یک الگوریتم مرتب سازی برای اتوماتای سلولی یک بعدی

مهدی شاه‌آبادی   محمد رضا میبدی

دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، آزمایشگاه سیستمهای نرم افزاری

E-mail:Shahabadi@gmail.com, mmeybodi@aut.ac.ir

چکیده – مرتب سازی داده ها یکی از مهمترین مسایل در علم کامپیوتر میباشد و بهمین دلیل الگوریتم های متعددی برای آن برای ماشین های مختلف ارائه شده است. برای مرتب سازی در اتوماتای سلولی یک بعدی کار زیادی انجام نگرفته است. تنها الگوریتم ارایه شده برای اتوماتای سلولی یک بعدی توسط گوردیلو و لونا طراحی شده است. این الگوریتم *n* عنصر را با استفاده از *n* سلول در *2n-3* مرحله مر تب می نماید. در این مقاله یک الگوریتم برای مرتب کردن *n* عدد برای اتوماتای سلولی یک بعدی پیشنهاد میگردد که *n* عنصر را در *n-1* مرحله مرتب مینماید و در نتیجه دارای سرعت اجرای حدودا دو برابر در مقایسه با الگوریتم گوردیلو و لونا میباشد.

کلید واژه: آتوماتای سلولی، مرتب سازی، پردازش موازی

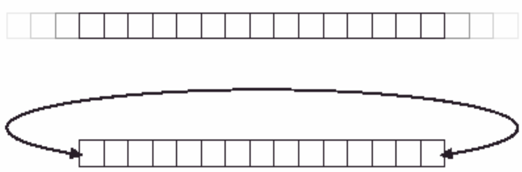
۱– مقدمه

مرتب سازی داده ها نقش مهمی در حل خیلی از مسائل در زمینه های مختلف از قبیل مسایل گرافها، هندسه محاسباتی و پردازش تصاویر بازی میکند و به همین دلیل الگوریتمهای متعددی برای آن طراحی شده است. الگوریتمهای مرتب سازی به دو گروه ترتیبی و موازی تقسیم بندی میشود. الگوریتمهای موازی متعددی برای مدلهای محاسباتی موازی مختلف گزارش شده است[12] [33-36] [34]. الگوریتمهای مرتب سازی برای آرایه های تپنده نیز پیشنهاد شده است[25]. در باره الگوریتم های مرتب سازی برای اتوماتای سلولی کار زیادی انجام نگرفته است. تنها الگوریتم گزارش شده برای اتوماتای سلولی یک بعدی توسط گوردیلو و لونا [6] و برای اتوماتای سلولی دو بعدی توسط گلزاری و میبدی[۳۷] میباشد. الگوریتم گوردیلو و لونا n عنصر را با استفاده از n سلول در 2n-3 مرحله مرتب می نماید. الگوریتم گلزاری و میبدی $n^۲$ عنصر را در یک اتوماتای سلولی دو بعدی nxn در زمان $O(n^2)$ مرتب میکند. در این مقاله الگوریتم جدیدی برای مرتب کردن n عدد برای اتوماتای سلولی یک بعدی ارایه میگردد. این الگوریتم n عدد را در n-1 مرحله مرتب مینماید و در نتیجه دارای سرعت اجرای حدودا دو برابر نسبت به الگوریتم گوردیلو و لونا میباشد. الگوریتم دیگری نیز توسط گلزاری و میبدی برای اتوماتای سلولی دو بعدی nxn طراحی شده است که دارای پیچیدگی زمانی $O(n^2)$ میباشد.

ادامه مقاله بدین صورت سازماندهی شده است. دربخش ۲ شرح مختصری بر آتوماتای سلولی داده میشود. در بخش ۳، ابتدا الگوریتم گوردیلو و لونا شرح داده میشود و سپس یک الگوریتم جدید مرتب سازی برای آتوماتای سلولی یک بعدی ارائه می شود. بخش ۴ اختصاص به مقایسه کار انجام شده در این مقاله با کار قبلی دارد. بخش نهایی نتیجه گیری می باشد.

