سومین کنفرانس سالانه مهندسی برق ایران
دانشگاه علم و صنعت ایران (۲۸-۲۵ اردیبهشت ماه ۱۳۷۴)

تبدیل رشته ها از طریق پردازش موازی

سید رسول موسوی
دانشجوی کارشناسی ارشد

محمد رضا میبدی
عضو هیئت علمی

تهران ، خیابان حافظ ، دانشگاه صنعتی امیرکبیر ، دانشکده مهندسی کامپیوتر

چکیده

مرتب سازی، ترانهاده کردن ماتریس درحافظه، وارون ساختن رشته و بسیاری ازمسائل دیگر در رده مسائل تبدیل رشته می گنجد. در [1] نوعی گروه بندی برای مسائل تبدیل رشته صورت گرفته است. این مقاله ضمن معرفی کارهای انجام گرفته به نقد و بررسی آنها می پردازد. مشکلات موجود در طبقه بندی ارائه شده در [1] مورد بحث قرار می گیرد و در نهایت یک طبقه بندی جدید معرفی می گردد.

کلید واژه

پردازش موازی، الگوریتمهای موازی، تبدیل رشته.

۱- مقدمه

امروزه پردازش موازی بعنوان ابزاری برای افزایش سرعت مورد استفاده قرار می گیرد [2,3,4,5]. در این مقاله مسائلی که بتوانند بصورت تبدیل رشته فرموله شوند و مناسب برای پردازش موازی باشند مورد مطالعه قرار می گیرد. تبدیل رشته بعنوان یک موضوع در پردازش موازی برای اولین بار در [6] مطرح گردیده است. بحث تفصیلی این موضوع را می توان در [1,7,8,9] جستجو نمود.

شیوه اصلی حل مسائل تبدیل رشته پیدا کردن تابعی مثل f است که مکان هر نماد از رشته ورودی را به مکان متناظر در رشته خروجی می نگارد. همزمانی با محاسبه همزمان مکانهای نمادهای مختلف در رشته خروجی بدست می آید. از آنجاییکه رشته خروجی تبدیلی از رشته ورودی برای یک مسأله داده شده را ارائه می دهد که با جابجایی نماد های رشته ورودی حاصل می شود، تابع f یک نگاشت N به N است که N زیر مجموعه ای از اعداد طبیعی است یعنی $N=\{1,2,\ldots,n\}$ که در آن n طول رشته می باشد. این تابع الگوریتمی که تبدیلات مورد نظر را پیاده سازی می نماید بیان می سازد. موازی سازی با تعیین مکانهای خروجی نمادهای مختلف بطور همزمان حاصل می شود. بعنوان مثال مسأله معکوس کردن یک رشته کاراکتری را در نظر بگیرید. الگوریتم ترتیبی برای محاسبه معکوس رشته به زمان $O(n)$ نیاز دارد، با استفاده از تابع f که در زیر داده شده است یک کامپیوتر موازی با p پردازنده می تواند این عمل را در زمان $O(n/p)$ انجام دهد. بنابراین در صورت وجود n پردازنده پیچیدگی الگوریتم $O(1)$ خواهد بود. برای a_j یعنی j امین کاراکتر در رشته ورودی داریم:

$$f(a_j)=\begin{cases} n-j+1 & \text{اگر } 1<=j<=n \\ \text{تعریف نشده} & \text{در غیراینصورت} \end{cases}$$

فرض کنید رشته های $I=a_1\ a_2\ \ldots\ a_p$ و $O=b_1\ b_2\ \ldots\ b_p$ به ترتیب رشته ورودی و رشته خروجی یک الگوریتم باشند، مکان های ورودی و خروجی یک نماد به ترتیب در I و O بصورت زیر تعریف شده اند:

$inpos(a)=i$ اگر $a=a_i$

$outpos(a)=j$ اگر $a=b_j$

اگر f تابعی باشد که outpos(a) را تعریف کند، آنگاه برد تابع f بصورت $\{1,2,\ldots,p\}$ خواهد بود اما دامنه f برای الگوریتمهای مختلف متفاوت است. با تعریف f می توان مکان تمام نمادها را در رشته خروجی در زمان $L+\lceil n/p\rceil T$ تعیین نمود که در این فرمول، T زمان محاسبه f، p تعداد پردازنده ها، n طول رشته ورودی و L زمان جمع آوری اطلاعات مورد نیاز برای محاسبه تابع f می باشد.

