مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (30) العدد (4) 2008
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (30) No. (4) 2008

خوارزمية الفرز الأمثلية لسجلات قواعد البيانات المعقدة
Optimal Sorting Algorithm for complex database

الدكتور علي سليمان *

(تاريخ الإيداع 4 / 5 / 2008. قُبِل للنشر في 2008/10/23)

□ الملخّص □

منذ بدايات استخدام الحاسب جذبت مسائل فرز البيانات عدداً كبيراً من الباحثين، وذلك بسبب صعوبة إيجاد أفضل الحلول لها، حيث تم تحليل خوارزمية الفرز الفقاعي bubble sort في مطلع العام 1960، وعلى الرغم من أن العديد من الباحثين اعتبرها مسألة محلولة، فإن العديد من خوارزميات الفرز الجديدة لا تزال تستنتج حتى الآن (فعلى سبيل المثال تم نشر خوارزمية الفرز المكتبي library sort لأول مرة عام 2004).

سنتطرق إلى التصنيف من وجهات النظر التالية:

1-التصنيف حسب أسلوب التحقيق

2-التصنيف حسب أسلوب التصميم

3-التصنيف حسب مجال الدراسة.

4-التصنيف حسب درجة التعقيد.

يخلص البحث إلى تقويم مجموعة من خوارزميات الفرز ومقارنة درجات تعقيدها وتصنيفها واستخلاص حل أمثل لمسألة فرز سجلات قواعد البيانات المعقدة ومثال عليها قاعدة البيانات الخاصة بمزود خدمة الانترنت.

الكلمات المفتاحية: خوارزمية الفرز- درجة التعقيد- العودية- التكرارية- حتمية- قسم وتابع- البرمجة الديناميكية- الطريقة الطماعة- البرمجة الخطية- التخفيض- البحث والتعداد- الأسلوب الاحتمالي أو الإرشادي.

* أستاذ مساعد – قسم هندسة الحاسبات والتحكم الآلي كلية الهندسة الميكانيكية والكهربائية- جامعة تشرين – سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (30) العدد (4) 2008
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (30) No. (4) 2008

An Optimal Sorting Algorithm for Complex Databases

Dr. Ali Suleiman *

(Received 4 / 5 / 2008. Accepted 23 / 10 / 2008)

□ ABSTRACT □

Since the beginning of using computing, the sorting problem has attracted a large number of researchers; this is because of the complexity of finding the best solution to it. For example, bubble sorting was first analyzed in the early 1960s. Although many consider it a resolved problem, new sorting algorithms are still being invented to date (for example, library sorting was first published in 2004). In this paper, we study sorting algorithms as classified by:

- Implementation
- Design paradigm
- Field of study
- Complexity.

This research aims at assessing some sorting algorithms, comparing their complexity and classification, and finding the best solution to the problem of sorting records on an Internet Service Provider (ISP) database.

keywords: sorting algorithm, complexity, recursion, iteration, deterministic, divide and conquer, dynamic programming, reduction, search and enumeration, probabilistic and heuristic.

* **Associate Professor, Department of Computer and Automation Faculty of Mechanical and Electrical Engineering, Tishreen University, Lattakia, Syria.**

مقدمة:

خوارزمية الفرز من وجهة نظر علوم الحاسوب والرياضيات هي خوارزمية تضع عناصر لائحة ما بترتيب محدد. وأشكال الترتيب الأكثر استخداماً هي الترتيب بحسب الأرقام أو الترتيب بحسب الأحرف الأبجدية.

إن الفرز الفعال مهم جداً لتحسين استخدام الخوارزميات الأخرى (مثل خوارزميات البحث أو الدمج) التي تتطلب العمل على لوائح مرتبة لتعمل بشكل صحيح، كما أن الفرز الفعال مهم لقوننة البيانات لإنتاج خرج قابل للقراءة، بشكل أوضح، ويجب على الخرج أن يحقق الشرطين التاليين:

1.أن يكون الخرج بترتيب غير تنازلي (كل عنصر يجب أن لا يكون أصغر من العنصر الذي يسبقه بحسب الترتيب المرغوب).

2.الخرج أن يكون عبارة عن تبديل أو إعادة ترتيب للدخل.

منذ بدايات استخدام الحاسب جذبت مسائل فرز البيانات عدداً كبيراً من الباحثين، وذلك بسبب صعوبة إيجاد أفضل الحلول لها، حيث تم تحليل خوارزمية الفرز الفقاعي bubble sort في مطلع العام 1960، وعلى الرغم من أن العديد من الباحثين اعتبرها مسألة محلولة، فإن العديد من خوارزميات الفرز الجديدة لا تزال تستنتج حتى الآن (فعلى سبيل المثال تم نشر خوارزمية الفرز المكتبي library sort لأول مرة عام 2004).

يتناول هذا البحث خوارزميات الفرز من منظور مختلف يعتمد الأسلوب المستخدم في تحقيق هذا الفرز، حيث يسرد عدداً من الخوارزميات المختلفة المنتمية إلى كل صنف، ويتناول بالبحث واحدة من الخوارزميات من كل صنف بشيء أكبر من التفصيل، ويقوم بتطبيقها على سجلات قاعدة بيانات لمؤسسة الاتصالات تقوم بتسجيل المكالمات التي يجريها المشتركون ومقارنة النتائج.

أهمية البحث وأهدافه:

يقوم مخدم قواعد البيانات مزود خدمة الانترنت ISP في الشركات تلقائياً بتسجيل جميع الاتصالات التي تجري عبر الشبكة (وبعد التحقق منها) ببدء جلسة عمل session يقوم خلالها بتسجيل اسم المستخدم، وساعة الدخول، وتاريخ الدخول، وساعة الخروج. إن عدد الاتصالات التي تجرى كبير جداً، وبالتالي فإن عملية إصدار الفواتير أو عملية الاستعلام عن مفصل أوقات الاستخدام التي تمت من مشترك معين هي عملية مكلفة زمنياً بشكل كبير، إذا اعتمدت فقط على خوارزمية بحث عامة، لأنها ستتطلب اختبار كافة السجلات الموجودة في قاعدة البيانات في حين أن استخدام الخوارزمية المناسبة، هو بكل تأكيد أكثر فعالية وسرعة، لأن الاختبار لن يشمل سوى جزء بسيط من سجلات قاعدة البيانات. يهدف هذا البحث إلى تقييم مجموعة من خوارزميات الفرز ومقارنة درجات تعقيدها، واستخلاص حل أمثل لمسألة فرز سجلات قاعدة البيانات الخاصة بمزود الخدمة والتي تحتاج إلى حاسب شخصي عادي المواصفات لحساب ومقارنة درجات التعقيد، وتم تطبيق البحث على ثلاثة حواسيب في جامعة تشرين.

طريقة البحث ومواده:

انطلاقاً من المفاهيم النظرية الأساسية مثل نظرية التعقيد، التصنيف بحسب أسلوب (التحقيق، التصميم، مجال الدراسة، درجة التعقيد) نستخلص الفرز الأمثلي المستخدم في قواعد معطيات مزود خدمة الإنترنت.

- **نظرية التعقيد الحسابي:**

تقوم نظرية التعقيد الحسابي – كفرع من نظرية الحوسبة وعلوم الحاسوب – بدراسة المسائل المتعلقة بكمية الموارد اللازمة لتنفيذ الخوارزميات (زمن التنفيذ على سبيل المثال) والتعقيد الحاصل في تحقيق خوارزميات فعالة لمسائل حسابية محددة.

تطرح هذه النظرية السؤال النموذجي التالي: " عندما يكبر حجم الدخل لخوارزمية، كيف يتغير زمن التنفيذ ومتطلبات الذاكرة للخوارزمية وما هي النتائج المتأتية عن هذا التغير ؟. " بمعنى آخر، تعنى هذه النظرية بدراسة تدرج المسائل والخوارزميات الحسابية، وبشكل خاص تحدد النظرية الحدود الفعلية لما يستطيع الحاسوب إنجازه.

