نهمین کنفرانس سالانه انجمن کامپیوتر ایران
دانشگاه صنعتی شریف
۲۸-۳۰ بهمن ماه ۱۳۸۲

الگوریتمهای مرتب سازی جدید برای اتوماتای سلولی دو بعدی

مهدی شاه آبادی محمد رضا میبدی
آزمایشگاه سیستمهای نرم افزاری
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران

چکیده

اگر چه الگوریتم های متعددی برای مرتب سازی در کامپیوتر های ترتیبی و موازی ارائه شده است ولی هنوزکار زیادی بر روی مرتب سازی برای اتوماتای سلولی انجام نگرفته است . دو الگوریتم یکی منصوب به گوردیلو[1] و لونا[2] و دیگری منصوب به شاه آبادی و میبدی برای اتوماتای سلولی یک بعدی ارایه شده است. الگوریتم گوردیلو و لونا، n عنصر را با استفاده از n سلول در 2n-3 مرحله و الگوریتم شاه آبادی و میبدی n عنصر را با استفاده از n سلول در n-1 مرحله مر تب می نماید. تنها الگوریتم مرتب سازی برای اتوماتای سلولی دو بعدی nxn توسط گلزاری و میبدی ارایه شده است که دارای پیچیدگی زمانی $O(n^2)$ میباشد. در این مقاله دو الگوریتم مرتب سازی موازی جدید برای اتوماتای سلولی دو بعدی ارائه شده است. الگوریتمهای ارائه شده دارای پیچیدگی متوسط $O(\sqrt{n})$ می باشند که نسبت به الگوریتم گلزاری و میبدی دارای مرتبه بزرگی بمراتب کمتری میباشد.
کلمات کلیدی : آتوماتای سلولی، مرتب سازی، پردازش موازی

۱- مقدمه

مرتب سازی داده ها یکه از عملیات مهم در بسیاری از الگوریتمها برای حل مسایل میباشد وبه همین دلیل الگوریتمهای متعددی برای ان طراحی شده است. الگوریتمهای مرتب سازی به دو گروه ترتیبی و موازی تقسیم بندی میشود. الگوریتمهای موازی متعددی الگوریتم مانند مرتب کننده زوج و فرد و الگوریتم مرتب کننده Bitonic که هر دو الگوریتم توسط Batcher[2] ارائه شده، الگوریتم Orcut، الگوریتمThompson و Kung، الگوریتم Kummar و Hirschberg و HyperQuicksort برای مدلهای محاسباتی موازی مختلف گزارش شده است[3] [9] [6] [7][8] [4] [5][11]. الگوریتمهای مرتب سازی برای آرایه های تپنده[3] نیز پیشنهاد شده است[4].

---
[1] Gordillo
[2] Batcher Odd-Even Merg Algorithm
[3] Systolic Arrays

گرچه الگوریتم های متعددی برای مرتب سازی در کامپیوتر های ترتیبی و موازی ارائه شده است ولی هنوزکار زیادی بر روی مساله مرتب سازی برای مدل اتوماتای سلولی انجام نگرفته است . تا کنون دو الگوریتم یکی منصوب به گوردیلو و لونا [1]و دیگری منصوب به شاه آبادی و میبدی برای اتوماتای سلولی یک بعدی ارایه شده است. الگوریتم گوردیلو و لونا n عنصر را با استفاده از n سلول در 2n-3 مرحله و الگوریتم شاه آبادی و میبدی n عنصر را با استفاده از n سلول در n-1 مرحله مر تب می نماید. تنها الکوریتم مرتب سازی برای اتوماتای سلولی دو بعدی nxn توسط گلزاری و میبدی ارایه شده است که دارای پیچیدگی زمانی $O(n^2)$ میباشد[2]. این مقاله یک الگوریتم مرتب سازی موازی جدید برای اتوماتای سلولی دو بعدی ارائه شده است. الگوریتم ارائه شده دارای پیچیدگی متوسط $O(\sqrt{n})$ می باشد. در الگوریتمهای ارائه شده در این مقاله از تمام ظرفیت سلولها استفاده شده یعنی در شروع کار الگوریتمها تمام سلولها در صورت لزوم می توانند عمل جابجایی را انجام دهند، ولی الگوریتم گلزاری و میبدی اینگونه نیست. همچنین در الگوریتم گلزاری و میبدی می بایست بزرگترین و کوچکترین اعداد در لیست اعداد را در سلولهای اضافی که در دورتادور(پیرامون) آرایه دو بعدی قرار گرفته اند قرار داد. این عمل باعث افزایش تعداد سلولها موردنیاز میگردد. ولی در الگوریتم ارائه شده در این مقاله از ارتباطات Wraparound استفاده شده که علاوه بر سرعت بخشیدن به کار مرتب سازی نیازی به 4n سلول اضافی که در الگوریتم گلزاری و میبدی استفاده شده است نمیباشد.

