

في القسم السابق من هذه المقالة استعرضنا مشكلة الـ MAXSumme كما استعرضنا الخوارزمية الساذجة والخوارزمية الطبيعية ومن ثم قارنا بين سرعة الخورزميتين واكتشفنا كم من الممكن أن تؤثر الخورزمية على سرعة العمل.

في هذه المرة سوف نتعرف على اتجاه جديد في تصميم الخوارزميات. الفكرة الرئيسية لهذا النوع من الخوارزميات يمكن أن تتلخص بقول للقيصر لويس السادس : Divide et impera أو بالانكليزية Divide and Conquer. أو ما معناه بالعربية فرق تسد. فاسلوب الحل في هذه الخوارزمية يقوم على تقسيم المشكلة الى مشاكل مشابهة أصغر في الحجم. ومن ثم الحصول على حل المشكلة العامة بالاستعانة بهذه الأجزاء الصغيرة. من أجل حل المشاكل الصغيرة المكونة للمشكلة الكبيرة نقوم بتجزئتها هي ايضا الى مشاكل اصغر وهكذا حتى نصل الى مشاكل صغيرة جدا وبسيطة الحل.

3. خوارزمية فرق تسد Divide and Conquer

من أجل تبسيط الحل هنا سوف نعتبر أن طول المجال المراد إيجاد اكبر مجموع لاعداد متتالية عليه n من الشكل $n = 2^k$ أي وذلك من أجل تسهيل تقسيم المجال على 2 في كل مرة.

أول خطوة للحل هي تقسيم جميع الثنائيات $(i, j), 1 \leq i \leq j \leq n$ الى ثلاثة أقسام:

- $1 \leq i \leq j \leq n/2$ جميع الثنائيات الموجودة بشكل تام في الجزء الأول من المجال
- $1 \leq i \leq n/2 < j \leq n$ جميع الثنائيات الموجودة بشكل جزئي في القسم الأول والثاني من المجال
- $n/2 < i \leq j \leq n$ جميع الثنائيات الموجودة بشكل تام في الجزء الثاني من المجال

عندما نحل المشكلة لكل نوع من الثنائيات (i, j) وفق التقسيم المذكور في الأعلى فاننا سوف نحتاج الى مقارنتين اضافيتين من أجل الحصول على الحل النهائي للمشكلة. أو بكلمات أخرى بعد أن نحسب أكبر قيمة لكل فئة من الـ $f(i, j)$ سوف نحصل على ثلاث قيم بمقارنتين بينهم نستطيع الحصول على اكبر قيمة من الثلاثة. هذه القيمة هي بالطبع اكبر قيمة ممكنة لـ $f(i, j)$ على المجال كاملاً.

عندما ندقق في المشكلتين الاولى والثالثة سوف نلاحظ انهما من نفس النمط كالمشكلة الأساسية مع الفرق أن طول المجال الآن $n/2$ بدلا من n . أي أننا نستطيع تطبيق خوارزمية فرق تسد عليهما من جديد، أي بشكل recursive. حسنا وماذا بالنسبة للقسم الثالث، أي المجالات الموجودة في القسم الاول والقسم الثاني معا من المجال الكبير؟ كيف نستطيع التعامل معها؟ من أجل $1 \leq i \leq n/2 < j \leq n$ سوف نقوم بإنشاء تابعين جديدين من الشكل:

$$g(i) := a_i + \dots + a_{n/2} \text{ \& } h(j) = a_{n/2+1} + \dots + a_j.$$

لا أعرف ان كانت الإشارة =: معروفة للجميع ولكنها تعني ببساطة أن ما ياتي بعدها هو تعريف لما يأتي قبلها، أي أننا في حالتنا هذه نعرف تابعا جديدا لم يذكر فيما سبق.

باختصار $g(i)$ تعني مجموع جميع الأعداد ابتداءا بالعدد الموجود في الموقع i من المجال وحتى منتصف المجال. أما $h(j)$ فهي تعني بشكل مشابه مجموع جميع الأعداد ابتداءا من منتصف المجال وحتى العدد الموجود في الموقع j من المجال. وهكذا نستطيع كتابة $f(i, j) = g(i) + h(j), 1 \leq i \leq n/2 < j \leq n$. والآن كل ما علينا فعله من أجل الحصول على أكبر قيمة ممكنة لـ $f(i, j)$ هو الحصول على أكبر قيمة لـ $g(i)$ و $h(j)$.