أحد الاتجاهات الهامة لهذه النظرية هو تصنيف المسائل والخوارزميات الحسابية إلى صفوف تعقيد complexity classes، والسؤال المطروح الأكثر أهمية لنظرية التعقيد، هو فيما إذا كان صف التعقيد P (وهو عبارة عن مجموعة من مسائل القرار التي يمكن أن تحل بواسطة آلية حتمية خلال زمن علاقته على شكل كثير حدود) هو نفسه صف التعقيد NP (وهو عبارة عن مجموعة من مسائل القرار التي يمكن حلها بواسطة آلية غير حتمية خلال زمن علاقته على شكل كثير حدود)، أو أنه مجموعة جزئية كما يعتقد عموماً. لقد تم التحقق خلال فترة قصيرة من طرح هذا السؤال أن العديد من مسائل الصناعة المهمة في مجال أبحاث التشغيل هي من صنف جزئي NP يدعى NP-complete. إن المسائل من الصنف NP-complete تتصف بأن حلولها سريعة الاختبار. [3] تتعامل نظرية التعقيد complexity theory مع التعقيد الحسابي النسبي للتوابع القابلة للحساب. وهذا يختلف عن نظرية قابلية الحساب computability theory التي تعنى بدراسة، فيما إذا كانت المسألة قابلة للحل بغض النظر عن الموارد اللازمة.

تقوم نظرية التعقيد بتحليل درجة التعقيد للمسائل الحسابية بدلالة العديد من الموارد الحسابية، إن المسألة الواحدة يمكن التعبير عنها بدلالة الكميات اللازمة من العديد من أنواع الموارد الحسابية بما في ذلك الزمن، المساحة، العشوائية randomness، التعاقب alternation وغيرها من الموارد الأخرى.

إن أكثر الموارد الحسابية التي تمت دراستها هي الزمن اللازم deterministic time (DTIME) و المساحة اللازمة deterministic space (DSPACE).

إن مسألة منفردة هي عبارة عن مجموعة من الأسئلة المترابطة، حيث كل سؤال هو سلسلة منتهية الطول.

التعقيد الزمني time complexity لمسألة هو عدد الخطوات اللازمة لحل مسألة ما بدلالة حجم الدخل باستخدام الخوارزمية الأكثر فعالية. يقاس التعقيد الزمني باستخدام Big O notation.

التعقيد المكاني space complexity لمسألة هو قياس كمية الذاكرة اللازمة لتنفيذ الخوارزمية، يقاس التعقيد المكاني باستخدام Big O notation. [3]

- **مسائل القرار :**

يتعامل الجزء الأكبر من نظرية التعقيد مع مسائل القرار. مسألة القرار هي مسألة يكون فيها الجواب نعم أو لا. تميز نظرية التعقيد بين المسائل التي تحقق الجواب نعم أو التي تحقق الجواب لا. أما المسائل التي تقلب الأجوبة نعم و لا لمسألة أخرى تدعى المتمم لتلك المسألة.

- **المقاربات اللانهائية:**

إن الترميز Big O notation مفيد جداً لدى تحليل فعالية الخوارزميات، على سبيل المثال، الزمن (عدد الخطوات) اللازمة لحل مسألة من الحجم n يمكن أن يكون $T(n)=4n^2-2n+2$. عندما تكبر قيمة n كثيراً، فإن الحد n^2 سيصبح مهيمناً، وبالتالي يمكن إهمال باقي الحدود (فمثلاً عندما تكون n=500 فإن الحد $4n^2$ هو أكبر بـ 1000 مرة من الحد 2n، وبالتالي فإن تجاهله سيكون ذا تأثير يمكن إهماله).

وبالتالي يمكن أن نكتب:

$$T(n) \in O(n^2)$$

وتعني أن الخوارزمية ذات درجة تعقيد زمني من الدرجة n^2.

المقاربات اللامتناهية في الصغر:

يمكن استخدام الترميز Big O notation أيضاً لتعريف حد الخطأ error term خلال عملية التقريب لتابع رياضي، على سبيل المثال:

$$e^x = 1 + x + \frac{x^2}{2} + O(x^3) \qquad \text{as } x \to 0$$

يعبر عن حقيقة أن الفرق $e^x - (1 + x + \frac{x^2}{2})$ هو أقل بالقيمة المطلقة من $|x^3|$ عندما تكون x قريبة من 0.

بالإجمال وباختصار، يمكن القول إن f هو من مرتبة g (أي f(x)=O(g(x))) إذا وفقط إذا وجد عدد حقيقي موجب M وعدد حقيقي x_0 بحيث إنه من أجل جميع قيم x فإن $|f(x)| \le M.g(x)$ حيث $x>x_0$.

الجدولين (1و2) يبينان بعض الترميزات الأخرى التي تستخدم للتعبير عن تعقيد الخوارزميات مع حساب درجة التعقيد:

الجدول (1) يبين بعض الترميزات التي تستخدم للتعبير عن تعقيد الخوارزميات :

Notation	Intuition	Definition
$f(n) \in \mathcal{O}(g(n))$	g is an asymptotic upper bound on f (up to constant factor)	$\exists(C>0), n_0 : \forall(n > n_0)\ \lvert f(n)\rvert < \lvert Cg(n)\rvert$
$f(n) \in \Omega(g(n))$	g is an asymptotic lower bound on f (up to constant factor)	$\exists(C>0), n_0 : \forall(n > n_0)\ \lvert Cg(n)\rvert < \lvert f(n)\rvert$
$f(n) \in \Theta(g(n))$	f is bounded tightly by g asymptotically	$\exists(C, C'>0), n_0 : \forall(n > n_0)\ \lvert Cg(n)\rvert < \lvert f(n)\rvert < \lvert C'g(n)\rvert$
$f(n) \in o(g(n))$	f is dominated by g asymptotically	$\forall(C>0), \exists n_0 : \forall(n > n_0)\ \lvert f(n)\rvert < \lvert Cg(n)\rvert$
$f(n) \in \omega(g(n))$	f dominates g asymptotically	$\forall(C>0), \exists n_0 : \forall(n > n_0)\ \lvert Cg(n)\rvert < \lvert f(n)\rvert$
$f(n) \sim g(n)$	asymptotically equal	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$

الجدول (2) يبين تعقيد بعض الخوارزميات (حيث أن c هو ثابت عشوائي).

Notation	Name	Example
$\mathcal{O}(1)$	Constant	Determining if a number is even or odd
$\mathcal{O}(\alpha(n))$	inverse Ackermann	Amortized time per operation when using a disjoint-set (union-find) data structure
$\mathcal{O}(\log^* n)$	iterated logarithmic	The find algorithm of Hopcroft and Ullman on a disjoint set
$\mathcal{O}(\log n)$	Logarithmic	Finding an item in a sorted list with the binary search algorithm
$\mathcal{O}((\log n)^c)$	Polylogarithmic	Deciding if n is prime with the AKS primality test
$\mathcal{O}(n^c), 0 < c < 1$	fractional power	searching in a kd-tree
$\mathcal{O}(n)$	Linear	Finding an item in an unsorted list
$\mathcal{O}(n \log n)$	linearithmic, loglinear, or quasilinear	Sorting a list with heapsort, computing a FFT
$\mathcal{O}(n^2)$	Quadratic	Sorting a list with insertion sort, computing a DFT
$\mathcal{O}(n^c), c > 1$	polynomial, sometimes called algebraic	Finding the shortest path on a weighted digraph with the Floyd-Warshall algorithm
$\mathcal{O}(c^n)$	exponential, sometimes called geometric	Finding the (exact) solution to the traveling salesman problem (under the assumption that P ≠ NP)
$\mathcal{O}(n!)$	factorial, sometimes called combinatorial	Determining if two logical statements are equivalent[1], traveling salesman problem, or any other NP-complete problem via brute-force search
$\mathcal{O}(n^n)$	n to the n	
$\mathcal{O}\left(c_1^{c_2^n}\right)$	double exponential	Finding a complete set of associative-commutative unifiers[2]

- **تصنيف الخوارزميات:**

هناك العديد من الطرق لتصنيف الخوارزميات ولكل منها مستلزماتها. نبين فيما يلي بعضاً من هذه الطرق:

1 – التصنيف حسب أسلوب التحقيق:

تصنف الخوارزميات حسب أسلوب التحقيق إلى:

- خوارزمية العودية recursion أو التكرارية iteration: الخوارزمية العودية هي تلك التي تشير إلى نفسها بشكل متكرر إلى أن تتحقق بعض الشروط. تستخدم الخوارزميات التكرارية تستخدم مقاطع متكررة كالحلقات، وبنى

معطيات إضافية كالمكدسات لحل مسائل معينة. إن كل خوارزمية عودية لها خوارزمية تكرارية مكافئة (أكثر أو اقل تعقيداً) والعكس صحيح. [14],[9]

▪ خوارزمية المنطقية logical: يمكن النظر إلى الخوارزمية على أنها معالجة منطقية متحكم بها. هذا يعني أن الخوارزمية هي (algorithm=logic+control) حيث يعبر الجزء المنطقي عن البديهيات التي يمكن أن تستخدم في الحساب وأما جزء التحكم يحدد الطريقة التي تعالج بها تلك البديهيات. [8]

▪ خوارزمية تسلسلية serial، تفرعية parallel أو موزعة distributed: تتم مناقشة الخوارزميات عادة مع افتراض بأن الحواسيب تنفذ تعليمة من الخوارزمية في اللحظة ذاتها. تدعى هذه الحواسب أحياناً الحواسب التسلسلية، وتدعى الخوارزميات المصممة لمثل هذه الحواسيب باسم الخوارزميات التسلسلية. وبشكل معاكس نجد الخوارزميات التفرعية أو الموزعة، حيث تستفيد الخوارزميات التفرعية من الحواسيب التي تحوي بنيتها على العديد من المعالجات التي يمكن أن تستخدم لحل مسألة ما في اللحظة ذاتها. أما الخوارزميات الموزعة فإنها تستخدم العديد من الحواسيب المتصلة فيما بينها لتشكل شبكة. [8]

▪ خوارزمية حتمية deterministic أو غير حتمية non-deterministic: تقوم الخوارزميات الحتمية بحل المسألة بقرار دقيق في كل خطوة من الخوارزمية، في حين أن الخوارزمية غير الحتمية تحل المسألة من خلال تخمينات على الرغم من أن التخمينات النموذجية يمكن أن تكون أكثر دقة من خلال الإرشادات.

▪ خوارزمية دقيقة exact أو تقريبية approximate: في الوقت الذي تصل فيه العديد من الخوارزميات إلى حل دقيق، فإن الخوارزميات التقريبية تطرح تقريباً قريباً جداً من الحل الصحيح. يمكن أن يستخدم التقريب استراتيجية محددة أو عشوائية.

2 – التصنيف حسب أسلوب التصميم:

هناك العديد من الأساليب يختلف كل منها عن الآخر، كما أن كلاً من هذه الأصناف يمكن أن يحتوي على العديد من الخوارزميات، فيما يلي بعض من الأساليب المستخدمة:

▪ قسم وتابع divide and conquer: تقوم خوارزمية قسم وتابع بالتجزئة المتكررة للمسألة إلى مسألتين أصغر كل منهما مسألة شبيهة بالمسألة المطروحة إلى أن تصبح المسألة صغيرة بما يكفي لتحل ببساطة. [9]

▪ البرمجة الديناميكية dynamic programming: عندما تكون المسألة على شكل مجموعة من البنى الفرعية، أي أن الحل المثالي لمسألة ما يمكن بناؤه على شكل مجموعة من الحلول المثالية للمسائل الفرعية والمسائل الفرعية المتداخلة. [14]

▪ الطريقة الطماعة the greedy method: الخوارزمية الطماعة شبيهة بخوارزمية البرمجة الديناميكية، ولكن الفرق هو أن حل المسائل الفرعية لا يطلب أن يكون معروفاً في كل مرحلة. بدلاً من ذلك يتم إجراء خيار طماع لما يبدو أفضل في هذه اللحظة. حيث تعتمد هذه الخوارزمية على مبدأ الخيار الأفضل الممكن (وليس جميع الخيارات الممكنة).

▪ البرمجة الخطية linear programming: عند حل مسألة باستخدام البرمجة الخطية، توجد عدم مساواة inequalities تخص الدخل، وتتم محاولة تكبير (أو تصغير) بعض التوابع الخطية للدخل.

▪ التخفيض reduction: تتضمن هذه التقنية حل مسألة صعبة بتحويلها إلى مسألة أفضل معروفة، ونملك لها خوارزميات مقاربة مثالية. الهدف هو إيجاد خوارزمية تخفيض تعقيدها مسيطر عليه بالخوارزميات المخفضة الناتجة.

▪ البحث والتعداد search and enumeration: يمكن نمذجة العديد من المسائل على شكل مسائل حول المخططات graphs، تحدد خوارزمية استكشاف المخطط قواعد التجول عبر المخطط، وهي مفيدة لمثل هذه المسائل.

▪ الأسلوب الاحتمالي أو الإرشادي probabilistic and heuristic: الخوارزميات الاحتمالية هي تلك التي تأخذ خيارات عشوائية (أو شبه عشوائية) أما الخوارزميات الإرشادية في الخوارزميات التي لا يكون هدفها الرئيسي إيجاد حل مثالي بل حل تقريبي حيث زمن التنفيذ أو الموارد تكون محدودة.

3 – التصنيف حسب مجال الدراسة:

يملك كل مجال من مجالات العلوم مسائله الخاصة، ويفرض علينا خوارزمياته الفعالة، غالباً ما تتم دراسة المسائل المترابطة في مجال ما مع بعضها. من أمثلة هذه الخوارزميات خوارزميات البحث، خوارزميات الفرز، خوارزميات الدمج، الخوارزميات الرقمية، خوارزميات المخططات، خوارزميات السلاسل، خوارزميات التشفير، خوارزميات ضغط البيانات. .... وغيرها.

قد تتداخل المجالات مع بعضها، وبالتالي تستخدم خوارزمية من مجال ما في حل خوارزمية من مجال آخر.

4 – التصنيف حسب التعقيد:

يمكن تصنيف الخوارزميات بحسب مقدار الزمن اللازم لإنجازها بدلالة حجم الدخل، فبعضها يرتبط مع حجم الدخل بعلاقة خطية، وبعضها بعلاقة أسية. ... وهكذا. [3]

▪ **تصنيف خوارزميات الفرز:**

يمكن تصنيف خوارزميات الفرز المستخدمة في علم الحاسبات حسب ما يلي:

▪ التعقيد الحسابي: إذ تصنف إلى الخوارزميات ذات الأداء الأفضل best، الوسطي average أو الأسوأ worst ويعبر عنه بعدد المقارنات بدلالة حجم اللائحة.

▪ التعقيد الحسابي للتبديلات.

▪ استخدام الذاكرة.

▪ العودية.

▪ الاستقرار.

▪ اعتمادها على المقارنة.

▪ الطريقة العامة المستخدمة: الحشر insertion، التبديل exchange، الاختيار selection، الدمج merging. ... إلخ.

النتائج والمناقشة:

قمنا في هذا البحث بالتركيز على المعيار الأخير من معايير التصنيف، حيث قمنا بدراسته بشيء من التفصيل مستعرضين مجموعة من الخوارزميات من كل صنف مبتعدين عن الدراسة التفصيلية لجميع الخوارزميات – لأن المجال لا يتسع لمثل هذه الدراسة – إذ اكتفينا بدراسة تفصيلية لواحدة من الخوارزميات (من قبيل المثال) مبينين آلية عمل هذه الخوارزمية، كيفية حساب درجة تعقيدها، كيفية تقييم أدائها والوصول إلى النتائج اللازمة، فيما اكتفينا باستعراض النتائج بالنسبة لباقي الخوارزميات مركزين على آلية عمل هذه الخوارزميات، والتعرف على أدائها ومبتعدين عن الحديث عن التعبير عنها أو الشيفرة الزائفة pseudo-code الخاصة بها أو الشيفرة التي تعبر عنها باستخدام

إحدى لغات البرمجة ثم قمنا بتطبيق هذه الخوارزميات على المثال المستهدف ومقارنة أداء الخوارزميات المختارة واستخلاص النتائج والتوصيات.