در ادامه مقاله در قسمت ۲ شرح مختصری در باره آتوماتای سلولی داده می شود. در قسمت ۳ الگوریتمهای پیشنهادی برای مرتب سازی اعداد در آتوماتای سلولی دو بعدی ارائه و مراحل اجرای آنها بیان میگردد. در قسمت ۴ نتایج آزمایشها و مقایسه الگوریتمهای پیشنهادی با دیگر الگوریتمهای موجود میباشد.

۲- آتوماتای سلولی

آتوماتای سلولی مدل ریاضی برای سیستم هایی است که در آنها چندین مؤلفه ساده برای الگوهای پیچیده با هم همکاری می کنند. اتوماتای سلولی از یک شبکه منظم سلولها تشکیل شده است که هر سلول می تواند (K>1) مقدار مختلف به خود بگیرد. سلولهای اتوماتای سلولی در زمانهای گسسته بطور همزمان و برطبق یک قانون محلی بنام Φ بهنگام می شوند که در آن مقدار هر سلول براساس مقادیر سلولهای همسایه تعیین می گردد. اتوماتای سلولی براساس معیارهای مورد بررسی به دسته های مختلف تقسیم می گردد. بعنوان نمونه براساس معیار بعد شبکه، اتوماتای سلولی به اتوماتای سلولی یک بعدی، دوبعدی و غیره تقسیم می گردد و براساس مقدار k به اتوماتای سلولی دودوئی (k=2) و اتوماتای سلولی چند مقداره (k>2) تقسیم می شود. همچنین اتوماتای سلولی را براساس شبکه همسایه ها می توان به دودسته اتوماتای سلولی با مرز پریودیک و اتوماتای سلولی با مرز غیر پریودیک تقسیم نمود

در این مقاله دو الگوریتم مرتب سازی جدید برای اتوماتای سلولی دوبعدی ارایه میگردد. برای حل مسائل محاسباتی مانند مرتب سازی به یک ساختمان داده نیاز می باشد. این ساختمان داده شامل ورودی و خروجی و یک رویه برای تبدیل ورودی به خروجی می باشد. مراحل پردازش و تبدیل ورودی به خروجی در اتوماتای سلولی توسط انتقال حالتهای اتوماتای سلولی پیاده سازی می شود. هرچندَدر تعریف اتوماتای سلولی استاندارد حافظه منظور نشده است، ولی اگر اتوماتای سلولی خواسته باشد نقش یک ماشین محاسبه گر عمومی را بازی کند اولاً هر سلول نیاز به تعدادی حافظه و قابلیت نوشتن مقادیر خروجی در حافظه را داشته باشد. تعریف ارائه شده در زیر که در این مقاله بعنوان تعریف اتوماتای سلولی در نظر گرفته شده است،اصلاح شده تعریف اتوماتای میلی می باشد[1].

تعریف : هر اتوماتای سلولی را بصورت ۷ تایی به صورت $(Q, d, V, \Sigma, \Delta, \delta, \lambda)$ می باشد که در آن Q مجموعه حالاتی است که هر سلول می تواند اختیار کند، d ابعاد فضای سلولی را مشخص می نماید که به ازای d=1 یک CA یک بعدی و به ازای d=2 یک CA دوبعدی میباشد، $V(x) = \{x+v_0, x+v_1, \ldots, x+v_k\}$ مشخص کننده k+1 همسایه ای می باشد که بصورت مستقیم با سلول در ارتباطند، Σ الفبای ورودی اتوماتای سلولی می باشد، Δ الفبای خروجی اتوماتای سلولی می باشد، δ تابع انتقال است که بفرم

$\delta : (Q \times \Sigma^{n})^{k+1} \rightarrow Q$ می باشد که براساس تابع انتقال، حالت بعدی هر سلول به حالت و مقادیر حافظه ای تمامی همسایگان آن سلول در مرحله فعلی بستگی دارد (n تعداد ثبات های هر سلول است) و λ رابطه مبدل است که زیر مجموعه متناهی از می باشد .این مبدل مقدار هر حافظه سلول را با توجه به حالت و مقادیر حافظه ای همسایگانش مشخص می سازد.