۲– اتوماتای 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 قانون محلی Φ بهنگام می شوند که در آن مقدار هر سلول براساس مقادیر سلولهای همسایه ان سلول تعیین می گردد. اتوماتای سلولی یک بعدی با مرز پریودیک و اتوماتای سلولی با مرز غیر پریودیک در شکل ۱ نشان داده شده است.



شکل ۱: اتوماتای سلولی یک بعدی با مرز غیر پریودیک و اتوماتای سلولی یک بعدی با مرز پریودیک

در اتوماتای سلولی یک بعدی مقدار سلول i(برای $1 \le i \le n$) در زمان t که با $a_i(t)$ نشان داده میشود بصورت زیر محاسبه میگردد[1].

$$a_i(t+1) = \Phi[a_{i-1}(t), a_i(t), a_{i+1}(t)] \qquad (1)$$

در رابطه بالا، اگرقانون Φ فقط به مقدار همسایه ها بستگی داشته باشد آنراقانون general مینامند و اگر قانون Φ تابعی از مجموع مقادیر سلولهای همسایه و سلول مرکزی باشد آنرا قانون totalistic میگویند و بصورت زیر بیان میشود[2]

$$a_i(t+1) = \Phi[a_{i-1}(t) + a_i(t) + a_{i+1}(t)] \qquad (2)$$

در صورتی که قانون Φ تابعی از مجموع مقادیر سلولهای همسایه و مقدار سلول مرکزی باشد آنرا قانون Outer totalistic می گویند و بصورت زیر نشان داده می شود[3].

$$a_i(t+1) = \Phi[a_i(t), a_{i-1}(t) + a_{i+1}(t)] \qquad (3)$$

همچنین قانون Φ میتواند بصورت قطعی یا احتمالی باشد. در صورتیکه Φ یک تابع تصادفی باشد آنرا قانون احتمالی و در غیر اینصورت آنرا قانون قطعی میگویند.

در این مقاله به اتوماتای سلولی از دید یک ماشین محاسباتی عمومی نگریسته شده و یک الگوریتم برای حل مسله مرتب سازی توسط برای ان ارائه میگردد. برای حل مسائل محاسباتی 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 حالات در اتوماتای سلولی صورت میگیرد. هرچند در تعریف اتوماتای سلولی استاندارد حافظه منظور نشده است، ولی اگر اتوماتای‌سلولی خواسته باشد نقش یک ماشین محاسبه‌گر عمومی را بازی کند اولاً هر سلول بایستی مجهز به حافظه باشد و همچنین بایستی قابلیت نوشتن مقادیر خروجی در حافظه را داشته باشد. تعریف زیر بعنوان تعریف اتوماتای سلولی در نظر گرفته شده است[6] :

اتوماتای سلولی: یک اتوماتای سلولی بصورت ۷ تایی $(Q, d, V, \Sigma, \Delta, \delta, \lambda)$ نشان داد میشود که

- Q مجموعه حالاتی است که هر سلول می تواند اختیار کند.
- d بعد فضای سلولی را مشخص می نماید.
- برای هر سلولx در اتوماتای سلولی آرایه $V(x) = \{x+v_0, x+v_1, \dots, x+v_k\}$ مشخص کننده $k+1$ همسایه ای می باشد که بطور مستقیم با سلول در ارتباطند.
- Σ الفبای ورودی اتوماتای سلولی می باشد.
- Δ الفبای خروجی اتوماتای سلولی می باشد.
- $\delta : (Q \times \Sigma^n)^{k+1} \rightarrow Q$ تابع انتقال می باشد. حالت بعدی هر سلول به حالت و مقادیر حافظه همسایگان آن سلول در مرحله فعلی بستگی دارد. n تعداد رجیستر های هر سلول است.
- λ رابطه مبدل است که زیر مجموعه متناهی از $(Q \times \Sigma^n)^{k+1} \times \Delta^n$ می باشد. این مبدل مقدار حافظه هر سلول را با توجه به حالت و مقادیر حافظه های همسایگانش مشخص می سازد.