1- الفرز بالتبديل Exchange sorts: نميز من هذا النوع من الفرز الخوارزميات التالية:

▪ خوارزمية الفرز الفقاعي Bubble sort algorithm: خوارزمية الفرز الفقاعي هي خوارزمية بسيطة تعتمد على القيام بالتجول عبر اللائحة المراد فرزها بشكل متكرر ومقارنة عنصرين في كل مرة وتبديل مواقعهما إذا كان ترتيبهما غير صحيح، ويستمر التجول إلى أن لا تعود هناك حاجة لإجراء أي تبديل الأمر الذي يعني أن اللائحة أصبحت مفروزة. جاءت تسمية الخوارزمية من حقيقة كون العناصر الصغيرة (الفقاعات) تقفز إلى أعلى اللائحة، وبما أن هذه الخوارزمية تعتمد على المقارنة للعناصر، فهي تعتبر من خوارزميات الفرز بالمقارنة.

يمكن تلخيص هذه الخوارزمية من خلال مقطع الشيفرة الزائفة pseudo-code التالي حتى يتسنى لنا حساب درجة تعقيدها : [5],[13]

```
BUBBLE SORT  ALGORITHM
(* Algorithm to sort the list A[1],...A[n]  in ascending order. *)
1. set scan equal to 1
2. while scan less or equal to n-1 do the following:
a. set pos  equal to 1.
b. while pos less or equal to n-1 do the following:
     b1. if A[pos] > A[pos+1] then do the following:
        (* swap A[pos] ←→ A[pos+1])
         b11. set  temp equal to A[pos]
         b12. set  A[pos] equal to A[pos+1]
         b13 set  A[pos+1] equal to temp
    b2.set pos equal to pos+1
c. set scan equal to scan+1
```

الجدول (3) التالي يبين عدد مرات تكرار كل عبارة من عبارات الخوارزمية المذكورة سابقاً في الحالة الأسوأ (أي عندما تكون بيانات الدخل مرتبة ترتيباً معاكساً لما هو مطلوب) :

statement	# of times executed
1	1
2	N
A	N-1
B	$N\times(N-1)$
b1	$(N-1)\times(N-1)$
b11	$(N-1)+(N-2)+....+2+1=\sum_{scan=1}^{scan=N-1}(N-scan)=\frac{(N)\times(N-1)}{2}$
b12	$(N-1)+(N-2)+....+2+1=\sum_{scan=1}^{scan=N-1}(N-scan)=\frac{(N)\times(N-1)}{2}$
b13	$(N-1)+(N-2)+....+2+1=\sum_{scan=1}^{scan=N-1}(N-scan)=\frac{(N)\times(N-1)}{2}$
b2	$(N-1)\times(N-1)$
c	N-1

total:	$\frac{9}{2}N^2 - \frac{7}{2}N + 1$

وعليه يمكن التعبير عن زمن التنفيذ باستخدام الترميز Big O notation كما يلي:

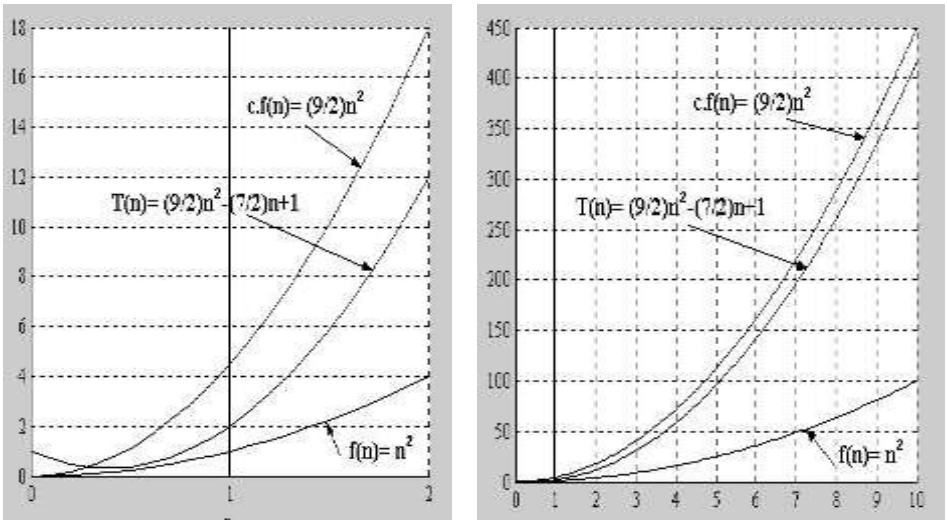
$$T(n) = \frac{9}{2}N^2 - \frac{7}{2}N + 1 \Rightarrow T(n) = O(n^2)$$

إذ أن:

$$\frac{9}{2}N^2 - \frac{7}{2}N + 1 \le \frac{9}{2}N^2 \; for N \ge 1$$

أي أن $f(N)=N^2$، $C=9/2$، $N_0=1$.

يبين الشكل (1) الخط البياني لكل من f(N)، T(N)، C.f(N): (تم استخدام matlab)



لاحظ أن 9/2.f(n) >= T(n) عندما n>= 1.

الشكل (1) يبين الخط البياني لكل من C.f(N),T(N),f(N).

- خوارزمية الفرز الكوكتيل cocktail sort algorithm: وتدعى أيضاً خوارزمية الفرز الفقاعي ثنائي الاتجاه bidirectional bubble sort، الفرز الهزاز الكوكتيل cocktail shaker sort، الفرز الهزاز shaker sort (وهي أحد أشكال الفرز بالاختيار selection sort)، الفرز بالموجة ripple sort، الفرز المكوكي shuttle sort التي هي نوع من الفرز الفقاعي (وهي أحد أشكال الفرزبالمقارنة). تختلف هذه الخوارزمية عن خوارزمية الفرز الفقاعي في أن عملية الفرز تتم في الاتجاهين في كل عبور للائحة. إن تحقيق هذه الخوارزمية أكثر تعقيداً من خوارزمية الفرز الفقاعي. ففي العبور الأول بالاتجاه اليميني تتم إزاحة العنصر الأكبر إلى مكانه السليم في نهاية اللائحة، وفي العبور التالي بالاتجاه اليساري تتم إزاحة العنصر الأصغر إلى مكانه السليم في بداية اللائحة وهكذا. بعد n تجوال يكون n عنصراً الواقعين في بداية اللائحة n عنصراً الواقعين في نهاية اللائحة في مواقعها الصحيحة. إن درجة تعقيد هذه الخوارزمية هي من الرتبة $O(n^2)$ في كل من الحالتين الأسوأ والوسطية، ولكنها تصبح من الرتبة O(n) في حال كانت اللائحة في الحالة الابتدائية مرتبة نوعاً ما.

▪ الفرز المشطي comb sort algorithm : إن خوارزمية الفرز المشطي هي خوارزمية بسيطة نسبياً، وهي عبارة عن نوع من التحسين لخوارزمية الفرز الفقاعي وتنافس من حيث السرعة عدداً من الخوارزميات المعقدة كخوارزمية الفرز السريع quicksort. الفكرة الأساسية تقوم على حذف القيم الصغيرة القريبة من نهاية اللائحة، حيث إن هذه القيم لدى العمل بالفرز الفقاعي تبطِّىء عملية الفرز بشكل ملحوظ. في الفرز الفقاعي تكون المسافة بين العنصرين المقارنين مساوية للقيمة 1 بينما في الفرز المشطي تكون المسافة أكبر من ذلك تبدأ قيمة هذه المسافة من المقدار المحسوب من ناتج تقسيم طول اللائحة على معامل الانكماش shrink factor والمساوي تقريباً للقيمة 1.3 ($1/(1-\frac{1}{e^{\varphi}}) \approx 1.247330950103979$) ثم تفرز اللائحة، بعد تذلك تقسم المسافة على معامل الانكماش من جديد ثم تفرز، وهكذا إلى أن تصبح هذه المسافة مساوية للقيمة 1 وبذلك تصبح اللائحة مفروزة.