۳- مرتب سازی با جابجایی موازی برروی اتوماتای سلولی دوبعدی

در این قسمت، اولین الگوریتم مرتب سازی در یک اتوماتای سلولی دوبعدی n×n مطرح می شود. در این اتوماتای سلولی سلولها در یک شبکه دو بعدی قرار گرفته اند و نوع همسایگی Moore در نظر گرفته شده،اگر سلول قرار گرفته در سطر i و ستون j را با $C_{i,j}$ نشان دهیم ، در این صورت همسایه های این سلول را بصورت زیر نشان می دهیم همسایه سمت چپ $C_{i,j-1}$ همسایه سمت راست $C_{i,j+1}$ همسایه بالا $C_{i-1,j}$ همسایه پایین $C_{i+1,j}$ همسایه سمت راست بالا $C_{i-1,j+1}$ همسایه سمت چپ بالا $C_{i-1,j-1}$ همسایه سمت راست پایین $C_{i+1,j+1}$ همسایه سمت چپ پایین $C_{i+1,j-1}$. همچنین سلولهای انتهای سمت راست همسایه سلولهایی می باشند که در انتهای سمت چپ قرار دارند و سلولهای ردیف اول همسایه سلولهای ردیف آخر می باشند یعنی همسایگی با wraparound در نظر گرفته شده. همچنین تعداد سطرهاو ستونهای CA دو بعدی بایستی زوج باشد. آزمایشها نشان داده اند هرچه تعداد سطر ستونها به هم نزدیک تر باشد این الگوریتم بهتر عمل می کند مثلا الگوریتم در CA ۴×۴ بهتر از CA ۸×۲ عمل می کند.

در این الگوریتم جابجایی ها بصورت محلی انجام می شود و برای اینکه از تصادم در هنگام مقایسه و جابجایی جلوگیری شود در هر سلول از یک پرچم(S) استفاده شده که می تواند یک یا صفر باشد. در ابتدا این پرچم را در سلولها بصورت یک در میان صفر ویک قرار می دهیم. فرایند مقایسه و جابجایی در دو مرحله انجام می گیرد.

مرحله اول: سلولهایی که پرچمS آنها یک می باشد مقادیرشان را با مقادیر سلولهای سمت راست مقایسه می کنند و سلولهایی که پرچم S آنها یک می باشد مقادیرشان را با مقادیر سلولهای سمت چپ خود مقایسه می کنند و در صورت نیاز به جابجایی، جابجایی صورت می گیرد. البته در اتوماتای سلولی امکان جابجایی مستقیم داده وجود ندارد ولیکن این امر را می توان توسط قانون قوانین اتوماتای سلولی که در ادامه مقاله بشرح آن خواهیم پرداخت.

مرحله دوم: سلولهایی که پرچمS آنها یک می باشد مقادیرشان را با مقادیر سلولهایی که در سمت راست و پایین قرار دارند مقایسه می کنند و سلولهایی که پرچم S آنها صفر می باشد مقادیرشان را با مقادیر به جابجایی، جابجا میشوند. مقادیر سلولهایی که در سمت چپ بالا قرار دارند مقایسه می شوند و در صورت نیاز جابجایی صورت میگیرد. نهایتا مقدار پرچم S در صورتی که یک باشد صفر می شود و در صورتی که صفر باشد یک میگردد. این قانون برای تمام سلولها بصورت همزمان انجام می شود. البته قوانین برای سلولهای مرزی با سلولهای غیر مرزی اندکی متفاوت است که بعداً در باره آن بیشتر توضیح داده می شود. بعد از اینکه الگوریتم خاتمه یافت برای اینکه لیست مرتب شده حاصل شود سلول ها بایستی طبق شماره گذاری شکل ۱ واکشی شود.

اساس کار الگوریتم بر جابجایی های محلی استوار است یعنی هر سلول مقدار خود را فقط با مقدار سلولهای همسایه اش مبادله کند. به منظور انتخاب همسایه مناسب و جلوگیری از تصادم، هر سلول نیاز به یک پرچم دارد همچنین برای نگهداری مقادیر عددی هرسلول نیاز به یک حافظه می باشد. در ابتدای کار اعدادی که باید مرتب شوند در سلولهای اتوماتای سلولی قرار می گیرند و مقدار اولیه پرچمها برای اتوماتای سلولی بصورت یک در میان صفر و یک مطابق اشکال ۲-الف ویا ۲-ب در نظر گرفته می شود.

