دراسة كفاءة خوارزميات الترتيب بالتطبيق على خوارزميات (Shell, Merge, Heap, Radix and NK-sorting) ومشكلة التصادم في خوارزمية NK-sorting

أ.د. السماني عبد المطلب أحمد
جامعة النيلين
كلية علوم الحاسوب وتقانة المعلومات
Profsmani@gmail.com

احمد فتح العليم عبيد الله
جامعة القرآن الكريم والعلوم الإسلامية
كلية علوم الحاسوب وتقانة المعلومات
ahmed_first_222@hotmial.com

المستخلص:

تتمحور هذه الدراسة حول مفهوم الترتيب حيث يعتبر ترتيب البيانات من الاستخدامات المتكررة في معالجة البيانات [17], تناولنا في هذا الدراسة عدد من خوارزميات الترتيب الحديثة وهي (خورزمية شل ،الترتيب بالدمج ،خوارزمية الكومة، خوارزمية رادكس، وخوارزمية إن كي).

ومن خلال دراسة ومقارنة هذه الخوارزميات وجدنا أن خوارزمية Nk-sorting تعطي نتائج بطيئة من حيث السرعة وذلك نسبة لوجود مشكلة التصادم في هذه الخوارزمية، كما وجدنا أن خوارزمية Shell تعطي نتائج بطيئة مع الأعداد العشوائية مقارنة ببقية الخوارزميات الأخرى.لكن بصورة عامة أثبتت خوارزمية Radix كفاءتها مع جميع أحجام البيانا ت التي تم استخدامها في الدراسة مع وجود بعض الخوارزميات التي أثبتت أنها الأسرع في حالات فردية.

الكلمات المفتاحية:

خوارزمية ، الترتيب ، المصفوفة ،رادكس

Abstract

This study focuses on the concept of Sorting, in which data Sorting is the repeated use in data processing, we mean by the data sorting is set records are sorted using sorting algorithm. That the most important uses of sorting is to facilitate or speed up the search for the record[20].

in this study we used number of modern sorting algorithm , this algorithm is (shell Sort Algorithm, Merge Sort algorithm, Heap Sort algorithm, Radix Sort algorithm and NK-Sorting algorithm) with varying efficiency of these algorithms in terms of speed ,space of memory and stability algorithm.

Keywords:

Radix, Sorting, Stability, Heap, Shell

مقدمة:

استخدمنا خوارزميات الترتيب التالية ومقارنتها لعرفة كفاءة كل واحدة ونقاط القوة والضعف في كل منها وهذه الخوارزميات هي:

Shell sort[1]

هذه الخوارزمية مختلفة قليلا عن بقية الخوارزميات في زمن التعقيد حيث أنها تختلف قليلا عن الصيغة O(long) لتصبح صيغتها مشابهه للصيغة بين O(nlog2n) و O(n1.5) اعتمدا على تفاصيل التنفيذ.

Heap sort[1]

هي خوارزمية شعبية ولكنها ليست الأسرع في كل الحالات . حيث يكون زمن تنفيذها دائماً اقل من O(nlogn) . ويعتبر الترتيب بهذه الخوارزمية جدير بالثقة .

Merge sort[1]

تمثل زمن تنفيذها بصورة O(nlogn) وتعتبر من الخوارزميات المستقرة stable (لا يتغير ترتيب العناصر المتساوية).

Radix Sort[24]

تقوم خوارزمية راديكس radix على مفهوم ترتيب البيانات على مستوى الخانات العشرية digits حيث أن كفاءة خوارزمية راديكس هي : O(n.k/d) حيث أن n هو عدد الأرقام المراد ترتيبها و k هو حجم أي مفتاح و d هو عدد الخانات العشرية digits .

NK-sorting Algorithm[2]

هي خورزمية تقوم على فكرة توزيع العناصر المراد ترتيبها على عدد من المصفوفات المؤقتة بناءا على عدد الخانات العشرية ومن ثم ترتيب كل مصفوفة على حدا ومن ثم تجميعها مرة اخرى.