▪ الفرز التمثالي Gnome sort algorithm: هذه الخوارزمية شبيهة بشكل كبير لخوارزمية الفرز بالحشر insertion sort إلا أن نقل عنصر إلى مكانه المناسب تتم من خلال سلسلة من التبديلات كما في حالة الفرز الفقاعي، تقوم هذه الخوارزمية دائماً بإيجاد أول مكان يكون فيه عنصران متجاوران موجودان بترتيب خاص إذ تقوم بتبديلهما مستفيدة من حقيقة أن إجراء التبديل، يمكن أن ينشئ قيماً متجاورة جديدة غير مرتبة، ثم يقوم بتبديلهما، فتتم بعدها عملية الاختبار مباشرة بعد إجراء كل عملية ترتيب. درجة تعقيد هذه الخوارزمية هي من الرتبة $O(n^2)$ ولكنها في الحقيقة تنجز عملها أسرع من ذلك.

▪ خوارزمية الفرز السريع Quicksort algorithm: وهي خوارزمية شهيرة جداً وتعتمد أسلوب قسم وتابع divide and conquer لتجزئة اللائحة إلى لائحتين جزئيتين صغيرتين، ومن ثم يتم اختيار عنصر ما من اللائحة ومن ثم ترتيب اللائحة بشكل يكون معه جميع العناصر الأصغر من هذا العنصر تأتي قبله والعناصر الأكبر تأتي بعده ومن ثم تقسيم اللائحة إلى لائحتين تحوي إحداهما العناصر الأصغر والأخرى العناصر الأكبر وفرزهما وهكذا. إن درجة تعقيد هذه الخوارزمية هي من الرتبة $O(n\log n)$ في الحالة الوسطية و $O(n^2)$ في الحالة الأسوأ. إن هذه الخوارزمية أسرع في الأداء من الخوارزميات الأخرى ذات درجة التعقيد $O(n\log n)$. [4]

2- الفرز بالاختيار Selection sorts: نميز من هذا النوع من الفرز الخوارزميات التالية:

▪ خوارزمية الفرز بالاختيار selection sort algorithm: وهي عبارة عن خوارزمية فرز بالمقارنة في المكان in-place comparison وتقوم هذه الخوارزمية بإيجاد أصغر عنصر في اللائحة ثم التبديل بينه وبين القيمة الموجودة في بداية اللائحة، تم تكرار هذه العملية على الجزء المتبقي من اللائحة، عملياً هذه الخوارزمية تقوم بتقسيم لائحة قيم الدخل إلى لائحتين الأولى للقيم المرتبة والثانية للقيم المتبقية المراد ترتيبها. أما درجة تعقيد هذه الخوارزمية، فهي من الرتبة $O(n^2)$ مما يجعلها غير فعالة في حال اللوائح الضخمة.

▪ خوارزمية الفرز بالكومة Heapsort algorithm: هذه الخوارزمية قائمة على المقارنة، وهي جزء من مجموعة خوارزميات الفرز بالاختيار، علماً أنها أبطأ بشكل كبير من أغلب الخوارزميات مثل خوارزمية الفرز السريع، وهذه الخوارزمية هي من خوارزميات التبديل بالمكان. تعتمد هذه الخوارزمية على حشر عناصر سلسلة دخل إلى بنية معطيات من النوع كومة heap حيث إنه وضع أكبر قيمة في أعلى الكومة أو أصغر قيمة في أسفل الكومة، وتستمر عملية انتزاع القيم إلى أن لا يبقى هناك قيم حيث تكون القيمة قد انتزعت بشكل مرتب. درجة تعقيد هذه الخوارزمية هي من الرتبة $O(n\log n)$. [10],[12]

▪ خوارزمية الفرز الساحلي strand sort algorithm: تعتمد هذه الخوارزمية على السحب بشكل متكرر لسلاسل جزئية من اللائحة ودمجها في مصفوفة واحدة، إذ تم في كل تكرار سحب سلسلة من العناصر المرتبة أصلاً ودمج هذه السلاسل معاً. درجة تعقيد هذه الخوارزمية هي من الرتبة $O(n \log n)$ في الحالة الوسطية، ومن الرتبة $O(n)$ في الحالة المثلى و $O(n^2)$ في الحالة الأسوأ.

3- الفرز بالحشر Isertion sorts: نميز من هذا النوع من الفرز الخوارزميات التالية: [11]

▪ خوارزمية الفرز بالحشر insertion sort algorithm: وهي عبارة عن خوارزمية فرز بسيطة جداً، تعتمد مبدأ الفرز بالمقارنة. في هذه الخوارزمية يتم حذف عنصر من لائحة الدخل في كل تكرار وحشره في الموقع المناسب من سلسلة مفروزة إلى أن لا يبق هناك أي عنصر في لائحة الدخل، وتكون عملية اختيار العنصر الذي ستتم إزالته من لائحة الدخل عشوائية. إن هذه الخوارزمية هي عبارة عن خوارزمية فرز بالمكان. ومن جهة الفعالية فإن هذه الخوارزمية أقل فعالية على اللوائح الكبيرة من باقي الخوارزميات، ولكنها تتميز بالعديد من المزايا أهمها بساطة التحقيق والفعالية من أجل حجوم البيانات الصغيرة، حيث إن فعاليتها في الحالة المثلى هي من رتبة $O(n)$ أما في الحالة الأسوأ والحالة الوسطية، فإن فعاليتها (درجة تعقيدها) من رتبة $O(n^2)$، وهذه الخوارزمية تكون فعالة عند إدخال القيم من لوحة المفاتيح، وفي هذه الحالة وباستخدام لوائح التخزين الديناميكية المؤلفة من عقد سجلات أو صفوف، والتي تخلصنا من الإزاحة اللازمة لحشر العنصر في مكانة وللاستفادة من زمن الإدخال سيتم حشر العنصر قبل إدخال العنصر التالي، وهذا يذكر بعمل المفسر أي عند الانتهاء من إدخال آخر عنصر ستكون اللائحة مرتبة، ولكن ذلك ينعكس على حجم البيانات بشكل سلبي.

▪ خوارزمية الفرز الصدفي shell sort algorithm: وهي عبارة عن خوارزمية فرز تمثل تعميماً لخوارزمية الفرز بالحشر مع ملاحظة أن الفرز بالحشر يعتبر فعالاً إذا كانت سلسلة الدخل مرتبة بنسبة كبيرة، كما أن الفرز بالحشر عموماً يعتبر خوارزمية غير فعالة، لأنها لا تقوم بتحريك العنصر سوى بمقدار موقع واحد في كل مرة. قامت خوارزمية الفرز الصدفي بتحسين خوارزمية الفرز بالحشر بمقارنة عناصر غير متجاورة مباشرة الأمر الذي يقلل من عمليات المقارنة اللازمة لنقل عنصر ما إلى موقعه الصحيح. تبلغ درجة تعقيد هذه الخوارزمية الرتبة $O(n)$ في الحالة الأسوأ في حين أنها من الرتبة $O(n\log^2 n)$ في الحالات الأخرى.

▪ هذا بالإضافة إلى أصناف أخرى أقل شيوعاً من الفرز بالحشر مثل الفرز الشجري tree sort، الفرز المكتبي library sort. .. وغيرها.