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

شکل ۱: اعداد مرتب شده بصورت صعودی در CA دوبعدی

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

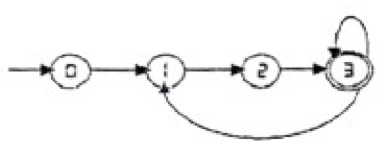
0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

شکل ۲: ترکیب اولیه پرچم های S برای CA دوبعدی

در هر مرحله از اجرای الگوریتم سلولهایی که دارای پرچم S=1 هستند مقادیرشان را با مقادیر سلولهای سمت راست و سمت راست پایین خود مقایسه کرده و همزمان با آنها سلولهایی که دارای پرچم S=0 هستند مقادیرشان را بامقادیر سلولهای سمت چپ و سمت چپ بالای خود مقایسه می کنند ودر صورت لزوم مقادیرشان را با هم جابجا می کنند. آزمایشها همانطور که در ادامه این مقاله به آن اشاره میشود نشان داده اند که حداکثر مراحل مورد نیاز (بدترین حالت) $\sqrt{n} \times 4$ میباشد. بدیهی است که در اکثر موارد زمان اجرای الگوریتم کمتر از بدترین حالت بوده و داده ها زودتر از آن مرتب می شوند. سلولهایی که در یک مرحله عمل جابجایی انجام داده اند، سلولهای همسایه خود را اگرچه در مرحله قبلی عمل جابجایی انجام نداده باشند وادار به عمل مقایسه و جابجایی میکنند و بدین ترتیب موجی ایجاد می شود که با توجه به شرایط ممکن است سرتاسر اتوماتای سلولی را طی کند و سلولها را به ادامه کار مجبور کند. همانطور که بالا ذکر شد در صورتی که در هیچ سلولی عمل جابجایی صورت نگیرد الگوریتم خاتمه میابد. برای پیاده سازی این روش احتیاج به اضافه کردن عناصر حافظه ای به هر سلول داریم که بعداً به شرح آن خواهیم پرداخت.

مراحل اولین الگوریتم پیشنهادی

هر سلول دارای ۴ حالت $\{q_0, q_1, q_2, q_3\}$ میباشد که انتقال بین این حالات طبق شکل ۳ انجام میگیرد. درهنگام شروع بکار الگوریتم، تمامی سلولهادر حالت q_0 بوده و سپس مطابق گامهای زیر عمل می کنند.



شکل ۳: نحوه انتقال بین حالات سلول

مرحله اول:(q_0): این مرحله، مرحله آغازین هر سلول بشمار می آید. در این مرحله مقادیر شمارنده Counter وپرچم Continue و OldContinue درهر سلول را برابر صفر قرار می دهیم. همچنین مقادیر پرچم S هر سلول بصورت شکل ۲ مقداردهی می کنیم.

مرحله دوم: (q_1): در این مرحله متغییر های x, Continue, Counter هر سلول بصورت زیر تعیین می شوند.

سلولهایی که دارای پرچم S=1 هستند بجز سلولهایی که در ستون آخر قرار دارند، مقدارxشان را با مقدار x سلولهای سمت راست خود مقایسه می کنند. در صورتی که مقدارx این سلولها از مقادیر xسلولها سمت راست خود بزرگتر باشد، بایستی مقدار xشان را با مقدار x همسایه سمت راستشان جایگزین کنند.

$$x_{i,j} = \begin{cases} x_{i,j+1} & if(S_{i,j} = 1 \wedge x_{i,j} > x_{i,j+1}) \\ x_{i,j} & otherwise \end{cases}$$

همچنین سلولهایی که دارای پرچم S=0 هستند (بجز سلولهایی که در ستون اول قرار گرفته اند) بایستی مقدارXشان را با مقادیر X سلولهای سمت چپ خود مقایسه کنند. در صورتی که مقدارX این سلولها کوچکتر از مقدارX سلولهای سمت چپ خود باشند مقدار Xشان را با مقدار X همسایه سمت چپ خود جایگزین میکنند.

$$x_{i,j} = \begin{Bmatrix} x_{i,j-1} & if\,(S_{i,j} = 0 \wedge x_{i,j} < x_{i,j-1}) \\ x_{i,j} & otherwise \end{Bmatrix}$$

سلولهایی که دارای پرچم S=1 هستند و در ستون آخر قرار دارند، مقدارXشان را با مقدار X سلولهای متناظرخود که در ستون اول قرار دارند مقایسه می کنند در صورتی که مقدار X این سلولها از X سلولها سمت راست خود کوچکتر باشد، مقدارXشان را با مقدار X همسایه سمت راستشان جایگزین میکنند.