فالنقم الآن بدراسة واحد فقط منها فالأخرى تحل بشكل مشابه تماما. وليكن $g(i)$. كيف نستطيع الحصول على أكبر قيمة لهذا التابع من أجل $1 \leq i \leq n/2$ ؟ الحل بسيط! بطريقة مشابهة للخوارزمية العادية التي استعرضناها في الجزء الأول نستطيع حساب $g(n/2), g(n/2-1), \dots, g(1)$ ومقارنتها من أجل حساب أكبر قيمة. أي أننا سوف نحتاج فقط الى $n/2-1$ عملية جمع و $n/2-1$ عملية مقارنة من اجل حساب أكبر قيمة لـ $g(i)$. بالطبع سوف نحتاج لنفس العدد من عمليات الجمع ولنفس العدد من المقارنات من أجل الحصول على أكبر قيمة ممكنة لـ $h(j)$ فطريقة الحل مشابهة تماما. وبهذا نكون قد حللنا المشكلة بالنسبة للفئة الثانية للثنائيات.

لنلخص عدد العمليات اللازمة اذن: عدد عمليات المقارنة سيكون مجموع عدد المقارنات التي تقام على الجزء الاول والجزء الثاني من المجال، أي $(n/2-1) + (n/2-1) = n-2$

أما بالنسبة لعمليات الجمع الازمة فهو عدد عمليات الجمع في القسم الأول بالاضافة الى عدد عمليات الجمع في القسم الثاني بالاضافة الى عملية جمع واحد تقام لجمع النتيجة التي نحصل عليها من القسم الأول مع النتيجة من القسم الثاني.

أي $(n/2-1) + (n/2-1) + 1 = n-1$. وبذلك يكون العدد الكلي للعمليات المقامة أجل الحصول على حل الفئة الثانية من الثنائيات: $(n-2) + (n-1) = 2n-3$.

فلنتنقل الآن الى تحليل عد العمليات الكلي لخوارزمية فرق تسد من اجل المجال الكلي أي ذو n عنصرا.
من أجل n عنصرا نحتاج الى تنفيذ نفس الخوارزمية مرتين على مجالين بنصف طول المجال الأصلي بالإضافة الى عدد العمليات من أجل الفئة الثانية من الثنائيات (أي ما قمنا بدراسته قبل قليل) بالإضافة الى عمليتي مقارنة بين نتائج الفئات الثلاث من الثنائيات وذلك للحصول على النتيجة النهائية الكبرى. وبذلك يكون:

$$T_3(1) = 0, T_3(n) = 2.T_3(n/2) + 2.n - 1.$$

من أجل الحصول على فكرة كيف من الممكن أن يكون حل هذه المعادلة سوف نقوم بتعويض المعادلة في نفسها عدد من المرات على أمل أن نحصل على فكرة ما :

$$\begin{aligned} T_3(n) &= 2.T_3(n/2) + 2.n - 1 \\ &= 4.T_3(n/4) + 2.(2.(n/2) - 1) + 2n - 1 \\ &= 4.T_3(n/4) + (2n - 2) + (2n - 1) \\ &= 8.T_3(n/8) + ((2n - 4) + (2n - 2) + (2n - 1)) \end{aligned}$$

الآن نستطيع أن نخمن أن المعادلة سوف تستمر على الشكل التالي :

$$T_3(n) = 2^l.T_3(n/2^l) + \sum_{i=1}^l (2n - 2^{i-1})$$

بالطبع في الرياضيات لا تقبل التخمينات. ويجب برهان هذا الاستنتاج. لكنني هنا لن أذكر هذا البرهان. ان كان أحد مهتم به فليخبرني. نستطيع أن نتناقش به في المرة القادمة. بالطبع ان برهان هذا العلاقة يتم عن طريق الاستقراء الرياضي Mathematical Induction. ان وجدت تجاوبا في هذا الموضوع سوف اقوم بكتابة مقدمة عامة في الاستقراء الرياضي وذلك لاهميته في تحليل تعقيد الخوارزميات.