مسائلی از این نوع را می توان بر اساس میزان اطلاعات مورد نیاز برای محاسبه f دسته بندی کرد. مجموعه ای از انواع اطلاعات، F، که حاوی شش نوع مختلف اطلاعات لازم برای کامپیوترهای موازی در محاسبه outpos(a) می باشد در [1] معرفی شده است. فرض کنید:

$F=\{F_1,F_2,\ldots,F_6\}$

$F_1=a,\quad F_2=inpos(a),\quad F_3=n$

$F_4 = T_1, T_2, \ldots, T_m$ اطلاعات ثابت

$F_5 = I = I_1, I_2, \ldots, I_p$, $F_6 = \text{outpos}(b),\ \forall b \mid \text{inpos}(a) \neq \text{inpos}(b)$

I_k زیرمجموعه ای از I با طول k می باشد. F_6 یعنی برای محاسبه f برای یک رشته ورودی خاص پردازنده باید outpos مربوط به کلیه نمادهای دیگر رشته ورودی را بداند (بدترین حالت). برای توضیح بیشتر به [10] مراجعه نمایید.

حال می توان یک شمای طبقه بندی کلی تعریف کرد که درآن هر مسأله نیاز به مجموعه ای از اطلاعات دارد که زیرمجموعه ای از مجموعه توانی F خواهدبود. لازم به توضیح است تبدیلاتی وجود دارند که حتی قابل محاسبه نیستند [8].

با توجه به اینکه مسائلی وجود دارند که فقط از F_1 ، F_2 یا F_6 استفاده می کنند و درضمن هر مسأله قابل رسیدن باید حداقل از F_1 ، F_2 یا F_6 استفاده نماید، F_1، F_2 و F_6 اطلاعات اصلی خوانده می شوند.

در بخش دوم مقاله با نقد و تصحیح طبقه بندی انجام شده در [1] یک طبقه بندی جدید از مسائل تبدیل رشته ارائه شده است. در بخش سوم ایده محلی سازی و فرآیند کاهش اطلاعات مورد بحث قرار می گیرد. بخش چهارم مجموعه F را مورد نقد و تصحیح قرار می دهد. بخش پنجم نتیجه گیری خواهد بود.

۲-طبقه بندی و مثال ها

حالت اول: در این حالت outpos(a) فقط تابع مکان a در رشته ورودی می باشد یعنی (outpos(a) =f(inpos(a)) رابطه همانی ($a_1\ a_2\ \ldots\ a_n$ ===> $a_1\ a_2\ \ldots\ a_n$) مثالی برای این حالت است. توجه کنید که با داشتن n پردازنده می توان تمام مکانهای خروجی را بطور موازی در زمان O(1) محاسبه نمود. با داشتن p پردازنده این کار در زمان O(n/p) قابل انجام است.

تذکر۱: در [1] تبدیل $a_1\ a_2\ \ldots\ a_n$ ===> $a_n a_{n-1}\ \ldots\ a_2\ a_1$ بعنوان مثالی برای حالت اول آورده شده است که برای آن تابع f بصورت زیر تعریف شده است:

$$f(\text{inpos}(a)) = \begin{cases} \text{inpos}(a)+1 & \text{اگر inpos(a) فرد است} \\ \text{inpos}(a)-1 & \text{اگر inpos(a) زوج است} \end{cases}$$

همانطور که ملاحظه می شود تابع f برای رشته هایی که طول آنها زوج است قابل اعمال است. در واقع f بطور ضمنی تابع n نیز می باشد و نتیجتاً در طبقه بندی اول قرار نمی گیرد.