مشكلة التصادم في خوارزمية NK-Sorting

تعتمد الخوارزمية على مقارنة الأعداد لإجراء عملية الترتيب .إذن كلما كان عدد المقارنات بسيط كلما كانت الخوارزمية أسرع.

تقوم الخوارزمية بتقسم المصفوفة إلى مصفوفات فرعية Sub-arrays حيث يتم ترتيب كل مصفوفة على حدا.

تقوم الخوارزمية بالبحث عن اصغر عنصر واكبر عنصر في كل المصفوفات الفرعية حيث تتم هذه العملية مرة واحدة فقط. إذن الحالة الأفضل Best case لهذه الخوارزمية هو O(n).لكن بالرغم من ذلك قد تحدث مشكلة تصادم في الخوارزمية مما يتطلب عملية بحث عن موقع آخر للعنصر المراد تريبه وذلك بسبب أن الموقع المعطى تم السكين فيه من قبل وبالتالي تحتاج الخوارزمية إلى زمن أطول قد يصل إلى $O(n^2)$ في أسوأ الحالات Worst case

كذلك نجد أن الخوارزمية تقوم بتقسيم المصفوفة الأصلية إلى عدد من المصفوفات الفرعية مجموع أطوال هذه المصفوفات يساوي طول المصفوفة الأصلية.وكذلك لا نحتاج إلى مساحة إضافية في المصفوفة الأصلية لإجراء المقارنات.إذن في كل الحالات تحتاج الخوارزمية إلى مساحة O(n) بالإضافة إلى مساحات بسيطة للمتغيرات S ,min ,max لكل مصفوفة فرعية. كما يمكننا أن نضع المصفوفة المصفوفات الفرعية في وسط تخزين خارجي وإدخال مصفوفة واحدة فقط للذاكرة وبعد انتهاء ترتيبها يتم إخراجها وإدخال المصفوفة الثانية وهكذا.يمكن استخدام هذه الطريقة مع البيانات الضخمة.

عند حساب موقع العنصر في المصفوفة الفرعية من خلال علاقة العنصر مع وقعة قد يحدث أن يعطي أكثر من عنصر نفس الموقع وذلك بسبب أن العنصرين متساويين في القيمة وبالتالي يعطيان نفس الموقع أو أن الفرق بين العنصرين صغير وبالتالي يكون الفرق بين موقعيهما عبارة عن كسر صغير وكما نلاحظ أن الخوارزمية تقوم بتقريب الأعداد الكسرية (لان المواقع هي أرقام صحيحة) وبالتالي يعطيان نفس الموقع. مما يتسبب في حدوث تصادم collision

لحل مشكلة التصادم تقوم الخوارزمية بإزاحة العناصر ناحية اليمين أو الشمال (حسب نوع الترتيب، تصاعدي أو تنازلي). عملية البحث عن موقع خالي وإزاحة العناصر يتطلب زمنا أطول وكذلك قد يؤدي إلى غموض في الخوارزمية نسبة لان الخوارزمية تقترح أن حاصل ضرب قيمة العنصر في الميل S يقود إلى موقع العنصر في المصفوفة وهذا ليس صحيح دائما (في حالة التصادم).

المقارنة بين خوارزميات الترتيب :

معظم خوارزميات الترتيب تعمل من خلال مقارنة البيانات .حيث إن هذه الخوارزميات تقوم بمقارنة عدد كبير من البيانات، وهنال ك خوارزميات تقوم بمقارنة جزء فقط من البيانات (مثلا ترتيب أسماء أشخاص تقوم بمقارنة الحروف الأولى فقط من الاسم).

وعادة ما يحدد أفضلية الخوارزمية هو كفاءتها وكفاءة الخوارزمية في هذه الحالة تكون بناءا على كمية البيانات المراد ترتيبها ومدى سرعة ترتيب هذه البيانات وكمية المساحة المستخدمة من الذاكرة. حيث نجد إن معظم خوارزميات الترتيب لها كفاءة أما O(n^2) أوO(n*log(n))

وهنالك بعض خوارزميات الترتيب ترتيب البيانات أسرع من O(n*log(n)) هذه الخوارزميات لا تعتمد على مقارنة البيانات وإنما تقوم على مجموعة من الحيل.