۳- مرتب سازی برروی اتوماتای سلولی یک بعدی

در این قسمت ابتدا به شرح الگوریتم گوردیلو و لونا که در سال ۱۹۹۴ پیشنهاد شده است میپردازیم و سپس الگوریتم پیشنهادی ارایه میگردد.

۳-۱- ۱-۱- الگوریتم مرتب سازی گوردیلو و لونا

در این الگوریتم هر سلول دارای دو وضعیت q_0 و q_1 می باشد. در ابتدای الگوریتم تمامی سلولها در حالت q_0 قرار دارند.در انتقال ازq_0 به q_1 هر سلول مقدارش را با مقدار هر دو سلول همسایه اش مقایسه می کند و در صورتی که مقدارش از مقدار همسایه سمت راستش بزرگتر باشد آن را به سمت راست انتقال دهد و در صورتی که مقدارش ازمقدار همسایه سمت چپش کوچکتر باشد آن را به سمت چپ انتقال دهد. بنابراین یک سلول می تواند در یک زمان مقدارش را هم با همسایهٔ راستش و هم با همسایهٔ چپ خود جابجا نماید که در این صورت تصادم پیش می آید. بهمین دلیل فقط سلولی می تواند مقدارش را با همسایهٔ چپی خود جابجا نماید که مقدارش کوچکتر یا مساوی مقدار همسایه سمت راست باشد. در نتیجه هر سلولی که مقدارش کوچکتر یا مساوی مقدار همسایه سمت راستش و کوچکتر از مقدار همسایه سمت چپش باشد مقدار محدودیت سد کننده خود را برابر یک و درصورتی که مقدارش بزرگتر از مقدار همسایه سمت راستش باشد مقدار محدودیت سد کننده راست خود را برابر یک قرار می دهد. مقادیر محدودیت های سد کننده چپ و راست بصورت زیرمحاسبه میگردد.

$$S_i^r = \begin{cases} 1 & x_i > x_{i+1} \\ 0 & otherwise \end{cases} \qquad (4)$$

$$S_0^1 = \begin{cases} 1 & x_i < x_{i-1} \wedge x_i \le x_{i+1} \\ 0 & otherwise \end{cases} \qquad (5)$$

در روابط فوق S_i^r محدودیت سد کننده راست سلول i ام و S_i^l محدودیت سد کننده چپ سلول i ام می باشد.

درانتقال از q_1 به q_0 براساس مقادیر محدودیت های سد کننده چپ و راست تصمیم گیری می شود که جابجایی صورت گیرد یا خیر؟ بدین صورت که اگر مقدار محدودیت سد کننده چپ سلول و مقدار محدودیت سد کنند راست همسایه سمت چپ سلول برابر یک باشند مقدار همسایه چپ جایگزین این سلول می شود و در صورتی که مقدار محدودیت سد کننده راست سلول و مقدارمحدودیت سد کنندهٔ چپِ سلول سمتِ راست برابر یک باشد، مقدار سلول سمت راست جایگزین این سلول می شود و در غیر اینصورت مقدار سلول تغییر نمی کنند این عملیات را می توان با رابطه (6) نشان داد.

$$x_i \begin{cases} x_{i+1} & S_i^r = 1 \wedge S_i^L = 1 \\ x_{i-1} & S_i^L = 1 \wedge S_{i-1}^K = 1 \\ x_i & otherwise \end{cases} \qquad (6)$$

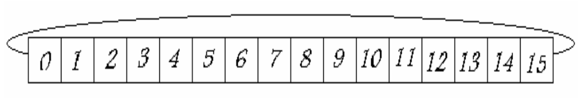
در این رابطه x_i مقدار سلول i ام می باشد. این الگوریتم زمانی خاتمه میابد که چرخه انتقال q_0 به q_1 و q_1 به q_0 $2n-3$ بار تکرار شده باشد. این تعداد تکرار برای مرتب کردن بدترین حالت قرار گرفتن اعداد در اتوماتای سلولی مورد نیاز میباشد. بدترین حالت زمانی رخ می دهد که اعداد برعکس حالتی که می بایست مرتب شوند در آرایه قرار گیرند. این وضعیت در شکل ۳ نشان داده شده است.