قضیه: اگر outpos(a)=f(inpos(a)) باشد، آنگاه تنها تبدیل ممکن تبدیل همانی است. برای اثبات این قضیه به [10] مراجعه نمایید.

حالت دوم: outpos(a)=f(inpos(a),n)

معکوس سازی رشته ($a_1\ a_2\ \ldots\ a_n$ ===> $a_n\ \ldots\ a_2\ a_1$) مثالی برای این حالت است که برای آن تابع f بصورت

$f(inpos(a),n)=n-inpos(a)+1$ تعریف شده است .

حالت سوم: در این حالت f تابعی است از inpos(a) و اطلاعات ثابت $t_1,t_2,\ldots,t_m$ یعنی

$$outpos(a)=f(inpos(a), t_1,t_2,\ldots,t_m)$$

مثالی برای این حالت ترانهاده کردن ماتریس می باشد. فرض می کنیم که ماتریس بصورت سطری ذخیره شده است. با این فرض آرایه دو بعدی $A(1..l_1, 1..l_2)$ را می توان بصورت l_1 سطر در نظر گرفت ($row1,row2,\ldots,rowl_1$) که در اینجا هر سطر دارای l_2 عنصر است. بنابراین برای $a=A[i,j]$ داریم: $inpos(a)=l_2(i-1)+j$ که در اینجا

$$i=\lfloor (inpos(a)-1)/l_2 \rfloor+1$$
$$j=(inpos(a)-1) \bmod l_2)+1$$

برای محاسبه ترانهاده، $A[i,j]$ به $A^T[j,i]$ تبدیل می شود که مکان $l_1(j-1)+i$ را اشغال می کند. در اینصورت

$$f(inpos(a),l_1,l_2) = l_1((inpos(a)-1) \bmod l_2)+\lfloor (inpos(a)-1)/l_2 \rfloor+1$$

پس از اتمام جمع آوری اطلاعات می توان این تابع را در زمان $O(1)$ با $l_1 \times l_2$ پردازنده و در زمان $O(n/p)$ با p پردازنده محاسبه نمود.

تذکر۲: در حالت سوم، اطلاعات ثابت $t_1,t_2,\ldots,t_m$ (بطور مثال l_2 و l_1 در ترانهاده کردن ماتریس) بایستی به نحوی با I_n رابطه داشته باشد زیرا در غیراینصورت می توان نوشت :

$$f(inpos(a),t_1,t_2,\ldots,t_m) = f_{t_1,t_2,\ldots,t_m}(inpos(a))$$

که دراینصورت تبدیل موردنظر در طبقه بندی شماره یک قرار خواهد گرفت .

حالت چهارم: $outpos(a)=f(a)$

مثال: رشته ورودی یک ترتیب تصادفی از اعداد صحیح $1,2,\ldots,n$ است و خروجی مرتب شده این اعداد بطور صعودی می باشد که دراینصورت $outpos(a)=a$. برای توضیحات و مثالهای بیشتر به [10] مراجعه نمایید.

حالت پنجم: $outpos(a)=f(inpos(a),I)$

مثال۱ - مرتب سازی : برای مرتب کردن نمادها بطور صعودی می توان از تابع زیر برای تعیین مکان یک نماد استفاده نمود. برای $a=a_i$:

$$f=\begin{cases} n_i+1 & \text{اگر } 1<=i<=n \\ \text{تعریف نشده} & \text{در غیر اینصورت} \end{cases}$$

n_i تعداد نمادهای کوچکتر یا مساوی iامین نماد در رشته ورودی است که قبل از این نماد قرار گرفته اند. در [11] نشان داده شده است که n_i را می توان در زمان $O(\log_2 n)$ با استفاده از $n \log_2 n$ پردازنده مشخص نمود. پس از تعیین مقدار n_i ها، مکان نهایی هر نماد با استفاده از n پردازنده در زمان $O(1)$ محاسبه می شود.