$$x_{i,j} = \begin{Bmatrix} x_{i,j+1} & if\,(S_{i,j} = 1 \wedge x_{i,j} < x_{i,j+1} \\ x_{i,j} & otherwise \end{Bmatrix}$$

همچنین سلولهایی که دارای پرچم S=0 هستند و در ستون اول قرار گرفته اند مقدارX شان را با مقادیر X سلولهای متناظر خود که در ستون آخر قرار گرفته اند مقایسه کنند. در صورتی که مقدار X این سلولها بزرگتر از مقادیرX سلولهای سمت چپ خود باشد مقدار X همسایه سمت چپ، جایگزین مقدارX شان میکنند.

$$x_{i,j} = \begin{Bmatrix} x_{i,j-1} & if\,(S_{i,j} = 0 \wedge x_{i,j} > x_{i,j-1} \\ x_{i,j} & otherwise \end{Bmatrix}$$

مقدار پرچم Continue نیز بصورت زیر تعیین می شود.

$$Continue_{\,i,j} = \begin{cases} 1 & \begin{aligned} &if\,((j \neq n \wedge S_{i,j} = 1 \wedge x_{i,j} \succ x_{i,j+1}) \vee \\ &(j \neq 1 \wedge S_{i,j} = 0 \wedge x_{i,j} \prec x_{i,j-1}) \vee \\ &(j = n \wedge S_{i,j} = 1 \wedge x_{i,j} \prec x_{i,j+1}) \vee \\ &(j = 0 \wedge S_{i,j} = 0 \wedge x_{i,j} \succ x_{i,j-1})) \end{aligned} \\ Continue_{\,i,j} & Otherwise \end{cases}$$

یعنی در این مرحله اگر جابجایی صورت گرفته باشد مقدار متغیر Continue یک می شود و در غیر اینصورت تغییری در مقدار این پرچم داده نمی شود. همچنین بایستی برای تمام سلولها، مقدار شمارنده Counter هم یکی افزایش پیدا کند.

$$Counter_{i,j} = Counter_{i,j} + 1$$

مرحله سوم($q2$):در این مرحله سلولهایی که دارای پرچم S=1 هستند بجز سلولهایی که در سطر آخر قرار دارند، مقدارXشان را با مقادیر Xسلولهای سمت راست پایین خود مقایسه می کنند. در صورتی که مقدارX این سلولها از X سلولها سمت راست پایین خود بزرگتر باشند بایستی مقدارX شان با مقدارX همسایه سمت راست پایین خود جایگزین شود.

$$x_{i,j} = \begin{Bmatrix} x_{i+1,j+1} & if\,(S_{i,j} = 1 \wedge x_{i,j} > x_{i+1,j+1} \\ x_{i,j} & otherwise \end{Bmatrix}$$

همچنین سلولهایی که دارای پرچم S=0 هستند بجز سلولهایی که در سطر اول قرار گرفته اند مقدارxشان را با مقادیر x سلولهای سمت چپ بالای خود مقایسه می کنند. در صورتی که مقدارx این سلولها کوچکتر ازمقدار x سلولهای سمت چپ بالای خود باشد مقدارx شان با مقدار x همسایه سمت چپ بالای خود جایگزین می شود.

$$x_{i,j} = \begin{cases} x_{i-1,j-1} & if\,(S_{i,j} = 0 \wedge x_{i,j} < x_{i-1,j-1}) \\ x_{i,j} & otherwise \end{cases}$$

سلولهایی که دارای پرچم S=1 هستند و در سطر آخر قرار دارند، مقدارxشان را با xسلولهای سمت راست پایین خود مقایسه می کنند و در صورتی که مقدارx این سلولها از مقادیر x سلولها سمت راست پایین خود کوچکتر باشد بایستی مقدارxشان با مقدار x همسایه سمت راست پایینشان جایگزین شود.

$$x_{i,j} = \begin{cases} x_{i+1,j+1} & if\,(S_{i,j} = 1 \wedge x_{i,j} < x_{i+1,j+1}) \\ x_{i,j} & otherwise \end{cases}$$

سلولهایی که دارای پرچم S=0 هستند و در سطر اول قرار گرفته اند مقدارxشان را با xسلولهای سمت چپ بالای خود مقایسه می کنند و در صورتی که مقدارx این سلولها بزرگتر ازمقادیر x سلولهای سمت چپ بالای خود باشد مقدارx شان با مقدار همسایه سمت چپ بالایشان جایگزین می شود.

$$x_{i,j} = \begin{cases} x_{i-1,j-1} & if\,(S_{i,j} = 0 \wedge x_{i,j} \succ x_{i-1,j-1}) \\ x_{i,j} & otherwise \end{cases}$$

