الفصل الأول : مدخل إلى الخوارزميات
Introduction to Algorithms

إن استخدام الحاسوب في معالجة أية مسألة يتطلب القيام بعمل كثير و بذل الجهد الكافي لحل هذه المسألة ، فالحاسوب عبارة عن آلة قادرة على تنفيذ سلسلة من التعليمات بسرعة كبيرة جدا ، و لحل أية مسألة تقع على المبرمج الأعباء التالية :

1. فهم المسألة المراد حلها و الإلمام بأبعادها و عناصرها كلها .
2. صياغة الخوارزمية المناسبة لهذه المسألة كأن تكون الخوارزمية محددة الخطا و متسلسلة بشكل منطقي و أن تكون قابلة للتنفيذ من أجل أية مجموعة قيم ابتدائية مسموح بها ، أي تستطيع الخوارزمية حل كل المسائل المنتمية إلى نوع واحد و أن تصل الخوارزمية في النهاية إلى نتيجة منطقية و صحيحة لحل هذه المسألة .
3. صياغة الخوارزمية على شكل برنامج Program تتم كتابته بإحدى لغات البرمجة ، أي أن **البرنامج** هو عبارة عن توصيف لخوارزمية حل مسألة معينة بإحدى لغات البرمجة التي يقبلها الحاسوب .

1-1 تعاريف

لغة البرمجة (Programming Language): هي مجموعة من المفردات و القواعد و الدلالات المعرفة التي تسمح بكتابة برنامج يمكن تنفيذه على الحاسوب .

المترجم (Compiler) : هو برنامج يفهم البرنامج المكتوب بلغة برمجة معينة ، و يحوله إلى برنامج مكافئ بلغة التجميع (Assembly Language) أو بلغة الآلة (Machine Language) .

الخوارزمية (Algorithm) : اشتقت كلمة الخوارزمية نسبة إلى العالم العربي محمد بن موسى الخوارزمي و هو من أعظم علماء العرب الذين تركوا بصمات جلية في التراث الحضاري العالمي . و قد عاش في عهد المأمون ، و انصرف إلى دراسة الرياضيات و

الجغرافية و الفلك و التاريخ و من أهم كتبه في الرياضيات كتابه المعروف في الجبر (حساب الجبر و المقابلة) و قد نقل الكتاب إلى اللغة اللاتينية .

و يقصد بكلمة خوارزمية بأنها مجموعة الخطوات (أو التعليمات) المنتهية و المتسلسلة التي تؤدي إلى حل مسألة معينة و الوصول إلى نتائجها ، إضافة إلى ذلك ، معظم الخوارزميات تحقق المعايير التالية :

1. عدد المدخلات التي تزود بها الخوارزمية صفر أو أكثر .
2. يوجد خرج واحد على الأقل للخوارزمية .
3. كل خطوة من خطوات الخوارزمية يجب أن تكون واضحة (clear) و غير غامضة (unambiguous)
4. الخوارزمية يجب أن تنتهي بعد عدد منته من الخطوات ، و أن إتباع هذه الخطوات من نقطة البداية إلى نقطة النهاية يؤدي إلى حل المسألة .

2-1 توصيف الخوارزمية Algorithm Specification

هناك عدة طرق لتوصيف (كتابة) الخوارزميات أو طرائق لحل و معالجة مسألة ما و هذه الطرائق يمكن أن تختلف من شخص إلى آخر من ناحية أسلوب الحل و لكنها جميعا تشترك بالنتيجة المتوخاة . و من أهم هذه الطرائق:

1. كتابة الخوارزميات على شكل خطوات باستخدام اللغة المتداولة كاللغة العربية أو الإنكليزية . يفضل استخدام هذه الطريقة عندما تكون الخطوات واضحة .
2. كتابة الخوارزميات باستخدام المخططات البيانية أي تمثيلها بواسطة رسومات بيانية متعارف عليها أشهرها المخططات التدفقية (Flowcharts)، يفضل استخدام هذه الطريقة عندما تكون الخوارزمية بسيطة و قصيرة .
3. كتابة الخوارزميات باستخدام لغة الترميز الزائف (Pseudo Code) و الذي هو مزيج من لغة برمجة مثل C++ و اللغة الانكليزية

والمثال التالي يوضح عملية تحويل المسألة إلى خوارزمية:

مثال: خوارزمية (Selection sort) : بفرض أن لدينا قائمة من العناصر a[1 : n] من نوع ما Type حيث $n \geq 1$ ، و لنرتب عناصر القائمة ترتيبا" تصاعديا" .