تعداد			نمونه مرتب سازی اعداد برای 7 عدد							
	-1	7	6	5	4	3	2	1	#	#
1	-1	7	6	5	4	3	1	2	#	#
2	-1	7	6	5	4	1	3	2	#	#
3	-1	7	6	5	1	4	2	3	#	#
4	-1	7	6	1	5	2	4	3	#	#
5	-1	7	1	6	2	5	3	4	#	#
6	-1	1	7	2	6	3	5	4	#	#
7	-1	1	2	7	3	6	4	5	#	#
8	-1	1	2	3	7	4	6	5	#	#
9	-1	1	2	3	4	7	5	6	#	#
10	-1	1	2	3	4	5	7	6	#	#
11	-1	1	2	3	4	5	6	7	#	#

شکل ۳- عملکرد الگوریتم مرتب سازی موازی برای ۷ عدد

۱-۲- ۳-۲- یک الگوریتم مرتب سازی موازی برای اتوماتای سلولی یک بعدی

در این بخش، یک الگوریتم مرتب سازی جدید برای اتوماتای سلولی یک بعدی با خصوصیت wraparound پیشنهاد میشود. همسایه های سلول C_i سلولهای سمت چپ و سمت راست این سلول، یعنی سلولهای C_{i-1} و C_{i+1} میباشند. سلول انتهای سمت راست همسایه سلولی می باشد که در انتهای سمت چپ قرار دارد تعداد سلولهای اتوماتای‌سلولی زوج در نظر گرفته شده است. در این الگوریتم جابجایی ها بصورت محلی انجام می شود و برای اینکه از تصادم در هنگام مقایسه و جابجایی جلوگیری شود هر سلول مجهز به یک پرچم میباشد که می تواند مقدار یک و یا صفر را اختیار کند. در ابتدای کار این پرچم ها را بصورت یک در میان صفر و یک در سلولها قرار می دهیم. هر مرحله از اجرای الگوریتم شامل دو مرحله است. در مرحله اول سلولهایی که پرچم آنها یک می باشد مقادیرشان را باسلولهای سمت راست مقایسه می کنند و سلولهایی که پرچم آنها یک می باشد مقادیرشان را با سلولهای سمت چپ خود مقایسه می کنند و در صورت نیاز جابجایی صورت می گیرد. لازم به ذکر است که در اتوماتای سلولی امکان جابجایی مستقیم مقادیر بین دو سلول غیر همسایه وجود ندارد. چگونگی انجام این چنین جابجایی از طریق جابجایی بین سلولهای همسایه انجام میگیرد که در ادامه شرح داده خواهد شد. قوانین برای سلولهای مرزی و سلولهای غیر مرزی متفاوت است که بعداً در باره آن بیشتر توضیح داده می شود. زمانیکه الگوریتم خاتمه میابد، مقادیر در سلولهای اتوماتای سلولی یک بعدی بصورت سعودی مرتب میشوند (شکل ۴).