مقدار پرچم Continue بصورت زیر تعیین می شود:

$$Continue_{i,j} = \begin{cases} 1 & \begin{array}{l} if\,((i \neq n \wedge S_{i,j} = 1 \wedge x_{i,j} \succ x_{i+1,j+1}) \\ \vee (i \neq 1 \wedge S_{i,j} = 0 \wedge x_{i,j} \prec x_{i-1,j-1}) \\ (i = n \wedge S_{i,j} = 1 \wedge x_{i,j} \prec x_{i+1,j+1}) \\ \vee (i = 1 \wedge S_{i,j} = 0 \wedge x_{i,j} \succ x_{i-1,j-1})) \end{array} \\ Continue_{i,j} & Otherwise \end{cases}$$

یعنی در این مرحله اگر جابجایی صورت گرفته باشد مقدار متغیر Continue یک می شود و در غیر اینصورت تغییری در مقدار این پرچم داده نمی شود.

مقدار شمارنده Counterیکی افزایش پیدا می کند.

$$Counter_{i,j} = Counter_{i,j} + 1$$

مرحله چهارم ($q3$): این مرحله، مرحله پایانی اتوماتا درهر سلول می باشد.در این مرحله کارهای زیر انجام می شود. در صورتی که Counterبرابر ۲ باشد مقدار آن را برابر صفر قرارمی دهیم و در غیر اینصورت مقدار آن را یک واحد افزایش می دهیم. در این مرحله در صورتی که Counter برابر صفر باشد مقدار پرچم S در صورتی که صفر باشد یک و درصورتی که یک باشد مقدار آن صفر میشود. همچنین در صورتی که مقدار پرچم خود این سلول و یا یکی از هشت همسایه سلول برابر یک باشد مقدار پرچم Continue برابر یک میگردد. سپس براساس Continueو OldContinueتصمیم گرفته می شود که آیا به مرحله دوم برویم و یا در همین مرحله باقی بمانیم. درصورتی که یکی از مقادیر پرچمهایContinueو یا OldContinueبرابر یک باشند بدین معنی است که کار مقایسه و جابجایی باید ادامه پیدا کند و در اینصورت باید به مرحله دوم برویم و گرنه باید در همین مرحله پایانی باقی بمانیم.

$$Counter = \begin{cases} 0 & if\, Counter = 2 \\ Counter + 1 & Otherwise \end{cases}$$

$$OldContinue = \begin{cases} Continue & if\, Counter = 0 \\ OldContinue & Otherwise \end{cases}$$

$$S_{i,j} = \begin{cases} 0 & if\,(S_{i,j} = 1 \wedge Counter\ = 0) \\ S_{i,j} & Otherwise \\ 1 & if\,(S_{i,j} = 0 \wedge Counter\ = 0) \end{cases}$$

$$C_{i,j} = \begin{cases} C_{i,j} \vee C_{i-1,j} \vee C_{i+1,j} \vee C_{i-1,j-1} \vee C_{i,j-1} \vee C_{i+1,j-1} \vee C_{i-1,j+1} \vee C_{i,j+1} \vee C_{i+1,j+1} & if\, Counter = 0 \\ C_{i,j} & Otherwise \end{cases}$$

```
if(Continue=1)
     then goto Step1
else  goto step 4
```

در اینجا بخاطر محدودیت فضای نوشتاری، در فرمول بجای عبارت $\text{Continue}_{i,j}$ از عبارت $C_{i,j}$ استفاده کرده ایم.

الگوریتم پیشنهادی با تغییرات جزیی میتواند بهبود قابل ملاحظه ای پیدا کند. الگوریتم بدین بصورت تغییر داده شد که در هر مرحله از اجرای الگوریتم سلولهایی که دارای پرچم S=1 هستند مقادیرشان را با سلولهای سمت راست و سمت راست پایین و سمت راست بالای خود مقایسه کرده و همزمان با آنها سلولهایی که دارای پرچم S=0 هستند مقادیرشان را با سلولهای سمت چپ و سمت چپ بالاو سمت چپ پایین خود مقایسه می کنند ودر صورت لزوم مقادیرشان را با هم جابجا می کنند. الگوریتم اول بعلاوه تغییرات بالا الگوریتم دوم پیشنهادی نامیده است. برای اطلاعات بیشتر در باره این تغییر میتوان به [10] مراجعه نمود.

