می‌کنیم. سپس آن را در ستون c´ از سطر r تا سطر r´ هدایت می‌کنیم . این الگوریتم به عنوان "مسیریابي اولین سطر" شناخته می‌شود.

به طور واضح اگر بخواهیم مسیر یابي اولین ستون را اجرا کنیم یا از ترکیب عمودي و افقي گام‌ها براي گرفتن نود مقصد طي کوتاه‌ترین مسیر استفاده کنیم. اگر نودهاي مش همانند شکل 2.4 شاخص گذاري شوند، به جاي اینکه عبارات شماره‌های سطر و ستون‌ها، ما به سادگي شاخص پردازنده i میاني را که مسیر اولین سطر تعیین کرده بود، مشخص می‌کنیم. مسئله ما به دو مسئله تجزیه شد:

مسیریابي افقي از i تا l و سپس مسیریابي عمومي از l تا j. سپس بسته‌های چند گانه باید بین نودهاي مبدا و مقصد مسیریابي شوند.

الگوریتم بالا می‌تواند براي هر بسته مستقلي از دیگران به کار رود. گرچه بسته‌های چندگانه ممکن است براي لینک‌های خروجي در مسیرهایشان یا مقاصد مربوطه رقابت کنند. پردازنده‌ها باید فضاي بافر کافي براي ذخیره بسته‌هایی که باید در نقاط (نوبتشان) صبر کنند قبل از شروع فوروارد شدن در ستون. جزئیات در فصل 10 تشریح خواهد شد.

همه پخشي:

همه پخشي در دو فاز انجام می‌شود:

فاز 1 : همه پخشي بسته‌ها به همه پردازنده‌ها در نودهاي سطر مبدا و

فاز 2: همه پخشي در همه ستون‌ها.

این امر حداکثر $2\sqrt{p}-2$ گام دارد. اگر مقادیر چندتایي به وسیله پردازنده، همه پخشي شود سپس حرکت داده موردنیازمي تواند به صورت لوله کشي باشد. مثل اینکه هر همه پخشي اضافي نیاز به تنها يك گام اضافي دارد.

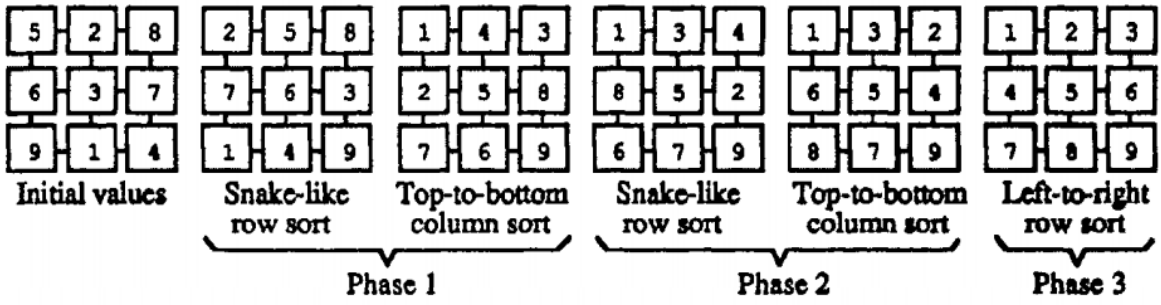
مرتب سازي:

ما بدون ارائه هیچ دلیلي، نسخه ساده‌ای از الگوریتم مرتب سازي را بنام "shortest" توضیح می‌دهیم . دلیل کامل و متغیرهاي کاراتر بیشتر در فصل 9 بیشتر شرح داده خواهند شد.

الگوریتم شامل $\lceil \log_2 r \rceil + 1$ مرحله در مش 2 بعدي با r سطر است. در هر مرحله به جز مرحله آخر، همه سطرها مستقلا مثل مار مرتب مي شوند:

سطرهاي شماره‌های زوج 0و2و... از چپ به راست و سطرهاي شماره‌های فرد 1و3و ... از راست به چپ. سپس همه ستون‌ها از بالا به پائین به طور مستقل مرتب می‌شوند.

به عنوان مثال در يك مش 3*3، دو مرحله لازم است که در شکل 2.14 نشان داده شده است.



شکل 2. 14 الگوریتم shortest در مش 3*3

در فاز نهايي سطرها مستقلا از چپ به راست مرتب شده‌اند. چنانچه از قبل می‌دانیم مرتب سازي روي سطر و مرتب سازي روي ستون در مش مربعي p پردازنده $\sqrt{p}$ گام مقايسه – تعويض دارد.