شکل ۴ اعداد مرتب شده بصورت صعودی در CA

الگوریتم پیشنهادی بر اساس جابجایی های محلی میباشد. یعنی هر سلول مقدار خود را می تواند فقط با مقدار سلولهای همسایه اش تعویض کند. به منظور انتخاب همسایه مناسب برای جابجایی و همچنین جلوگیری از تصادم، هر سلول نیاز به یک پرچم دارد که آن را با S نشان میدهیم. همچنین برای نگهداری مقادیر عددی هرسلول نیاز به یک حافظه می باشد که آن را با x نشان می دهیم. در ابتدا اعدادی که می بایست مرتب شوند در متغیر های x سلولهای اتوماتای سلولی قرار داده میشوند و مقدار اولیه پرچمها S برای سلولهای اتوماتای سلولی بصورت یک در میان صفر و یک در نظر گرفته می شود. مقادیر اولیه پرچم های S به یکی از دو صورت شکل ۵-الف و یا شکل ۵-ب بایستی باشد. در هر مرحله از اجرای الگوریتم سلولهایی که دارای پرچم S=1 هستند مقادیرشان را با سلولهای سمت راست خود مقایسه کرده و سلولهایی که دارای پرچم S=0 هستند مقادیرشان را با سلولهای سمت چپ خود مقایسه می کنند ودر صورت لزوم مقادیرشان را با یکدیگر جابجا می کنند.

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

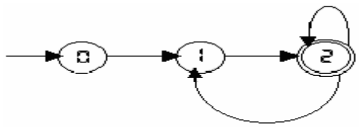
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

شکل ۵-الف وب: ترکیب اولیه پرچم ها برای CA

برای همزمان سازی و حفظ ترتیب پرچم ها بصورت یک درمیان صفر و یک، از متغیر دیگری به نام Counter استفاده میشود. مقدار این متغیر در مرحله اول صفر می باشد. کار این متغیر بگونه ای است که در صورتی که سلولی در مرحله سوم باقی بماند بطریقی عمل می کند که با سلولی که به مرحله دوم می رود هماهنگ باشد. طرز عمل این متغیر را می توان با ردیابی الگوریتم بهتر درک کرد. به منظور پیاده سازی تست خاتمه، برای هر سلول دو پرچم بنامهای Continue وOldContinue در نظر گرفته شده است. در صورتی که هر دو پرچم یک سلول صفر باشند سلول به حالت خاموش می رود و در غیر اینصورت به کار خود ادامه می دهد. طرز عمل این الگوریتم به شکلی است که سلولها همگی با هم خاموش نمیشوند. درهر مرحله مقایسه و جابجایی، اگر سلولی جابجایی انجام دهد، مقدار حافظه Continue خود را برابر یک قرار می دهد و در مرحله تست پایان کار سلول، هر سلول این مقدار را به حافظه Continue دو همسایه دیگر خود منتقل می سازد. این کار بوسیله OR نمودن مقادیر Continue همسایه ها با یکدیگر انجام می شود.

۱-۳- ۳-۳ - مراحل الگوریتم

هر سلول در اتوماتای سلولی دارای سه حالت q_0, q_1, q_2 میباشد. در شروع الگوریتم، تمامی سلولهادر حالت q_0 قرار میگیرند. الگوریتم طبق گامهای زیر عمل می کند.



شکل ۶: نحوه انتقال بین حالات هر سلول

مرحله اول:(q_0**)**: این مرحله، مرحله آغازین هر سلول میباشد. در این مرحله مقادیر Counter، Continue و OldContinue هر سلول برابر صفر قرار داده میشوند و مقادیر پرچمهای S هر سلول بر طبق یکی از الگوهای بیتی شکل ۵ مقدار دهی میشوند.

مرحله دوم: (q_1**)**: در این مرحله هر سلولی که دارای پرچمS یک میباشد مقدار خود را با مقدار سلول سمت راست خود مقایسه می کند و در صورتی که مقدار این سلول از مقدار سلول سمت راست خود بزرگتر باشد مقدارش را با مقدار همسایه سمت راستش جایگزین میکند.

همچنین سلولی که دارای پرچم صفر میباشد مقدارش را با مقدار سلول سمت چپ خود مقایسه می کند و در صورتی که مقدار این سلول کوچکتر از مقدار سلول سمت چپش باشد، مقدارش برابر مقدار همسایه سمت چپش میگردد. در صورتیکه مقدار یک سلول تغییر کند پرچم Continue آن سلول به یک و در غیر اینصورت تغییری پیدا نمی کند. سپس Counter برای تمام سلولها افزایش پیدا میکند.