۴- تجزیه و تحلیل الگوریتم های ارائه شده

تفاوت عمده دو الگوریتم ارایه شده در این مقاله در این است که در الگوریتم اول بطور کلی هر سلول می تواند مقدارش را با سلولهای چپ و راست خود و سلول بالا ی سمت راست خود و یا سلول پایین سمت راست خود مقایسه کند یعنی مقایسه سطری در دو جهت و مقایسه ستونی در یک جهت انجام میگیرد ولی در الگوریتم بهبود یافته هر سلول بصورت سطری و ستونی در سه جهت عمل مقایسه را انجام می دهد. دلیل اینکه این دو الگوریتم قادر به مرتب کردن اعداد هستند اینستکه در یک آرایه دو بعدی در صورتی که هر سطرو ستون بصورت صعودی مرتب شده باشند و برای تمام سطرها داشته باشیم که عنصر انتهای سمت راست سطر بالا از عنصر ابتدای سمت چپ سطر پایین کوچکتر باشد آن آرایه دوبعدی بصورت صعودی مرتب شده می باشد.

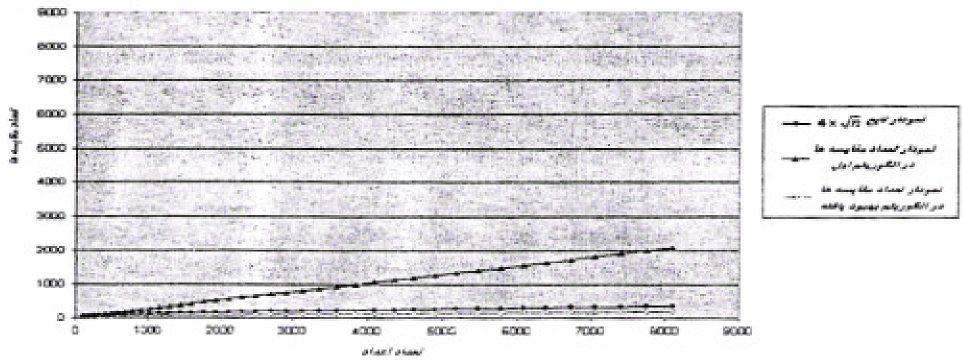
در اولین الگوریتم اگر هر سلول در هر سطر مقدارش را با سلول سمت چپ و سلول سمت راست خود مقایسه ودر صورت لزوم جابجاکند، در اینصورت با این عمل در چند مرحله هر سطر را می توان بصورت صعودی مرتب نمود. همچنین اگر هر سلول مقدارش را با سلول سمت راست پایین و سلول سمت راست خود مقایسه و جابجا نماید در نهایت هر ستون آرایه دو بعدی نیز مرتب می شوند. حال با توجه به اینکه همسایگی برای اتوماتای سلولی با اتصالات Wraparound در نظر گرفته شده است بنابراین سلولهای انتهای سمت راست هر سطر با سلولهای ابتدای سطر پایین خود نیز مقایسه و جابجا می شوند. یعنی بزرگترین عنصر سطر بالایی با کوچکترین عنصر سطر

پایینی عمل مقایسه و جابجایی را انجام می دهد. بنابراین می توان نتیجه گرفت که این الگوریتم می تواند اعداد را بصورت صعودی یا نزولی مرتب نماید. به دلایل مشابه می توان نتیجه گرفت که الگوریتم دوم نیز قادر به مرتب سازی میباشد.

در الگوریتمهای ارائه شده در این مقاله از تمام ظرفیت سلولها استفاده شده یعنی در شروع کار الگوریتم تمام سلولها در صورت لزوم می توانند عمل جابجایی را انجام دهند، ولی الگوریتم گلزاری و میبدی اینگونه نیست. همچنین در الگوریتم گلزاری می بایست بزرگترین و کوچکترین اعداد نسبت به اعداد موردنظر که قرار است مرتب شوند را در سلولهای اضافی که در دورتادور(پیرامون) آرایه دو بعدی قرار گرفته اند قرار داد که باعث افزایش تعداد سلولها موردنیاز میگردد. در الگوریتم ارائه شده در این مقاله بجای اینکار از ارتباطات بین سلولهای مرزی[4] استفاده شده که علاوه بر سرعت بخشیدن به کار مرتب سازی نیازی به $4n$ سلول اضافی که در الگوریتم گلزاری و میبدی استفاده شده است ندارد. در شکل ۴ زمان اجرای الگوریتمهای پیشنهاد شده در این مقاله و تابع $4\times\sqrt{n}$ مقایسه شده اند. مقایسه بین این دو الگوریتم و دو الگوریتم گزارش شده برای اتوماتای سلولی یک بعدی[10][1] در نمودار شکل ۵ نشان داده شده است.