هنالك العديد من خوارزميات الترتيب لها نفس الكفاءة ولكنها تختلف في السرعة مع نفس المدخلات .لذا للحكم على الخوارزمي ة يجب دراسة الحالة الأفضل best-cast والحالة الوسط average-case والحالة الأسوأ worst-case . والمقياس الثاني هو استخدام معامل ثابت هو Big O notation الذي يعطي تفاصيل عن سير العمليات ويعطي لمحة عامة عن الخوارزمية.

أما العامل الثاني الذي من خلاله يمكن الحكم على الخوارزمية هو حجم المساحة المطلوبة .حيث إن هنالك يمكن إن تقوم بترتيب البيانات في نفس المساحة دون الحوجة إلى مساحة إضافية (فرز الكومة heap sort ، على سبيل المثال ، يمكن ترتيبه في نفس المكان ، ولكن يمكن أن يتم في بعض الحالات في كومة منفصلة).. وفي بعض الحالات قد يتوقف حجم المساحة المطلوبة على نوع بنية البيانات المستخدمة.(على سبيل المثال الترتيب بواسطة خوارزمية الدمج merge sort مع بيانات مخزنة في مصفوفات arrays يختلف عنه مع بيانات مخزنة في قوائم متصلة linked-list).

العامل الثالث للحكم على الخوارزمية هو استقرار الخوارزمية stability وهل الخوارزمية تقوم بترتيب البيانات دون تغيير المفاتيح keys أم لا

الجدول التالي يمثل قياس أداء خوارزميات الترتيب :

algorithm	Time				
Sort	**Best**	**Average**	**Worst**	**Space**	**Stability**
Shell Sort	O(n)	O($n(\log n)^2$) Or O($n^{3/2}$)	depends on gap sequence. Best known: $O(n\log^2 n)$	Constant	Instable
Heap Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Constant	Instable
Merge Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Depends	Stable
LSD Radix Sort	--	O($n \cdot \frac{k}{d}$)	O($n \cdot \frac{k}{d}$)	n	Stable
MSD Radix Sort	--	O($n \cdot \frac{k}{d}$)	O($n \cdot \frac{k}{d}$)	$n + \frac{k}{d} \cdot 2^d$	Stable
NK-sorting	O(n)	O(n*log(2))	O(n^2)	constant	Instable

جدول(1) مقارنة بين أداء خوارزميات الترتيب

من خلال الجدول السابق (1) نجد ان :

- Shell sort :

- زمن التنفيذ time: زمن تنفيذ خوارزمية Shell تقدم أداءاً مميزا في الحالة الأفضل Best case لكن سرعان ما ينحدر إلى الأسوأ في الحالة المتوسطة Average case والحالة الأسوأ worst case .
- المساحة space: المساحة التي تتطلبها خوارزمية shell هي مساحة ثابتة constant.
- استقرار الخوارزمية stability: الخوارزمية غير مستقرة Instable تتطلب تسكين العناصر في عناصر فرعية وترتيب كل قائمة فرعية على حدا ومن ثم جمعها مع بعض مما يقود إلى تغيير في المواقع Keys بالنسبة للقائمة الأصلية.

