

مقدمة في الخوارزميات باستخدام NET.

يكاد لا يستغني أي عمل بسيط نقوم به في حياتنا اليومية من استخدام خوارزمية ما، قد تكون واضحة وقد تكون ضمنية حتى أننا لا نشعر بوجودها. فالخوارزمية هي ببساطة الخطوات التي نقوم بها من أجل حل أي مشكلة ما. لنأخذ مثالا عمليا. ماهي الخطوات اللازمة لإرسال إيميل لصديق ما؟ بالطبع إن أحد لا يملك خوارزمية جاهزة من أجل إرسال إيميل مكتوبة على ورقة أو في كتاب. لكننا في حقيقة نستخدم خوارزمية ما نقوم باتباعها بشكل طبيعي ودون أن ننتبه. فمثلا يمكننا كتابة خوارزمية إرسال الإيميل على الشكل التالي :

- 1- شغل الكمبيوتر.
- 2- افتح برنامج الاوت لوك.
- 3- اكتب نص الإيميل.
- 4- اختر عنوانا مناسباً له واختر اسم المرسل إليه.
- 5- اضغط على زر الإرسال.

في علم الكمبيوتر سوف نكون ينصب اهتماما على الخوارزميات القابلة للتنفيذ بواسطة الكمبيوتر. أو بمعنى آخر سوف تكون مهمتنا تصميم الخوارزمية الملائمة من أجل المشكلة المطروحة وذلك بوصف الخطوات الواجب تنفيذها على الكمبيوتر للحصول على النتيجة المطلوبة **بأسرع وقت وبأقل تكلفة**.

حسنا ما المقصود بأسرع وقت؟ المقصود هنا بأقل عدد ممكن من العمليات الرياضية والحسابية أو بعملية يتعلق عددها وسرعتها بالبيانات الداخلة للخوارزمية والتي تتطلب وقتا من الكمبيوتر (حتى في حالة كون الوقت اللازم للكمبيوتر كي يقوم بعملية ما صغير جدا إلا أننا هنا نتحدث عن معطيات كبيرة جدا - مثلا ترتيب مليون عدد ترتيبا تصاعديا)

لنفرض أننا نريد ترتيب n عددا ترتيبا تصاعديا وإن الخوارزمية المستخدمة تحتاج إلى إجراء n^2 عملية مقارنة. وعلى أن فرض أن كل عملية مقارنة تكلف فقط جزء من الألف من الثانية فمن أجل ترتيب 1000 عددا سوف نحتاج إلى 1000000 ثانية أي تقريبا 17 دقيقة وهو وقت طويل نسبيا. فكيف سيكون الوضع إذا كان ما نريد ترتيبه ليس أعداد وإنما ملفات نريد المقارنة بين محتوياته بحيث أننا لا نستطيع الاحتفاظ بالملفات كلها في الذاكرة فعليا عمليا من أجل كل عملية مقارنة القراءة من القرص الصلب. وعلى فرض أن كل عملية مقارنة تكلف 100 جزء من الألف من الثانية. أي أننا سوف نحتاج إلى 28 ساعة لترتيب 1000 ملف (في آخر فحص للفيروس على كمبيوتر قام نورتن بإجراء الفحص على حوالي 2 مليون ملف. بكلمة أخرى 1000 ملف ليس عددا مبالغاً). لذلك فإن إيجاد الخوارزمية الملائمة قد يكون عاملا حاسما فيما إذا كانت العملية قابلة للتنفيذ أو لا. فخوارزمية تحتاج إلى 20 ساعة لحساب الطريق الأقصر بين مكانين ضمن مدينة ما سوف تكون غير قابلة للتنفيذ لدى سيارات الأسعاف.