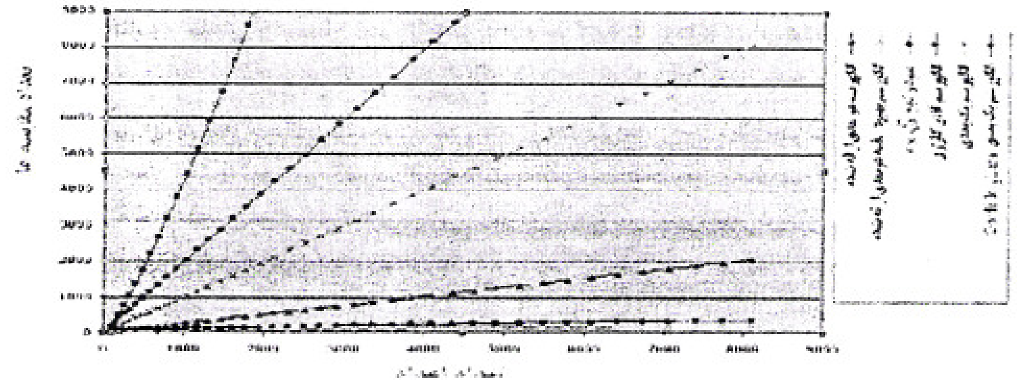
۵- نتیجه گیری

در این مقاله دو الگوریتم مرتب سازی جدید برای اتوماتای سلولی دو بعدی ارایه گردید و با تنها الگوریتم ارایه شده برای اتوماتای سلولی دو بعدی منصوب به گلزاری و میبدی مقایسه شدند. این دو الگوریتم جدید n^2 عنصر را با استفاده از n^2 سلول در $O(\sqrt{n})$ مرحله مرتب می نماید و در نتیجه دارای مرتبه بزرگی بمراتب کمتری در مقایسه با الگوریتم گزارش شده توسط گلزاری و میبدی میباشد.



شکل ۴: نمودار نمودار مقایسه الگوریتم ارائه شده دو بعدی و بهبود داده شده دوبعدی

4 Wraparround Connection



شکل ۵: مقایسه الگوریتم های پیشنهادی با سایر الگوریتم ها مرتب سازی اعداد توسط اتوماتای سلولی

مراجع

[1] L. Gordillo and V. Luna, *"Parallel Sort on a Linear array of Cellular Automata"*, IEEE Trans. Comput, vol. 2, pp. 1904-1910, 1994.
[2] Sh. Golzari and M. R. Meybodi, *"Sorting Algorithms for Two Dimensional Cellular Automata"*, Computer Engineering Dept. Technical Report, Amirkabir University of Technology, 2000.
[3] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*, Addison-Wesley, 1994.
[4] M. Mitchel , P. T. Hraber ,and J. P. Crutchfield, *"The Evolution of Emergent Computation"* , Proceedings of the National Academy of Sciences, USA, Vol. 92, No. 23, 1995.
[5] W. Burks, *Essays On Cellular Automata*, Urbana, IL :University of Illinois Press, 1970
[6] R .K. Squier, K. Steiglitz,and M. H. Jakubowski, *"General Parallel Computation Without CPUs: VLSI Realization of Practical Model"*, Tech.Rep. TR-484-95, Computer Science Department, Princeton University, Princeton, NJ,1995.
[7] D. Thompson, and H. T. Kung, *"Sorting on a Mesh Connected Parallel Computer,"* Communication ACM, Vol, 20, 1977 ,pp. 263-271.
[8] K. E. Batcher, *"Sorting Network and Their Applications,"* AFIP Proc, Vol.32, 1968,pp.307-314.
[9] S. Orcutt, *"Computer Organization and Algorithnes for very High Speed Computations"*, Ph, D. Thesis, Stanford University, 1984.
[10] M. Shahabadi and M. R. Meybodi, *" Sorting Algorithms for Two Dimensional Cellular Automata"*, Technical Report, Computer Engineering and Information Department
[11] D.E. Knuth, *The Art of Computer Programming: Sorting and Searching*.Addison Wesley, 1973.

انجمن کامپیوتر ایران
Computer Society of Iran

مجموعه مقالات

(جلداول)

(مشتمل بر مقالات فارسی)

۲۸ تا ۳۰ بهمن ماه ۱۳۸۲
دانشگاه صنعتی شریف

نهمین
کنفرانس سالانه
انجمن کامپیوتر ایران

ITRC

مرکز فناوری اطلاعات و ارتباطات پیشرفته