- Heap sort :
  - زمن التنفيذ Time: زمن تنفيذ الخوارزمية يتشابه في كل الحالات (الأفضل والمتوسطة والأسوأ) لأنها في جميع الحالات تقوم بتوزيع العناصر على شجرة الكومة وإجراء كل المقارنات بغض النظر عن العناصر مرتبة أم لا، وبالتالي لا فرق بين كل الحالات (بالنسبة لعدد المقارنات).
  - المساحة Space :المساحة التي تتطلبها الخوارزمية هي مساحة ثابتة .
  - استقرار الخوارزمية Stability : تعتبر الخوارزمية غير مستقرة instable وذلك لان الخوارزمية لا تحافظ على الترتيب النسبي للعناصر بسبب أن العناصر يتم توزيعها على شجرة ثنائية مما يؤدي إلى تغيير في ترتيب مواقع العناصر بالنسبة للقائمة الأصلية

- Merge Sort:
  - زمن التنفيذ Time: نلاحظ أن زمن تنفيذ الخوارزمية لا يتغير في كل الحالات (الأفضل والأسوأ والمتوسط).حيث أن الخوارزمية في كل الحالات تقوم بتقسيم القائمة إلى قوائم اصغر وتقسيم القوائم الأصغر إلى قوائم اصغر وهكذا ومن ثم أجراء المقارنات وإعادة دمج القوائم مرة أخرى إلى أن تصبح قائمة واحدة.
  - المساحة Space : المساحة التي تتطلبها الخوارزمية من الذاكرة تعتمد على عدد العناصر المراد ترتيبها .فكلما كان عدد العناصر كبير أصبحت عملية التقسيم أكثر وبالتالي عدد قوائم فرعية أكثر ومساحة اكبر من الذاكرة.
  - الاستقرار Stability :تعتبر خوارزمية مستقرة ،لأنها تقوم بتقسيم ودمج العناصر بنفس ترتيب مواقعها في القائمة الأصلية .وبالتالي لا تتأثر مواقع العناصر في القائمة الأصلية.

- Radix sort:
  - زمن التنفيذ Time: يعتبر زمن تنفيذ الخوارزمية اقرب إلى O(n) في الحالة الأفضل.حيث أن الخوارزمية لا تستخدم مفهوم المقارنة وإنما ترتب العناصر بناءا على ترتيب الخانة العشرية.وفي الحالة الأفضل تصبح الخوارزمية خوارزمية خطية.لكن في الحالة المتوسطة والحالة الأسوأ يصبح زمن التنفيذ معتمدا على عدد الخانات العشرية في الأرقا م المراد ترتيبها وكذلك حجم كل المفاتيح المستخدمة في الترتيب.
  - المساحة في الذاكرة space: تحتاج الخوارزمية إلى مساحة ثابتة في الذاكرة هي n حيث أن n هو عدد العناصر المراد ترتيبها(بالنسبة لخوارزمي ة LSD

Radix) أما Radix MSD فقد تحتاج مصفوفة إضافية لإتمام عملية الترتيب.

- الاستقرار Stability:تعتبر الخوارزمية مستقرة ولا تغير في الترتيب النسبي للعناصر.

- NK-Sorting :

- زمن التنفيذ: يعتبر زمن تنفيذ الخوارزمية هو سريع (نظريا) وذلك في الحالة الأفضل Best Case لأن الخوارزمية توجد العلاقة بين قيمة العنصر وموقعة مباشرة دون الحاجة إلى مقارنات وتبديلات.لكن بسبب حدوث تصادم في الخوارزمية يصبح أداء الخوارزمية بطيء جدا ويشبه خوارزمية الترتيب بالإدخال في بعض الأحيان.حيث أنه وفي حالة حدوث تصادم (وهو محاولة تسكين عنصر في موقع مسكن به عنصر من قبل) تضطر الخوارزمية إلى البحث عن موقع آخر فارغ وتقوم بإزاحة العناصر لتسكن العنصر في مكانه المناسب. يمين أو يسار الموقع الذي يفترض أن يسكن به.
- المساحة في الذاكرة Space: لا تحتاج الخوارزمية إلى مساحة إضافية حيث أن الخوارزمية تقسم العناصر على عدد من المصفوفات الفرعية .طول هذه المصفوفات الفرعية مجتمعة يساوي طول المصفوفة الأصلية.
- استقرار الخوارزمية Stability :

تعتبر الخوارزمية غير مستقرة لأنها تلجأ إلى حساب مواقع العناصر في القائمة الفرعية ومن ثم جمع القوائم مع بعضها على حسب عدد الخانات العشرية .لذا تنتج قائمة مختلفة الترتي ب النسبي للعناصر – بالنسبة لمواقع العناصر- عن القائمة الأصلية (قبل الترتيب).