4- الفرز بالدمج Merge sorts:

وهي عبارة عن خورازمية فرز قائمة على إجراء $O(n \log n)$ مقارنة، وهي خوارزمية مستقرة الأمر الذي يعني أنها تحافظ على ترتيب الدخل للعناصر المتساوية في الخرج المفروز، وتعتمد على مبدأ قسم وتابع في الفرز. تعمل هذه الخوارزمية كما يلي: يتم تقسيم اللائحة إلى لائحتين متساويتين تقريباً بالطول، ومن ثم تقسيم كل لائحة جزئية إلى لائحتين جزئيتين أيضا، ويستمر التقسيم إلى أن نصل إلى لوائح بطول عنصر واحد، وبالتالي تكون اللائحة مفروزة، ثم يتم إعادة دمج اللائحتين الجزئيتين بشكل مرتب، وهكذا إلى أن يتم إعادة تشكيل لائحة الدخل نفسها، ولكن بشكل مفروز. يتمثل التحسين الذي تقدمه هذه الخوارزمية في الفكرتين التاليتين: إن فرز اللائحة الصغيرة يتطلب وقتاً أصغر من الكبيرة، كما أن تشكيل لائحة مفروزة من لائحتين مفروزتين يعتبر أسرع من تشكيل لائحة مفروزة من لائحتين غير مفروزتين. [1]

5- الفرز الذي لا يعتمد على المقارنة Non-comparison sorts: نميز من هذا النوع من الفرز الخوارزميات التالية:

- خوارزمية الفرز الأساس radix sort algorithm: وهي عبارة عن خوارزمية فرز تقوم بفرز الأعداد الصحيحة من خلال معالجة خاناتها خانة خانة. تقوم أغلب الحواسيب الرقيمة بتمثيل بياناتها بشكل ثنائي، وبالتالي فإن معالجة خانات العدد الصحيح هي عبارة عن معالجة مجموعات من الخانات الثنائية، حيث تتم عملية المقارنة على أحد طريقتين، إما بمقارنة الخانة الأقل أهمية Less Significant Digit وإما الخانة الأكثر أهمية Most Significant Digit. في حالة الفرز الأساس بمقارنة الخانة الأقل أهمية LSD قد تكون المفاتيح سلسلة من المحارف والخانات الرقمية بحسب أساس معين، تبدأ معالجة المفاتيح بدءاً من الخانة ذات الأهمية الدنيا (الخانة في أقصى اليمين)، وتنتقل إلى الخانة الأكثر أهمية (في أقصى اليسار)، ويكون تتالي الخانات التي تتم معالجتها في حالة الفرز بحسب الخانة الأقل أهمية معاكساً لتالي الخانات التي تتم معالجتها بحالة الفرز وفق الخانة الأكثر أهمية. الحد الأقصى لزمن التنفيذ هو من رتبة O(n).
- خوارزمية الفرز الصناديقي bucket sort algorithm: تقوم هذه الخوارزمية بتجزئة المصفوفة إلى عدد منته من الصناديق يتم فرز كل منها بعدئذ بشكل منفصل باستخدام أي من خوارزميات الفرز، أو بالاستخدام المتكرر لخوارزمية الفرز الصناديقي. تعمل خوارزمية الفرز الصناديقي كما يلي: تتم تهيئة مصفوفة فارغة على شكل صندوق فارغ، ثم يتم التجول عبر المصفوفة الأصلية ووضع كل عنصر في صندوقه الخاص، ومن ثم فرز كل صندوق غير فارغ وإعادة العناصر من الصناديق غير الفارغة إلى المصفوفة الأصلية. درجة تعقيد خوارزمية الفرز وهي من رتبة O(n).
- هذا بالإضافة إلى مجموعة من الخوارزميات الأقل شيوعاً مثل counting sort، pigeonhole sort، tally sort.

6- أشكال أخرى: ومن أمثلتها Topological sort، Network sort.

الاستنتاجات والتوصيات:

يبين الجدول (4) تلخيصاً للنتائج التي توصلنا إليها من خلال الدراسة المبينة أعلاه يتضمن الحالة الوسطية والحالة الأسوأ لكل من الخوارزميات التي تمت دراستها حيث n هو عدد العناصر المراد فرزها.

Name	Average	Worst	Memory	Stable	Method	Other notes
Bubble sort	—	$O(n^2)$	O(1)	Yes	Exchanging	
Cocktail sort	—	$O(n^2)$	O(1)	Yes	Exchanging	
Comb sort	O (n log n)	O(n log n)	O(1)	No	Exchanging	Small code size
Gnome sort	—	$O(n^2)$	O(1)	Yes	Exchanging	
Selection sort	$O(n^2)$	$O(n^2)$	O(1)	No	Selection	Can be implemented as a stable sort
Insertion sort	O(n + d)	$O(n^2)$	O(1)	Yes	Insertion	d is the number of inversions, which is $O(n^2)$
Shell sort	—	$O(n \log^2 n)$	O(1)	No	Insertion	
tree sort	O(n log n)	O(n log n)	O(n)	Yes	Insertion	When using a self-balancing binary search tree
Library sort	O(n log n)	$O(n^2)$	O(n)	Yes	Insertion	
Merge sort	O(n log n)	O(n log n)	O(n)	Yes	Merging	
Heapsort	O(n log n)	O(n log n)	O(1)	No	Selection	
Quicksort	O(n log n)	$O(n^2)$	O (log n)	No	Exchanging	Naïve variants use O(n) space; can be O(n log n) worst case if median pivot is used
Patience sort	—	$O(n^2)$	O(n)	No	Insertion	Finds all the longest increasing subsequences within O(n log n)
Strand sort	O(n log n)	$O(n^2)$	O(n)	Yes	Selection	
Pigeonhole sort	O(n+2k)	O(n+2k)	O(2k)	Yes	Non comparison	
Bucket sort	O(n·k)	$O(n^2 \cdot k)$	O(n·k)	Yes	Non comparison	Assumes uniform distribution of elements from the domain in the array.
Counting sort	O(n+2k)	O(n+2k)	O(n+2k)	Yes	Non comparison	
LSD Radix sort	O(n·k/s)	O(n·k/s)	O(n)	Yes	Non comparison	
MSD Radix sort	O(n·k/s)	O(n·(k/s)·2s)	O ((k/s)·2s)	No	Non comparison	

لاحظنا أن درجات التعقيد التي صادفناها للخوارزميات المدروسة هي من الرتب التالية:

$O(1)$, $O(n)$, $O(\log(n))$, $O(n \log(n))$, $O(\log(\log(n)))$, $O(kn)$,$O(n^2)$…….

حيث إن $O(1)$ تدل على ثبات الخوارزمية وعدم اعتمادها على حجم الدخل و $O(n)$ تدل على درجة تعقيد الخوارزمية الخطية و $O(n^2)$ تدل على درجة تعقيد مربعة و $O(kn)$ تدل على درجة التعقيد الأسية. …

والجدول (5) يعطي مقارنة عددية بين قيم التوابع المختلفة لهذه الدرجات وفقاً لقيم n مختلفة:

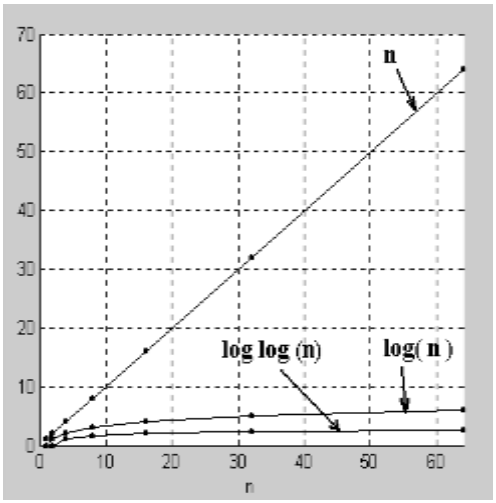
log log n	log n	n	n log n	n^2

-	0	1	0	1
0	1	2	2	4
1	2	4	8	16
1.85	3	8	24	64
2	4	16	64	256
2.32	5	32	160	1024
2.6	6	64	384	4096
3	8	256	$2.05x10^3$	$6.55x10^4$
3.32	10	1024	$1.02x10^4$	$1.05x10^{12}$
4.32	20	1048576	$2.1x10^7$	$1.1x10^{12}$

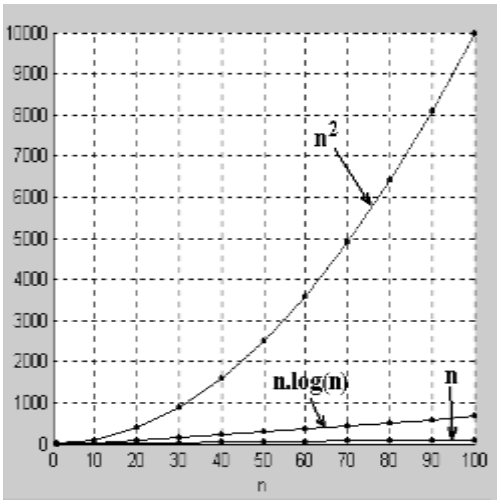
ويبين الشكل(2 a,b,c,d) مقارنات بيانية بين درجات التعقيد المختلفة حيث:

الشكل(2 –a) مقارنة (log log n, log n, n). الشكل (2 –b) مقارنة ($n, n \log n, n^2$).