في علم الخوارزميات هناك عدد من المسائل العامة التي تصف مشكلة بسيطة لكنها في الواقع ليست إلا تبسيطا لفكرة ما بحيث أن كل واحدة من هذه خوارزميات حل هذه المسائل قابلة للتطبيق على عدد كبير جدا من المشاكل المختلفة والتي نقوم على مبدأ واحد. من هذه المسائل الشهيرة: Traveling Salesman Problem و Knapsack Problem و Bin Packing Problem و All-Pairs-Shortest-Paths Problem والكثير الكثير. فعلم الخوارزميات هو علم كبير جدا كما أنه ممتع ويحتاج إلى معرفة عميقة بفروق متنوعة للرياضيات كالتحليل والجبر والمنطق وحساب الاحتمالات.

لكسب شعور عما من الممكن أنجازه بقليل من التفكير من أجل تصميم الخوارزمية الأنسب للمشكلة سوف نقوم في المثال التالي بدراسة مشكلة بسيطة ثم نصمم لها خوارزمتين ونشاهد الفرق الكبير في السرعة بين الخوارزمتين. وفي المرة القادمة سوف نستعرض خوارزمية أخرى من أجل حل نفس المشكلة ونرى مجددا الفرق الكبير في السرعة.

MAXSUMME problem

في هذه الحالة سوف يكون الداخل Input مصفوفة Array من الأعداد الصحيحة $\mathbb{Z}$ (أي موجبة وسالبة) والمطلوب منا هو إيجاد أكبر مجموع لأعداد متجاورة أو بكلمات أخرى علينا البحث عن مجال بداخل المصفوفة (مجموعة أعداد وراء بعضها) بحيث يكون مجموع هذه الأعداد اعظما

أي إذا كانت المصفوفة على الشكل التالي $a_1, a_2, a_3, a_4, a_5, \dots, a_l$ فإننا نعرف عليها تابعا من الشكل $f(i, j) = a_i + a_{i+1} + a_{i+2} + \dots + a_j$ والمطلوب هو إيجاد قيمة ل i و j بحيث يكون المجموع $f(i, j)$ اعظما (في الحقيقة لن نهتم ب i و j بهما فقط المجموع).

1. الخوارزمية الساذجة Naive Algorithm

نقوم بإيجاد من أجل كل قيمة ممكن ل i و j بحسب مجموع الاعداد بينهما ونقارن الناتج

```
private int GetMaxSumme(int[] arr)
{
    int max = 0 ;
    int temp;
    for(int i = 0 ; i < arr.Length ; i++)
    {
        for (int j = i ; j < arr.Length ; j++)
        {
            temp = f(arr,i,j);
            if (temp > max)
                max = temp;
        }
    }
    return max;
}

private int f(int[] arr, int i, int j)
{
    int result = 0;
    for( int y = i ; y <= j; y++)
        result += arr[y];
    return result;
}
```

ولكن المشكلة في هذه الخوارزمية انها تقوم بحساب $a_1 + a_2$ n-1 مرة مثلاً. فلنقم بحساب عدد المقارنات التي تنفذ في هذه الخوارزمية. سنلاحظ بأنه يساوي عدد الثنائيات (i, j) الممكنة حيث $i < j$. وهو ما يساوي $\binom{n}{2}$ مقارنة، بالإضافة الى n مقارنة لعدد من النوع (i, i) فيكون عد المقارنات الفعلي :

$$C_1(n) = \frac{1}{2} \cdot n \cdot (n-1) + n = \frac{1}{2} \cdot n \cdot (n+1) = \frac{1}{2} \cdot n^2 + \frac{1}{2} \cdot n.$$

أما بالنسبة لعمليات الجمع فعددها:

$$\begin{aligned} A_1(n) &= \sum_{i=1}^n \sum_{j=i}^n (j-i) = \sum_{i=1}^n \sum_{k=0}^{n-i} k \\ &= \sum_{i=1}^n \frac{(n-i) \cdot (n-i+1)}{2} \\ &= \sum_{i=1}^{n-1} \frac{i \cdot (i+1)}{2} \\ &= \frac{1}{2} \cdot \left(\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i \right) = \dots = \frac{1}{6} n^3 - \frac{1}{6} n. \end{aligned}$$