بالعودة الآن لمثالنا . بما أننا فرضنا أن n من الشكل $n = 2^k$ فاننا نستطيع استبدال k في هذا المثال بـ l :

$$\begin{aligned} T_3(n) &= 2^k.T_3(1) + \sum_{i=1}^k (2n - 2^{i-1}) \\ &= \sum_{i=1}^k (2n - 2^{i-1}) \text{ (Because : } T_3(1) = 0 \text{)} \\ &= 2nk - (1 + 2 + 4 + \dots + 2^{k-1}) \\ &= 2nk - (2^k - 1) \\ &= 2nk - (n - 1) \\ &= (2k - 1).n + 1 \\ &= n.(2 \log n - 1) + 1. \end{aligned}$$

إن مثل هذه المعادلات تطلب بعض الوقت ومعرفة جيدة في الرياضيات لفهمها. فمثلا في السطر الخامس قد استخدمنا صيغة مشهورة جدا وتستخدم بشكل كبير في علم الخوارزميات : $(1 + 2 + 4 + \dots + 2^{k-1}) = (2^k - 1)$. كمان أن $\log$ مقصود به اللوغاريتم للأساس 2. في علم الخوارزميات بشكل عام عندما يكتب اللوغاريتم بدون توضيح الأساس فان المقصود به اللوغاريتم للأساس 2 بينما في كتب الرياضيات بشكل عام يقصد به اللوغاريتم للأساس e . وهذا ما يوضح استخدامنا للوغاريتم حيث : $n = 2^k \Leftrightarrow k = \log n$.

في حالة كون أي عملية أخرى في هذا المثال غير واضحة فانا جاهز لتوضيحها.

في هذا المثال نلاحظ مقدار التغير الكبير الذي طرأ على سرعة الخوارزمية فبدلا من n^2 في الخوارزمية الطبيعية تتطلب هذه الخوارزمية فقط $n.(2 \log n - 1) + 1$. أي مثلا اذا كان طول المجال $n = 1024$ فان عدد العمليات اللازمة للخوارزمية العادية هو 1048576 عملية بينما في حالة استخدام خوارزمية فرق تسد نحتاج فقط الى 19457 عملية.

أما بالنسبة لتطبيق هذه الخوارزمية بلغة C# :

```

public int GetMaxSummeDnC(int[]arr)
{
    return GetMaxSummeDnC( arr,0,arr.Length-1);
}
public int GetMaxSummeDnC(int[]arr, int l, int r)
{
    int length = r-l+1;
    if(length < 1) throw new ArgumentException("The left index musst be smaller than the right one") ;
    if(length == 1) return arr[l];
    int mitte = l+ (int)Math.Ceiling(length/2);
    int maxL = GetMaxSummeDnC(arr,l,mitte-1);
    int maxR = GetMaxSummeDnC(arr, mitte,r);
    int maxM = g(arr,l,mitte)+h(arr,r,mitte);
    int ret = Math.Max(maxR,maxL);
    ret = Math.Max(ret, maxM);
    return ret;
}
private int g(int[]arr,int l, int mitte)
{
    int max = arr[mitte-1];
    int sum = arr[mitte-1];
    for ( int i = mitte -2 ; i >= l; i--)
    {
        sum += arr[i];
        max = Math.Max(max, sum);
    }
    return max;
}
private int h(int[]arr,int r, int mitte)
{
    int max = arr[mitte];
    int sum = arr[mitte];
    for ( int i = mitte+1; i <= r ; i++)
    {
        sum += arr[i];
        max = Math.Max(max, sum);
    }
    return max;
}

```

لقد قامت هذه الخوارزمية بتحسين سرعة حل المشكلة بشكل كبير ولكن هل توجد خوارزمية تحل هذه المشكلة بسرعة خطية من الشكل $a \cdot n + b$ حيث a, b عددين ثابتين لا يتعلقان ب n ؟
 الجواب في الجزء الثالث.
 تحياتي
 DotNetExpert

المراجع كما في القسم الأول