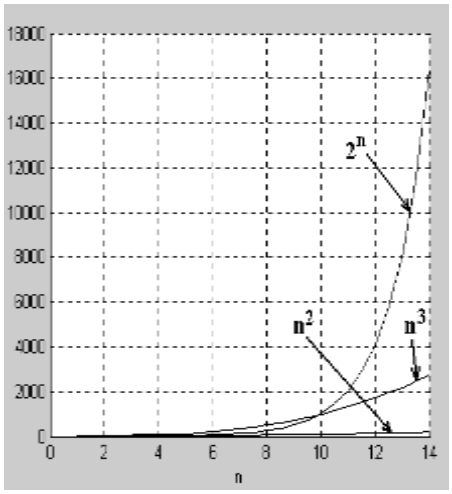
الشكل(2 –c) مقارنة ($n^2, n^3, 2^n$). الشكل(2 –d) فيظهر الدرجات جميعها معاً:



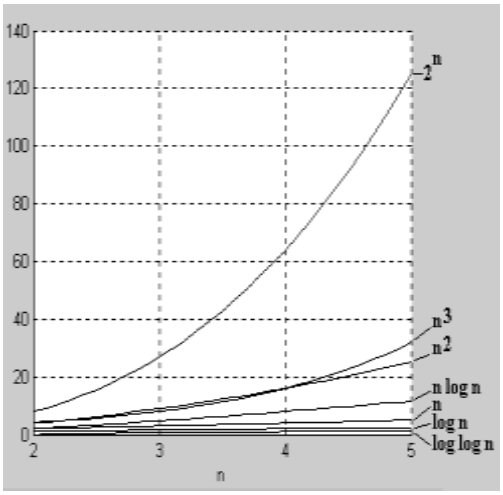
(a) مقارنة (log log n, log n, n).



(b) مقارنة ($n, n \log n, n^2$).



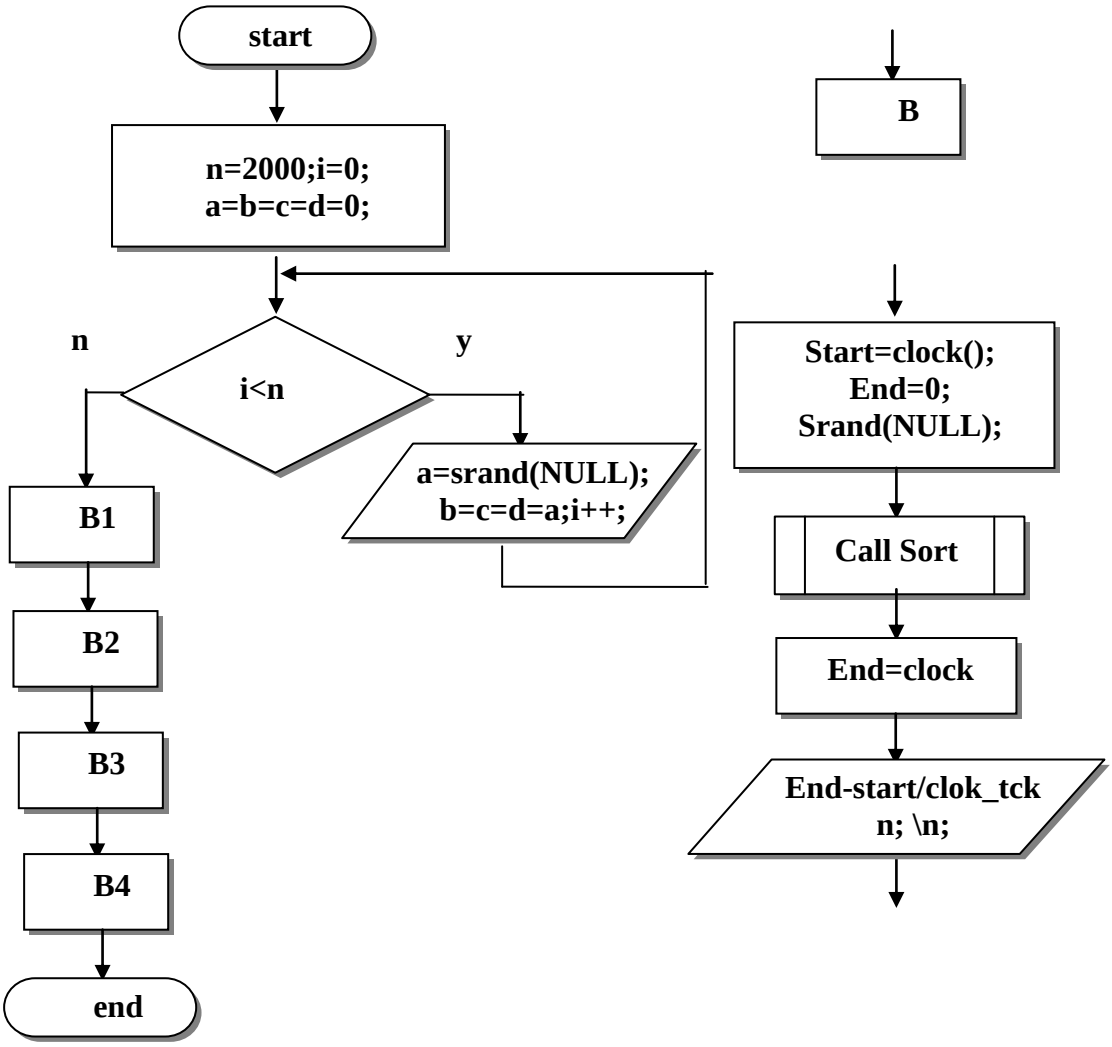
(c) مقارنة ($n^2, n^3, 2^n$)



(d)يمثل جميع الحالات

الشكل (2) مقارنة بيانية بين درجات التعقيد المختلفة.

وكتطبيق عملي قمنا بتنفيذ الخوارزميات الأربعة التالية Quick, merge, bubble, insert على مصفوفة من القيم الصحيحة العشوائية واختبارها من أجل عدة قيم مختلفة وذلك على حاسب يملك المواصفات التالية processor Celeron 1.60GHz, RAM 256MB. وفق المخطط (1):

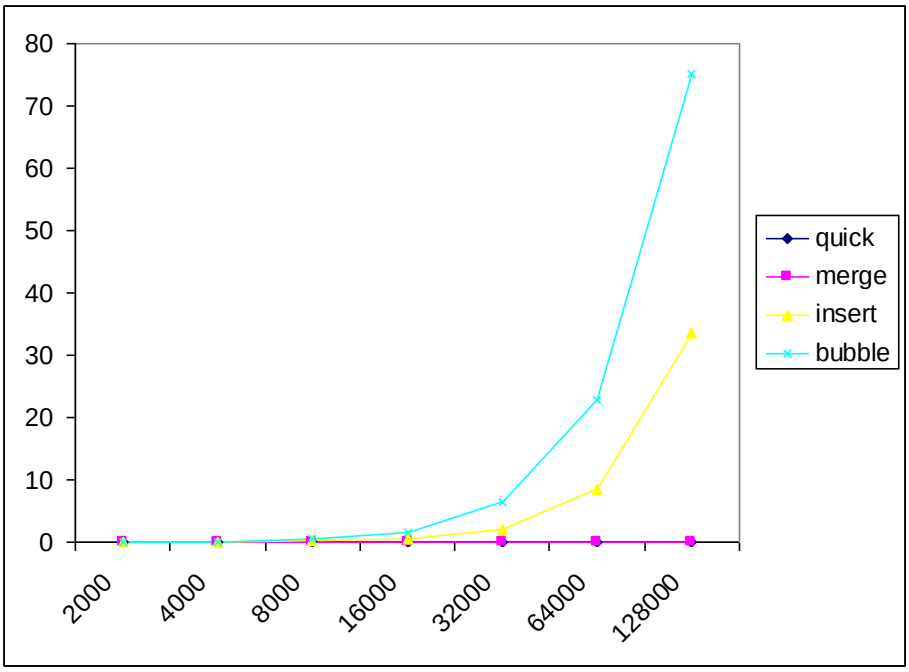


المخطط (1) يبين المخطط التدفقي للخوارزمية التي اعتمدت في الحساب حيث إن a=asai[n], b=asaq[n], c=asam[n], d=asal[n] والتي استخدم كل منها مع نداء التابع الفرعي للخوارزميات المشروحة والممثلة B1, B2, B3, B4 والتي رسم المخطط الانسيابي لها في الشكل اليميني.