مع ملاحظة أننا استخدمنا مكان النقط صيغتين مشهورتين جداً وهما

$$\sum_{i=1}^n i = \frac{n \cdot (n+1)}{2}.$$

$$\sum_{i=1}^n i^2 = \frac{1}{3} \cdot n^3 + \frac{1}{2} \cdot n^2 + \frac{1}{6} \cdot n.$$

وبذلك يكون الزمن الكلي (او عدد العمليات الكلي) اللازم لهذه الخوارزمية

$$T_1(n) = C_1(n) + A_1(n) = \frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n.$$

هل هذه الخوارزمية هي افضل ما يمكننا تصميمه ؟ الا نستطيع حساب المطلوب بعدد اقل من المقارنات والجمع ؟ بالطبع نستطيع ذلك بقليل من التفكير من الممكن ان نصمم خوارزمية افضل.

2. الخوارزمية الطبيعية Normal Algorithm :

ان هذه الخوارزمية تعتمد في تحسين الاداء على استخدام ما قد جمع مسبقا في خطوات سابقة وبذلك تقوم بتقليل عمليات الجمع.

```
private int GetMaxSummeNormal(int[] arr)
{
    int max = arr[1];
    int s;
    for(int i = 0 ; i < arr.Length ; i++)
    {
        s=arr[i];
        if(s > max)
            max=s;
        for(int j = i+1; j < arr.Length ; j++)
        {
            s += arr[j];
            if ( s > max)
                max = s;
        }
    }
    return max;
}
```

ان هذه الخوارزمية واضحة ولا تحتاج لشرح. بالطبع هنا نلاحظ ان تعديلا بسيطا قد قلل عد عمليات الجمع بشكل كبير. فمثلا الآن نحتاج لحساب كل قيم $f(i,.)$ فقط $n-i$ عملية جمع (حيث النقطة تعني جميع الاعداد الممكنة). بينما في الخوارزمية السابقة احتجنا لحساب كل قيم $f(i,.)$ الى $(n-i+1) + (n-i) + \dots + (1)$ وبهذا يكون عدد عمليات الجمع الكلي اللازم :

$$A_2(n) = \sum_{1 \leq i \leq n} (n-i) = \sum_{i=1}^{n-1} i = \frac{1}{2}n^2 - \frac{1}{2}n.$$

أما عدد عمليات المقارنة فهو تماما كما في المقال السابق, أي

$$C_2(n) = C_1(n) = \frac{1}{2}n^2 + \frac{1}{2}n.$$

وبذلك يكون العدد الكلي للعمليات الحسابية اللازمة :

$$T_2(n) = C_2(n) + A_2(n) = n^2.$$

وهنا نلاحظ الفرق الكبير جدا بين الخوارزميتين فبدلا من n^3 لدينا الآن n^2 (بغض النظر عن المعامل قبل الحد ذو الدرجة الأعلى).

حسنا وماذا الآن؟ هل حصلنا على الخوارزمية المثالية التي لا يمكن تصميم افضل منها ؟ بانتظار جوابكم.

تحياتي

DotNetExpert

المراجع التي اعتمدت عليها والتي قد اعتمد عليها في المستقبل

Datastrukturen Algorithmen und Programmierung 2.
Prof. Dr. Ingo Wegener, PD Dr. Thomas Hofmeister 1992-2004.

Theoretische Informatik - eine algorithmische Einführung.
Teubner. Wegener, I. (1999).

Kompendium Theoretische Informatik - eine Ideensammlung.
Teubner. Wegener, I. (1996).

Komplexitätstheorie - Grenzen der Effizienz von Algorithmen.
Springer-Verlag. Wegener, I. (2003).

A.V. Aho, J.E. Hopcroft, J.D. Ullman: Data structures and algorithms, Addison-Wesley, 1983