این مراحل را می توان بصورت قوانین زیر بیان کرد:

برای سلولهای غیرمرزی قانون جایگزاری برای x_i و $\mathrm{Continue_i}$ طبق رابطه(7) و برای سلولهای مرزی قانون جایگزاری برای x_i و $\mathrm{Continue_i}$ طبق رابطه (۸) میباشد

$$x_i = \begin{cases} x_{i+1} & if\ (S_i = 1 \wedge x_i \rangle x_{i+1}) \\ x_i & otherwise \\ x_{i-1} & if\ (S_i = 0 \wedge x_i \langle x_{i-1}) \end{cases} \tag{7}$$

$$\mathrm{Continue_i} = \begin{cases} 1 & \mathrm{if}((S_i = 1 \wedge x_i \succ x_{i+1}) \vee (S_i = 0 \wedge x_i \prec x_{i-1})) \\ \mathrm{Continue_i} & \mathrm{Otherwise} \end{cases}$$

$$x_i = \begin{cases} x_{i+1} & if\ (S_i = 1 \wedge x_i \langle x_{i+1}) \\ x_i & otherwise \\ x_{i-1} & if\ (S_i = 0 \wedge x_i \rangle x_{i-1}) \end{cases} \tag{8}$$

$$Continue_i = \begin{cases} 1 & if((S_i = 1 \wedge x_i \prec x_{i+1}) \vee (S_i = 0 \wedge x_i \succ x_{i-1})) \\ Continue_i & Otherwise \end{cases}$$

مرحله سوم (q_2): این مرحله، مرحله پایانی هر سلول می باشد.

در این مرحله کارهای زیر انجام می شود. در صورتی که Counterبرابر ۱ باشد مقدار آن برابر صفر و در غیر اینصورت مقدار آن را یکی افزایش میابد. در این مرحله در صورتی که Counter برابر صفر باشد پرچم S در صورتی که صفر باشد یک و درصورتی که یک باشد صفر میشود. در ادامه اگر Counter برابر صفر باشد مقدار Continue در OldContinue کپی میشود و در صورتی که مقدار Continueاین سلول و یا یکی از دو همسایه سلول برابر یک باشد مقدار Continue برابر یک قرارمیگیرد. این کار با OR کردن پرچم Continue سلول با پرچم Continue همسایه هایش انجام میگیرد. سپس با توجه به مقادیر Continue و OldContinue تصمیم گرفته می شود که آیا الگوریتم به مرحله دوم و یا در همین مرحله باقی بماند. درصورتی که مقدار یکی از پرچم های Continue و یا OldContinue برابر یک باشد، بدین معنی است که کار مقایسه و جابجایی باید ادامه پیدا کند و الگوریتم بایستی به مرحله دوم برود. اگر مقادیر Continue و OldContinue هر دو برابر صفر باشند الگوریتم بایستی در مرحله پایانی باقی بماند.

$$Counter = \begin{cases} 0 & Counter = 1 \\ 1 & Counter = 0 \end{cases}$$

$$\mathrm{OldContinue_i} = \begin{cases} \mathrm{Continue_i} & \mathrm{Counter} = 0 \\ OldContinue_i & Counter = 1 \end{cases}$$

$$S_i = \begin{cases} 0 & if\ (S_i = 1 \wedge Counter = 0) \\ S_i & Otherwise \\ 1 & if\ (S_i = 0 \wedge Counter = 0) \end{cases}$$

$$Continu_i = \begin{cases} Continu_{i-1} \vee Continu_i \vee Continu_{i+1} & Counte = 0 \\ Continu_i & \mathrm{otherwise}_i \end{cases}$$