مثال۲ : تبدیل نشانگذاری میانوندی به نشانگذاری پسوندی [1] .

حالت ششم: در این حالت f تابعی است از inpos(a)، اطلاعات ثابت $t_1,t_2,\ldots,t_m$ و n یعنی

$$outpos(a)=f(inpos(a),t_1,t_2,\ldots,t_m,n)$$

مثال: دوران رشته ورودی r نماد به سمت راست: $outpos(a)= (inpos(a)+r-1) \bmod n +1$

حالت هفتم: outpos(a) تابعی است از a ، مکان a در رشته ورودی و اطلاعات محلی اضافی در رابطه با همسایه های a و مکان آن در رشته ورودی یعنی

$$outpos(a)=f(a_{i-k_1} \ldots a_{i+k_2}, inpos(a_{i-k_1}), \ldots, inpos(a_{i+k_2}))$$

که k_1,k_2 برای $a=a_i$ ثابت هستند.

مثال : رشته خروجی از نمادهای ورودی که به شیوه محلی مرتب شده اند تشکیل می یابد. بعنوان مثال، پنج نماد اول در رشته ورودی اولین پنج نمادی هستند که در رشته خروجی مرتب می شوند و سپس پنج نماد بعدی و ... .

حالت هشتم: $outpos(a) =f(outpos(b), \forall b \mid inpos(a) \neq inpos(b))$

مثال: رشته خروجی ترتیب تصادفی از رشته ورودی می باشد. فقط درصورتی می توان شماره خروجی j را به نماد a نسبت داد که هیچ نماد دیگری به شماره j نسبت داده نشده باشد.

تذکر۳: حالت زیر را که در [1] به آن اشاره شده است درنظربگیرید

$$outpos(a) =f(n , outpos(b), \forall b \mid inpos(a) \neq inpos(b))$$

می توان نشان داد که اطلاعات $outpos(b), \forall b \mid inpos(a) \neq inpos(b)$ برای بدست آوردن outpos(a) کافی بوده و نیازی به n نمی باشد. الگوریتم زیر این عمل را انجام می دهد: (فرض کنید B مجموعه اعدادی است که F_6 بدست می دهد.)

۱. بزرگترین عضو مجموعه B را پیدا کرده، آنرا n_1 بنامید.

۲. مجموعه $A=\{i \mid i \in$ مجموعه اعداد طبیعی ، $i<=n_1\}$ را بدست آورید.

۳. مجموعه $D=A-B$ را تشکیل دهید.

۴. اگر $D=\{\}$، آنگاه $outpos(a)=n_1+1$ درغیراینصورت $outpos(a) \in D$.

حالت نهم: $outpos(a) = f(a,n)$

مثال ۱: رشته ورودی یک ترتیب تصادفی از اعداد مضرب k کوچکتر یا مساوی nk است و رشته خروجی مرتب شده این اعداد بصورت نزولی می باشد یعنی $f(a,n) = n - a/k + 1$. این مثال از تلفیق دو حالت دوم و چهارم حاصل شده است [10].

مثال۲: رشته ورودی یک ترتیب تصادفی از اعداد مضرب k کوچکتر یا مساوی nk است (k عددی است فرد) . رشته خروجی $O=XY$ است که زیر رشته X اعداد زوج بصورت صعودی، و زیر رشته Y اعداد فرد بصورت نزولی می باشد. پس :

$$f(a,n) = \begin{cases} a/2k & \text{اگر } a \text{ زوج باشد} \\ n - (a-k)/2k & \text{اگر } a \text{ فرد باشد} \end{cases}$$

حالت دهم: در این حالت f تابعی است از inpos(a)، اطلاعات ثابت $t_1, t_2, \ldots, t_m$ و a یعنی

$$outpos(a) = f(inpos(a), t_1, t_2, \ldots, t_m, a)$$

مثال: رشته ورودی بطور صعودی مرتب شده است. در این تبدیل رشته خروجی از رشته ورودی با انتقال نماد x به اول رشته حاصل می شود . پس:

$$f(inpos(a), x, a) = \begin{cases} inpos(a)+1 & \text{اگر } a<x \\ 1 & \text{اگر } a=x \\ inpos(a) & \text{اگر } a>x \end{cases}$$

