المحاضرة الأولى

تصميم وتحليل الخوارزميات
Design and Analysis of Algorithms

المصادر المعتمدة:

1- Computer Algorithms/ C++ , by Ellis Horowitz, 1997.
2- Algorithms Design Techniques and Analysis, by M. H. Alsuwaivel, 1999.

مقدمة

في حقل علوم الحاسبات، يمكن في الغالب حل مسألة ما باستخدام عدد من الطرق المختلفة بعضها يكون أكفأ من الاخر. لهذا السبب يعد موضوع تصميم وتحليل الخوارزميات اساسيا في هذا الحقل. حتى ان بعض العلماء ذهبوا الى تعريف علوم الحاسبات بانها علوم دراسة الخوارزميات، وهذا ليس بالشيء الغريب بسبب اعتماد كل مجال في علوم الحاسبات اعتمادا كبيرا على تصميم خوارزميات فعالة. وما انظمة التشغيل او المؤلفات الا تطبيقات مباشرة لخوارزميات خاصة الاغراض.

الخوارزمية/ هي مجموعة محددة من التعليمات (خطوات الحل) التي تؤدي إلى انجاز مهمة معينة ويجب ان تتوافر فيها الشروط التالية :-

١- المدخلات (Inputs) :- صفر أو أكثر من القيم.
٢- المخرجات (Outputs) :- قيمة واحدة على الأقل.
٣- الوضوح (Definiteness) :- كل خطوة فيها واضحة المعاني وغير غامضة (يجب ان تفهم من قبل الناس جميعا)
مثال / العبارة "Add 6 or 7 to x" غير مسموح بها في الخوارزمية لأنها عبارة غير واضحة.
٤- المحدودية (Finiteness) :- كل خطوات الخوارزمية يمكن حلها في فترة زمنية محددة.
مثال / العبارة " قسم الرقم ١٠ على ٣ بدقة كاملة " غير محددة ويجب أن لا يسمح بها داخل الخوارزمية.
٥- المحلولية (Effectiveness) :- كل خطوة اولية ممكنة الحل.

الفرق بين الخوارزمية والبرنامج

في النظرية الاحتسابية هناك فرق بين الخوارزمية والبرنامج. ففي الخوارزمية يجب توفر الشروط الخمسة ويمكن وصفها بطرق عديدة (لغة طبيعية مع التأكيد على شرط الوضوح، لغة خوارزمية (pseudocode)، مخططات انسيابية (flowcharts)) . بينما يمكن في البرنامج عدم تحقق الشرط الرابع (مثلا نظام التشغيل) ويوصف البرنامج بلغة الحاسوب.

ملاحظة / البرنامج = خوارزمية + هيكل بياني(طريقة لتنظيم البيانات).

خطوات تطوير البرامج

تمر عملية تطوير البرامج بخمس خطوات رئيسية:

١- توصيف المتطلبات (Requirements Specification) : تحديد (فهم) المدخلات والمخرجات

٢- التصميم (Design) : تحديد العمليات الرئيسية التي تطبق على كل كيان بياني وافتراض وجود اجهزة معالجة لتنفيذ هذه العمليات.

٣- التحليل (Analysis) : المفاضلة بين الخوارزميات المتوفرة لحل نفس المسألة تبعا لمقاييس مفاضلة متفق عليها (تعقيدات الوقت، تعقيدات الخزن) لاختيار أفضلها.

٤- تحسين وترميز (Refinement & coding) : في هذه الخطوة يتم تحديد التمثيل البياني لكل كيان ثم كتابة إجراءات لكل عملية على تلك الكيانات وتكوين نسخة متكاملة للبرنامج.

٥- التحقق من الصلاحية (Verification) : تتضمن هذه الخطوة ثلاث جوانب مختلفة :-

أ) البرهنة على الصحة (Proving):- يجب إثبات ان البرنامج صحيح قبل استخدامه، حيث يتم استخدام طرق معينة للبرهنة على الصحة.

ب) الاختبار (Testing) :- هي عملية توليد نماذج بيانية يعمل عليها البرنامج حيث إن الهدف منها هو إعطاء إشارة على وجود أخطاء في البرنامج.

جـ) تشخيص الأخطاء (Debugging) :- هي عملية تحديد مواقع الأخطاء في البرنامج وتصحيح الايعازات التي سببت تلك الأخطاء.

تحليل الخوارزميات (Algorithms Analysis)

يقصد بالتحليل هنا تقييم الخوارزمية ومقارنتها مع الأخريات حيث يوجد مقياسان مرتبطان مباشرة بانجازيه الخوارزمية هما :-

أ) تعقيدات الخزن (Space Complexity) : هي كمية الذاكرة التي يتطلبها تشغيل البرنامج حتى اكتماله، حيث يعتمد هذا الخزن على جزئين :-

١- جزء ثابت مستقل عن خصائص المدخلات والمخرجات، حيث يمثل هذا الجزء الخزن الخاص بالتعليمات (Code space) .

٢- جزء متغير يتألف من الخزن الذي يتطلبه البرنامج للمتغيرات المركبة والذي يعتمد حجمها على مثال المسألة المراد حله إضافة إلى الخزن الخاص بالمكدس المستخدم في التداخل (Recursion) وعليه يمكن صياغة تعقيدات الخزن S(P) للبرنامج P كالتالي:-

$$S(P)=Const. + S_P \text{ (خصائص المثال)}$$

Program Code	Java Stack	→ Free Memory ←	Memory heap

صورة الذاكرة للبرنامج في Java Virtual Machine(JVM)

الاسباب الداعية الى الاهتمام بحساب تعقيدات الخزن للبرنامج

١- اذا كان البرنامج يراد تنفيذه على نظام حاسب متعدد المستعملين (Multi-user)، قد يكون علينا في هذه الحالة تحديد كمية الذاكرة المراد تخصيصها للبرنامج.
٢- يكون من المرغوب سلفا لأي نظام حاسوبي معرفة فيما إذا كانت هناك ذاكرة كافية لتنفيذ البرنامج.
٣- قد تمتلك المسألة العديد من الحلول الممكنة بمتطلبات خزن مختلفة، الاختيار الأفضل يكون للحل الذي يعطي متطلبات خزن اقل.
٤- يمكن استعمال تعقيدات الخزن لتخمين حجم اكبر مسألة يمكن حلها من خلال البرنامج.

ب) تعقيدات الوقت (Time Complexity) : هي كمية الوقت التي يتطلبها تشغيل البرنامج حتى اكتماله ويتألف من حاصل جمع وقتي التأليف (Compilation) وتشغيل البرنامج (Running) ويمكن صياغته كالتالي :-

$T(P)=Const. + T_P$ (خصائص المثال)

الاسباب الداعية الى الاهتمام بحساب تعقيدات الوقت للبرنامج

١- بعض أنظمة الحاسبات تطلب من المستعمل توفير حد أعلى على كمية الوقت التي يحتاجها البرنامج للتنفيذ وحالما يصل البرنامج إلى هذا الحد يتم إخراجه من دائرة التنفيذ.
٢- قد يكون على البرنامج الذي يتم تطويره توفير استجابات وقت حقيقي مرضي.
٣- إذا كان هناك عدة بدائل لحل مسألة ما فان القرار يعتمد بصورة مبدئية على اختلاف الأداء المتوقع بين تلك الحلول.