f(Continue=1)
then goto Step1
else goto step ۲

۴- تجزیه و تحلیل الگوریتم

بدترین حالت در الگوریتم پیشنهادی و الگوریتم گوردیلو و لونا زمانی رخ میدهد اگر قرار باشد یک لیست نزولی (صعودی) بصورت صعودی(نزولی) مرتب شود. الگوریتم گوردیلو و لونا در بهترین و بدترین حالت و حالت متوسط نیاز به 2n-3 مرحله و الگوریتم پیشنهادی همانطور که در شکل ۷ نشان داده شده است در بدترین حالت نیاز به n-1 مرحله دارد. لازم به ذکر است که در الگوریتم پیشنهادی در شروع کار تمام سلولها در صورت لزوم می توانند عمل جابجایی را انجام دهند ولی در الگوریتم گوردیلو و لونا اینگونه نیست.

10	9	8	7	6	5	4	3	2	1	initialize
9	10	7	8	5	6	3	4	1	2	1
2	7	10	5	8	3	6	1	4	9	2
2	7	5	10	3	8	1	6	4	9	3
2	5	7	3	10	1	8	4	6	9	4
2	5	3	7	1	10	4	8	6	9	5
2	3	5	1	7	4	10	6	8	9	6
2	3	1	5	4	7	6	10	8	9	7
2	1	3	4	5	6	7	8	10	9	8
1	2	3	4	5	6	7	8	9	10	9

شکل۷ : نحوه مرتب سازی داده ها در بدترین حالت برای الگوریتم پیشنهادی

۵- نتیجه گیری

در این مقاله یک الگوریتم مرتب سازی جدید برای اتوماتای سلولی یک بعدی ارایه گردید و با تنها الگوریتم ارایه شده برای اتوماتای سلولی یک بعدی منصوب به گوردیلو و لونا مقایسه شد. الگوریتم پیشنهادی در این مقاله n عنصر را با استفاده از n سلول در n-1 مرحله مرتب می نماید و در نتیجه سرعت اجرای آن حدودا دو برابر الگوریتم گوردیلو و لونا میباشد.

مراجع

[1] S. Wolfram, “Computation Theory of Cellular Automata”, Physica Scripta, Vol. 9,pp. 170-183, 1985.

[2] N. H. Packard and S. Wolfram,"Two-Dimensional Cellular Automata" Journal of Statistical Physics, Vol 38, pp 901-946, 1985.

[3] T. Toffoli and N. Margolus, Cellular Automata Machines: A New Environment for Modeling, MIT Press Series in Scientific Computation, 1987.

[4] P.l Pal Chaudhuri, D. Roy Chowdhury, S. Nandi, and S. Chattopadhyay, “Additive Cellular Automata Theory and Applications”, Vol., IEEE Computer Society Press, 1997.

[5] M. Alfonseca and A. Ortega, ”Representation of Some Cellular Automata by Means of Equivalent L Systems”, Complexity International, Vol. 7, Paper ID:alfons01,2000. URL: http://www.csu.edu.au/ci/vol07/alfons01

[6] L. Gordillo and V. Luna, “Parallel Sort on a Linear Array of Cellular Automata”, IEEE Trans. Comput, Vol. 2, pp. 1904-1910, 1994.

[7] C. Y. R. Chen, C. Y. Hou, and U. Singh, “Optimal Algorithms for Bubble Sort Based Non-Manhattan Channel Routing”, IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems, Vol. 13, No. 5, pp. 603 – 609, May 1994.

[8] K. Chaudhary and P. Robinson, "Channel Routing by Sorting" IEEE Transactions on Computer-Aided Design, Vol. 10, No. 6, pp. 754 – 760, June 1991.

[9] R. Pfeifer and H. kunz, Artificial life, URL:http://www.ifi.unizh.ch/ailab/teaching/AL99.

[10] S. Wolfram, Cellular Automata and Complexity: Collected Papers, Addison-Wesley, 1994.