۳- محلی سازی و فرآیند کاهش اطلاعات

یک تبدیل محلی نامیده می شود اگر بتوان رشته ورودی را به بخشهایی که می توانند مستقلاً تبدیل شوند تقسیم نمود. مثلاً در تبدیل نشانگذاری میانوندی به نشانگذاری پسوندی اگر نمادی را که در حالت پسوندی در انتهای رشته قرار می گیرد پیدا کنیم، می توانیم رشته ورودی را به دو زیر رشته تقسیم کنیم بطوریکه مکان خروجی هر نماد در یک زیر رشته مستقل از رشته دیگر باشد. فرض کنید بخش های رشته ورودی $r_1, r_2, \ldots, r_r$ باشند. برای هر نماد a در یک بخش زوج مرتب (i,j) را در نظر بگیرید که i مساوی شماره بخش حاوی نماد و j مساوی inpos(a) باشد.

می گوییم رشته ورودی بصورت ایستا تقسیم بندی شده اگر برای همه بخشهای r_i زوج های مربوطه $(i,i_1),(i,i_2),\ldots,(i,i_k)$ بوده و $i_1, i_2, \ldots, i_k$ رشته ای از اعداد صحیح متوالی باشد. در غیر اینصورت ، رشته ورودی بصورت پویا تقسیم بندی شده است . در تبدیل نشانگذاری میانوندی به نشانگذاری پسوندی قسمتهایی که با دانستن نمادی که به آخر رشته خروجی نگاشت می شود حاصل می شوند. قسمتهای ایستا هستند در حالیکه قسمتهایی که با یافتن موقعیت خروجی عضو محوری در الگوریتم quick sort بدست می آیند قسمتهای پویا هستند.

مفهوم محلی کردن تبدیل وابستگی نزدیکی به آنچه که ممکن است فرآیند کاهش اطلاعات نامیده شود دارد. بازبینی برخی از مثالهای ارائه شده نشان می دهد که دانستن موقعیتهای برخی از نمادهای ورودی این امکان را می دهد که محل سایر نمادها را بدون نیاز به همان اطلاعاتی که در محاسبه نمادهای اولیه بکار رفته است ، محاسبه کنیم. اگر اطلاع از محل خروجی یک یا مجموعه ای از نمادها میزان اطلاعات لازم برای محاسبه مکان خروجی نماد و یا مجموعه ای از نمادها را کاهش دهد می گوییم کاهش اطلاعات رخ داده است. میزان کاهش اطلاعات را می توان بعنوان معیاری برای مقایسه الگوریتمهای مختلف بکار رفته در یک تبدیل خاص بکار برد.

۴. طبقه بندی عمومی

با توجه به اینکه:

الف- $F_3=n$ را می توان در F_4 ادغام نمود. (برای یک رشته ورودی داده شده n جزء اطلاعات ثابت می باشد.)

ب- با داشتن F_6 ، outpos(a) بدون استفاده از هیچ اطلاعات دیگری، حتی آگاهی از صورت مسأله، قابل محاسبه می باشد.

مجموعه انواع اطلاعات F بصورت $F=\{F_1,F_2,F_3,F_4\}$ پیشنهاد می شود که در آن داریم:

$F_1=a$ ، $F_2=inpos(a)$ ، $F_3=t_1,\ldots,t_m$ اطلاعات ثابت ، $F_4=I$

از بین این اطلاعات، F_1 و F_2 اطلاعات اصلی می باشند. زیرا مسائلی وجود دارند که فقط از اطلاعات F_1 یا F_2 استفاده می کنند و ضمناً هر مسأله قابل رسیدن حداقل باید از اطلاعات F_1 یا F_2 استفاده نماید. این توضیح لازم است که اگر مسأله ای فقط از F_3 و F_4 استفاده کند، آنگاه برای یک رشته ورودی تابع f برای همه نمادهای ورودی دقیقاً یک محل را محاسبه می کند. این بدان خاطر است که برای یک رشته ورودی داده شده اطلاعات F_3 و F_4 ثابت می باشند و لذا هیچ متغیری در محاسبه وجود نخواهد داشت.