فكانت النتائج كما في الجدول (6)يبين القيم الزمنية الناتجة عن تنفيذ الخوارزميات الأربعة مقدرة بالميلي ثانية

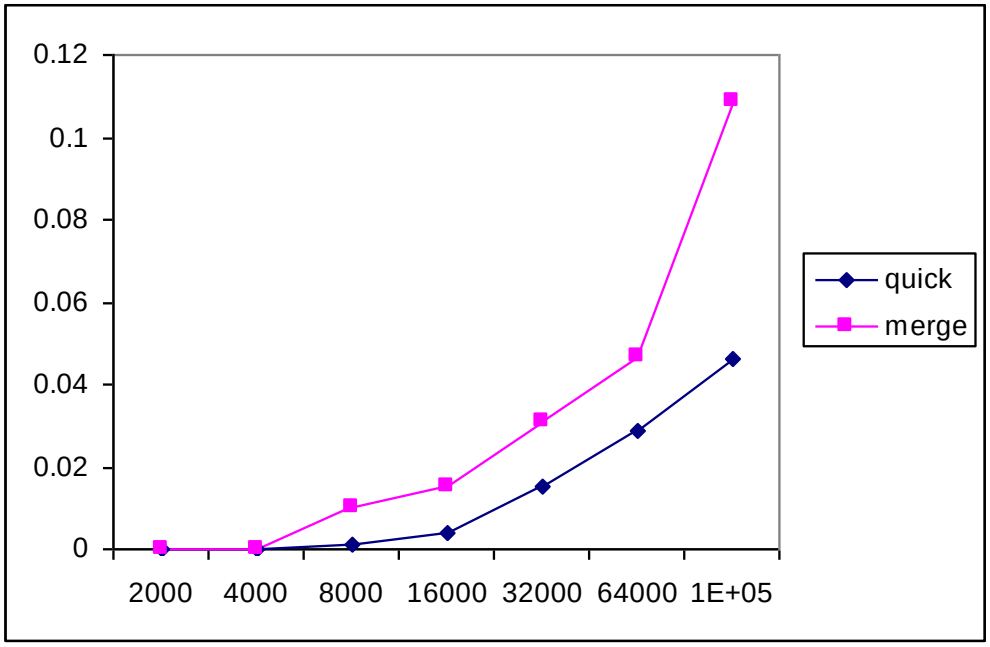
N	quick	merge	Insert	Bubble
2000	0	0	0.015	0.031
4000	0	0	0.031	0.109
8000	0.001	0.01	0.135	0.425
16000	0.004	0.015	0.54	1.625
32000	0.015	0.031	2.109	6.5
64000	0.029	0.047	8.515	22.89
128000	0.046	0.109	33.59	75.2

قمنا باستخدام أداة رسم المخططات البيانية المرفقة بالبرنامج Microsoft Excel 2003 فكانت الأشكال التالية: الشكل (3) يوضح أداء الخوارزميات الأربعة المبينة في الجدول (6).



الشكل (3) يوضح أداء الخوارزميات الأربعة

ونظراً لتقارب القيم وصغرها في خوارزميتي quick,merge مقارنة بالخوارزميتين الأخريين قمنا برسم مخطط خاص بهما فكان الشكل (4) :



الشكل (4) يوضح أداء الخوارزميتين quick, merge

كما ونفذنا هذا العمل على حاسبين بالمواصفات التالية:

processor Intel core™ 2 1.60GHz, RAM 1GB.
processor Penetum D 3.0GHz, RAM 1GB.

وكانت النتائج منسجمة مع الحاسب الأول من حيث علاقة الخوارزميات مع بعضها بعض.

بمعاينة الشكلين وجدنا أن:

– الخوارزميات quick,merge أكثر فعالية بشكل واضح من الخوارزميات insert,bubble وهذا ما يتوافق مع الدراسة النظرية.

– تتطلب خوارزمية merge استخدام مصفوفة مساعدة وعدد مرات تجول أكبر من خوارزمية quick مما يجعل خوارزمية quick تتفوق في الأداء كما زاد عدد العناصر.

ومن الواضح أن الخوارزميات التي تملك درجة تعقيد أسية تستخدم فقط عندما تكون قيم الدخل فيها صغيرة وذلك لصعوبة تحقيقها عمليا بسبب الزمن الكبير اللازم للتنفيذ.

يوصي البحث باعتماد إحدى الخوارزميات ذات درجة التعقيد اللوغاريتمية ومن أمثلتها الفرز المشطي، Shell sort، tree sort، Merge sort، Heapsort علماً أن هذه الخوارزميات وعلى الرغم من كونها تملك درجات تعقيد متشابهة في الحالة الأسوأ إلا أنها في الحقيقة غير متساوية في الأداء، إذ يختلف أداؤها باختلاف بنية المعطيات المستخدمة في تخزين البيانات المراد فرزها (وهنا لا يتسع المجال لكافة هذه المقارنات).

وعلى الرغم من كون حجم البيانات في قاعدة البيانات الخاصة بمزود خدمة الانترنت كبير جداً إلا أن عملية الفرز في قاعدة البيانات هي عملية فرز للفهارس indexes وهي عبارة عن قيم صحيحة، وبالتالي يمكن الاستفادة من التطبيق العملي السابق على قاعدة البيانات هذه، كما ويمكن تعميمه على أية قاعدة بيانات معقدة.

المراجع:

1- N. WIRTH, *Algorithms and Data Structures*, New York; Springer-Verlag,) August 2004

2- LARRY NYHOFF. *C++ An introduction to Data Structures*. Hall U.K. Limited. London 1998.
3- LARRY NYHOFF. *Data Structures in Pascal*. Hall U.K. Limited. London 1991.
4- BY H. M. DEITEL - DEITEL & ASSOCIATES, INC., P. J. DEITEL - DEITEL & ASSOCIATES, INC. *Java™ How to Program,* Sixth Edition, Prentice Hall August 04, 2004
5-, P. J. DEITEL - DEITEL & ASSOCIATES, INC. *C++ ™ How to Program*, Fifth Edition, Prentice Hall 2004.
6 TANTALO ASYMP, *Introduction to Analysis of Algorithms,* Fall 2003
7-LAWRENCE C PAULSON *Foundations of Computer Science*, Computer Laboratory, University of Cambridge, Copyright c ° 2000.
8- IAN PARBERRY, *parallel complexity Theory*, Pitman, London 1997.
9-XING HUANG, BO HUANG, description of program for Pennysort, School of Software, Tsinghua University, Beijing 100084, China, March 30, 2006
10-INA PARBERRY, *Lecture Notes on Algorithm Analysis and Computational Complexity (Fourth Edition)* Computer Science, University of North Texas, December 2001.
11-THOMAS NIEMANN, *Sorting and Searching Algorithms*, McGraw-Hill, New York 2001.
12-Additional Laboratory in ADA TDDB 57, *Data structures and algorithms*, Fall September 2006.
13- NSF GRANTS, Bubble *Sort*: *An Archaeological Algorithmic Analysis*, Computer Science Department, Duke University, 2000.
14- LAWRENCE C PAULSON *Foundations of Computer Science*, Computer Laboratory, University of Cambridge, Copyright c ° 2000.