تطبيق:

لاختبار سرعة ترتيب خوارزميات الترتيب المستخدمة قمنا بكتابة برامج بلغة C++ بحيث يتم تنفيذها عدة مرات مع بيانات عشوائية يتم توليدها بواسطة خوارزمية توليد بيانات عشوائية.

تم تنفيذ البرامج في جهاز Pentium 4 CPU 3 GHz ,1.12 of RAM

بحيث تم استخدام دالة ()rand توليد الأرقام العشوائية. تختلف هذه الأرقام العشوائية في كل مرة يتم فيها استدعاء هذه الدالة.

كما تم استخدام إحجام مصفوفات مختلفة لتوضيح الفرق في سرعة التنفيذ مع كل حجم .حيث نجد أن هنالك خوارزميات تكون فعالة مع بيانات قليلة ونجدها بطيئة مع البيانات الكبيرة.

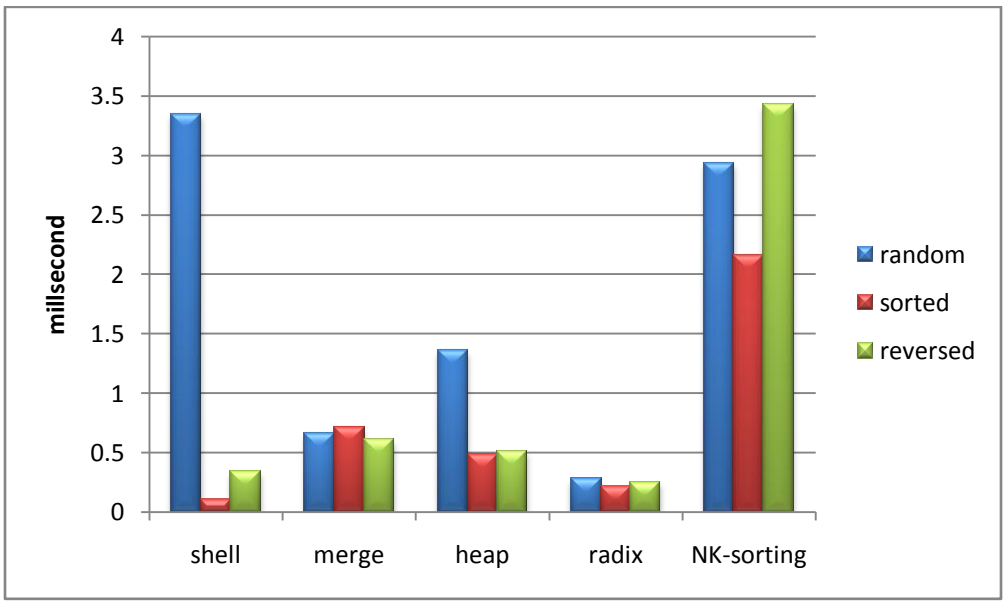
استخدمنا ثلاث توزيعات مختلفة من الأرقام العشوائية هي:

- أرقام عشوائية تماما (random).

- أرقام مرتبه مسبقا (sorted). وهي أرقام تم توليدها عشوائيا ومن ثم تم ترتيبها وتخزينها في مصفوفة لمعرفة فعالية الخوارزمية على إعداد مرتبه أصلا.
- أرقام معكوسة الترتيب (reversed). وهي أرقام تم توليدها وترتيبها بصورة معكوسة وتخزينها في صورة مصفوفة وذلك لقياس كفاءة وسرعة الخوارزمية في هذه الحالات .تعتبر الحالة الأسوأ worst case

	shell	merge	Heap	Radix	Nk-sort
random	3.344	0.657	1.36	0.282	2.934
sorted	0.109	0.719	0.484	0.218	2.165
reversed	0.344	0.609	0.515	0.25	3.426