[11] D. E. Knuth, The Art of Computer Programming: Sorting and Searching. Addison Wesley, 1973.

[12] M. J. Quinn, Parallel Computing: Theory and Practice: McGrow-Hill, Ing, 1994.

[13] P. Sarkar, “Brief History of Cellular Automata”, ACM Computing Survey, Vol. 32, No 1, 2000.

[14] G. Y. Vichniac, P. Tamayo, and H. Hartman. "Annealed and Quenched Inhomogeneous Cellular Automata", J. Statistical Phys., Vol. 45 pp. 875-883, 1986.

[15] S. Wolfram, "Cellular Automata", Los Alamos Science, Vol 9, pp. 2-21, 1983.

[16] M. R. Meybodi, H. Beigi and M. Taherkhani, “Cellular Learning Automata and Its Applications”, Computer Engineering & IT Dept. Technical Report, Amirkabir University of Technology, 2000.

[17] J. Von Neumann, “Theory of Self-Reproducing Automata”, University of Illinois Press, 1966.

[18] M. Taherkhani, “Proposing and Studying of Cellular Learning Automata as a Tool for Modeling Systems”, M.Sc. Thesis, Computer Eng. Dept, Amirkabir University of Technology, Tehran, Iran, 2000.

[19] M. Mitchell, “Computation in Cellular Automata: A Selected Review”, Technical Report, Santa Fe Institute, Santa Fe, USA, 1996.

[20] S. Nandi, B. K. Kar and P. Chaudhuri, “Theory and Applications of Cellular Automata in Cryptography”, Vol. 43, No 12, 1994.

[21] S. Wolfram, “Random Sequence Generation by Cellular Automata” Advances in Applied Mathematics, Vol. 7, pp. 123-169, 1986.

[22] S. Wolfram “Twenty Problems in the Theory of Cellular Automata”, Physica Scripta, Vol. 9, pp. 170-183, 1985.

[23] S. Wolfram, “Universality and Complexity in Cellular Automata”, Physica D, Vol. 10, pp. 1-35, 1984.

[24] 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.

[25] G. M. Megson, An Introduction to Systolic Algorithm Design, Clara don press Oxford, 1992.

[26] N. Packard, F. C. Richards and T. P. Mayer, “Extracting Cellular Automata Rules Directly from Experimental Data”, Physica D, Vol. 45, pp. 189-202, 1990.

[27] W. Li and N. Packard, “The Structure of Elementary Cellular Automata Rule Space”, Complex Systems, Vol 4, pp. 281-297, 1990.

[28] S. Wolfram, Theory and Applications of Cellular Automata, World Scientific, 1986.

[29] W. Burks, Essays On Cellular Automata, Urbana, IL :University of Illinois Press, 1970.

[30] S.G. Akl, Parallel Sorting Algorithms, Orlando, FL: Academic, 1985.

[31] R. K. Squier and K. Steiglitz, “Programmable Parallel Arithmetic in Cellular Automata Using a Particle Model”, Complex Systems, Vol. 8,pp. 311-323,1994.

[32]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.

[332] D. Thompson, and H. T. Kung, “Sorting on a Mesh Connected Parallel Computer”, Communication of ACM, Vol. 20, pp. 263-271, 1977.

[34] K. E. Batcher, “Sorting Network and Their Applications”, AFIP Proc, Vol. 32, pp. 307-314, 1968.]

[35] S. Orcutt, “Computer Organization and Algorithms for Very High Speed Computations”, PhD. Thesis, Stanford University, 1984.

[36] M. Kummar and D. S. Hirschberg, “An Efficient Implementation of Batcher’s Odd-Even Merge Algorithms and its Application to Parallel Sorting Schemes”, IEEE Transaction on Comput, C-32, pp. 254-264, 1983.
[37] Sh. Golzari and M. Meybodi, “Sorting Algorithms for Two Dimensional Cellular Automata”, Computer Engineering Dept. Technical Report, Amirkabir University of Technology, 2000