۵. نتیجه گیری

در مسائل تبدیل رشته اگر بتوان تابع f که مکان هر نماد در رشته ورودی را به مکان متناظر آن در رشته خروجی می نگارد پیدا نمود، آنگاه با در دست داشتن تعداد کافی پردازنده می توان مکانهای نمادها را در خروجی بطور همزمان محاسبه کرد. در این مقاله مسائلی که بتوانند بصورت تبدیل رشته فرموله شوند و مناسب برای پردازش موازی باشند مورد مطالعه قرار گرفت. سپس با نقد و تصحیح طبقه بندی انجام شده در [1] یک طبقه بندی جدید برای این مسائل ارائه گردید. ایده محلی سازی و فرآیند کاهش اطلاعات نیز مورد بررسی قرار گرفت.

مراجع

1. Williams, K. and Meybodi, M. R. "Notes on Parallel Computation for String Transformation Problems", Proceeding of IEEE Conference on Computers and Communications, Phoenix, Arizona, March 1986, pp. 90-94.

2. Flynn, H. J. "Some Computer Organizations and Their Effectiveness", IEEE Transaction on Computers, C-21(a) : 448-460, September 1972.

3. Kuck, D. J. "The Structure of Computers and Communication", Vol. 1, John Wiley and Sons,1978.

4. Kung, H. T. "Synchronized and Asynchronous Parallel Algorithms for Multiprocessors", Algorithms and Complexity : New Directions and Recent Results, J. F. Traub, ed., Academic Press, 1976, pp. 153-200.

5. Quinn, M. J. "Designing Efficient Algorithms for Parallel Computers", McGraw-Hill, 1987.

6. Meybodi, M. R. "On Writing Algorithms for Parallel Computers", WMU Technical Report, 1984.

7. Meybodi, M. R. and Williams, K. "Parallel Processors Applied to String Transformation : An Extended Abstract", Proceeding of ACM Computer Science Conference, Cincinnati, Ohio, February 1986, pp. 441.

8. Williams, K. and Meybodi, M. R. "Representing Problems as String Transformations", ACM Special Interest Group News on Automata and Computability Theory, Vol. 18, No. 3, 1987, pp. 24-30.

9. Meybodi, M. R. and Williams, K. "On Developing Algorithms to Solve Certain Class of Problems Using Parallel Computers", Proceeding of the Seventeenth Annual Conference on Modeling and Simmulations, Pittsburg, Pennsylvania, April 1986, pp. 853-857.

10. Meybodi, M. R. and Mousavi, S. R. "String Transformation Using Parallel Processing", Technical Report, Computer Engineering Department, Amir Kabir University, 1994.

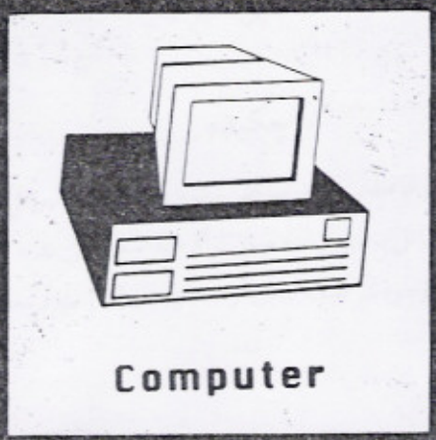
11. Muller, D. E. and Preparata, F. P. "Bounds to Complexities of Networks for Sorting and for Switching", Journal of ACM, 1975, pp. 195-201.

Proceedings

Third Iranian Conference on Electrical Engineering

May 15 - 18 , 1995

(ICEE - 95)



IUST

Iran University of Science & Technology

(IUST)