الگوريتم shortest به $(2\lceil \log_2 p \rceil + 1)\sqrt{p}$ گام مقايسه – تعويض براي مرتب كردن در سطر اصلي نياز دارد.

2.6 الگوریتم‌های با متغیر اشتراکي:

در این بخش مجددا روي توسعه الگوریتم‌هایی که لزوما کارا نیستند تمرکز می‌کنیم.

معماری‌های متغیر اشتراکي و الگوریتم‌های آن‌ها با جزئیات بیشتر در فصل 5و 6 خواهند آمد.

محاسبات زنجیره‌ای:

هر پردازنده آیتم‌های داده را از همه پردازنده‌های دیگر می‌گیرد و مستقلا محاسبات زنجیره‌ای را انجام می‌دهد . بدیهي است که همه پردازنده‌ها با نتایج مشابه تمام خواهند

شد. اين ديدگاه بسيار بي فايده است و معماري پيچيده‌ای دارد كه در شكل 2.5 آمده است . چون پيچيدگي زماني خطي الگوريتم، اساسا الگوريتم محاسبات زنجيره‌ای در مقياسه با معماري آرايه‌ای – خطي ساده‌تر است و از الگوريتم مش دو بعدي بدتر است.

محاسبات پيشوندي موازي:

شبيه محاسبات زنجيره‌ای به جز اينكه هر پردازنده تنها آيتم‌های داده را از پردازنده‌های با شاخص‌های (index) كوچكتر می‌گيرد.

مسيريابي بسته‌ای:

بديهي اين ديدگاه، مسير ارتباطي مستقيم بين هر جفت پردازنده است.

همه پخشي:

به طور جزئي هر پردازنده می‌تواند به طور مستقيم آيتم داده رابه همه پردازنده‌ها بفرستد. در واقع به خاطر اين دسترسي مستقيم، همه پخشي نياز نيست. هر پردازنده از قبل به هر داده‌ای كه بخواهد دسترسي دارد.

مرتب سازي:

الگوريتمي كه براي مرتب سازي متغيرهاي اشتراكي، شرح داده می‌شود، شامل دو مرحله است: رتبه دهي و جايگشت داده.

رتبه دهي شامل تعيين مرتبه مرتبط با هر كليد در ليست مرتب شده نهايي.

اگر هر پردازنده يك كليد نگه دارد، يك بار ديگر رتبه‌ها تعيين می‌شوند، كليد رتبهء j كه، می‌تواند به پردازنده‌j در مرحله جايگشت داده فرستاده شود، نياز به يك گام منفرد ارتباط

موازي دارد. پردازنده i مسئول رتبه دهي به كليد خودش x_i است. اين امر به وسيله مقايسه x_i با همه كليدهاي ديگر و شمارش اعداد كليدهايي كه از x_i كوچک‌تر هستند، انجام می‌شوند.

در حالتي كه مقادير كليدها برابر باشند، شاخص‌های پردازنده براي ساختن رتبه‌های نسبي، استفاده می‌شوند. به عنوان مثال اگر پردازنده 3 و 9 هر دو مقدار كليد 23 را نگه دارند، كليد وابسته به پردازنده 3،براي اهداف رتبه دهي، كوچک‌تر فرض می‌شود.

بايد واضح باشد كه هر كليدي با رتبه يك تا در محدوده 0 (كليد كوچک‌تر ديگري وجود ندارد.) تا p-1 (همه ديگر كليدهاي p-1 كوچک‌تر هستند) می‌تواند باشد.

مجددا، با وجود اينكه معماري متغير اشتراكي، پچيدگي بيشتري در مقايسه با معماری‌های آرايه خطي و درخت باينري دارد، با الگوريتم مرتب سازي بالا در با الگوريتم‌های 3 معاري ساده‌تر، زمان خطي احتياج دارد.

در بخش 6 خواهيم ديد كه الگوريتم‌های مرتب سازي با زمان لگاريتمي در حقيقت براي معماری‌های متغير اشتراكي می‌توانند توسعه يابند كه موجب تسريع خطي بيش از الگوريتم‌های ترتيبي می‌شود و براي مرتب كردن n داده، احتياج به گام‌های مقايسه – تعويض از مرتبه $n \log n$ دارد.