جدول (2) يوضح زمن التنفيذ لترتيب 1000000 عدد صحيح



شكل(1) يوضح زمن التنفيذ لترتيب 1000000 عدد صحيح

الخلاصة:

تتفاوت كفاءة خوارزميات الترتيب من حيث السرعة والمساحة المستخدمة من الذاكرة وكذلك استقرار الخوارزمية .و لمعرفة أي من هذه الخوارزميا ت هي الأكفأ قمنا بدراسة ومقارنة هذه الخوارزميات مع بعضها البعض مستخدمين مصفوفات من الأعداد العشوائية الصحيحة.

ومن خلال دراسة ومقارنة هذه الخوارزميات وجدنا أن خوارزمية Nk-sorting تعطي نتائج بطيئة من حيث السرعة وذلك نسبة لوجود مشكلة التصادم (collision) في هذه الخوارزمية، كما وجدنا أن خوارزمية Shell تعطي نتائج بطيئة مع الأعداد العشوائية مقارنة ببقية الخوارزميات الأخرى حيث تم تحسين هذه الخوارزمية لتصبح أكثر كفاءة وسرعة .لكن بصورة عامة أثبتت خوارزمية Radix كفاءتها مع جميع أحجام البيانا ت التي تم استخدامها في الدراسة مع وجود بعض الخوارزميات التي أثبتت أنها الأسرع في حالات فردية.كما وجدنا أن خوارزميتي Merge و heap تتفاوت سرعتهما من نوع بيانات إلى آخر ولكنهما بصورة عامة اقل سرعة من خوارزمية Radix .

3 المراجع:

1. Robert Sedgewick and Kevin Wayne , Algorithms, 4th edition, Princeton University, 2011.
2. Nidhal K. El abbadi, Zayed Yahya A.karem -NK-sorting ,University of kufa,Iraq ,2011.
3. Robert Sedgwick ,Algorithms in C ,Princeton university,
4. Aho, Alfred V., "Data structures and algorithms", Reading, Mass : Addison-Wesley, c1983.
5. Aho, Alfred V., "The design and analysis of computer algorithm".
6. Budd, Timothy A., "Classic data structures in C++ Reading", Mass. : Addison-Wesley Pub. Co., c1994.
7. Herbert S. Wilf , Algorithms and Complexity, University of Pennsylvania Philadelphia, PA 19104-6395

8. Schneider, M. and J. Gersting (1995), An Invitation to Computer Science, West Publishing Company, New York, NY, p. 9.
9. G P Raja Sekhar , Lecture Notes on Design & Analysis of Algorithms, Department of Mathematics I I T Kharagpur
10. Thomas Niemann, SORTING AND SEARCHING ALGORITHMS, Portland, Oregon.1999
11. Mark Allen Weiss, " Mark Allen Weiss, Data Structures and Algorithm Analysis in C++ (3rd Edition)" , Addison-Wesley, 2007
12. Augenstein & Tenenbaum Langsam, Data Structures in C & C++ (2nd Edition, Prentice Hall of India, 2007
13. Jasmin Blanchette, Mark Summerfield, C++ GUI Programming with Qt 4, Second Edition, Prentice Hall. February 14, 2008
14. السماني عبد المطلب أحمد , هياكل البيانات, منشورات جامعة السّودان المفتوحة، الطبعة الأولى 2005م.
15. http://en.wikipedia.org/wiki/Algorithm 21/10/2011 9:51 am.
16. http://en.wikipedia.org/wiki/Sorting_algorithm 21/10/2011 12:11 pm.
17. http://www.personal.kent.edu/~rmuhamma/Algorithms/algorithm.html 12/10/2011 20:43 pm.
18. http://www.cprogramming.com/tutorial/computersciencetheory/sortcomp.html 29/11/2011 00:24 am.
19. http://www.math.upenn.edu/~wilf/ 29/11/2011 01:30 am.
20. *http://en.wikipedia.org/wiki/Radix_sort* 29/11/2011 00:24 am.