يتم ترتيب عناصر القائمة a[1 : n] ترتيبا" تصاعديا" باستخدام الترميز الزائف التالي :

```
        for (i=1; i<=n;i++) {
  examine a[i] to a[n] and suppose the smallest element is at a[j];
                interchange a[i]  and a[j];
                 }
```

لتنفيذ الترميز السابق بلغة برمجة ، سنحتاج إلى مهمتين جزئيتين :

1. إيجاد العنصر الأصغر(Minimum) (ليكن a[j])
2. مبادلة العنصر الأصغر a[j] بـــالعنصر a[i] كما يلي:

```
Type t = a[i]; a[i] = a[j]; a[j]=t;
```

المهمة الجزئية الأولى تحل بافتراض أن minimum هو a[i] ، بعد ذلك يتم مقارنـــة a[i] بـــــ a[i+1], a[i+2], ... و كلما وجدنا عنصرا" أصغر نعتبره minimum جديدا" . و أخيرا يقارن a[n] بـــ minimum الحالي .

باستخدام الملاحظات السابقة يُكتب البرنامج SelectionSort :

```
void SelectionSort (Type a[], int n)
   // Sort the array a[1 : n] into non-decreasing order
 {
        for(int i=1; i<=n; i++){
          int j=i;
         for(int k=i+1; k<=n; k++)
              if (a[k] < a[j]) j=k;
          Type t = a[i]; a[i] = a[j]; a[j] = t;
              }
       }
```

نظرية: خوارزمية SelectionSort(a,n) ترتب بشكل صحيح مجموعـــة مكونـــة مـــن n (n>0) عنصـــــر ، النتيجـــة تخـــزن فـــي المتجهـــة a[1:n] بحيـــث إن:

$$a[1] \leq a[2] \leq \ldots \leq a[n].$$

الإثبات: نلاحظ من أجل أي i ، مثلا i=q يلي التنفيذ للخطوط من 5 إلى 8 ، تكون في حالة أن $a[q] \le a[r], \quad q < r \le n$. و كذلك نلاحظ عندما تكون i أكبر من q فإن a[1: q] لا تتغير. و من ثم يلي التنفيذ الأخير لتلك الأسطر (i.e., i=n) ، نجد أن
$$a[1] \le a[2] \le \ \dots \ \le a[n].$$
نلاحظ من هذه النقطة أن الحد الأعلى لحلقة for في السطر الرابع يمكن أن يتغير إلى n-1 بدون إلحاق أي ضرر لصحة الخوارزمية .

3-1 تحليل الخوارزميات Algorithms Analysis

توجد عدة معايير تمكننا من الحكم على الخوارزمية ، على سبيل المثال:

1. هل الخوارزمية تنفذ ما نرغب عمله ؟
2. هل الخوارزمية تعمل بشكل صحيح وفقا لتوصيفات المهمة ؟
3. هل يوجد توثيق يصف كيف نستخدم الخوارزمية و كيف نعمل بها ؟
4. هل الاجرءات المبنية للخوارزمية تنجز الدوال الجزئية المنطقية ؟
5. هل الكود قابل للقراءة ؟

تعتبر هذه المعايير مهمة بشكل أساسي و خاصة عندما نقوم بكتابة برمجيات للنظم الكبيرة .
في هذا الفصل ، لا نقوم بدراسة كيف يتم الوصول إلى هذه الأهداف ، بل نحاول تحقيق هذه المعايير و ذلك من خلال كتابة البرامج .
توجد معايير أخرى للحكم على الخوارزميات ذات علاقة مباشرة بانجاز الخوارزمية . هذه المعايير تعتمد على زمن التنفيذ و متطلبات تخزين الخوارزميات .

- **زمن التنفيذ:** و يقصد به الزمن اللازم لتنتهي الخوارزمية من إنجاز تعليماتها كافة ، يقاس زمن التنفيذ بحساب عدد التعليمات و الزمن اللازم لتنفيذ كل تعليمة .
- **حجم الذاكرة:** هي كمية الذاكرة اللازمة لتخزين البرنامج و المعطيات التي يعالجها .

كما أن الهدف من تحليل الخوارزميات لا يقف عند قياس المقدارين السابقين ، بل يفيد في مقارنة خوارزميات حل مسألة معينة ، أي أنه : " مهما تكن الآلة التي ستنفذ الخوارزمية و مهما تكن لغة البرمجة المستخدمة ، فإن الخوارزمية A أفضل من الخوارزمية B من أجل معطيات ذات حجم كبير"

أو الحكم من الطراز : " إن الخوارزمية A هي المثلى من حيث عدد العمليات الأساسية التي تقوم بها لحل المسألة " .

4-1 حساب زمن تنفيذ خوارزمية

لحساب زمن تنفيذ خوارزمية يجري التركيز على مسألتين أساسيتين:

- تحديد بعد (أو طول أو حجم) معطيات المسألة من أجل كتابة درجة التعقيد بدلالة هذا البعد .

أمثلة:

1. في مسائل كثيرات الحدود (Polynomials) : يكون البعد هو درجة كثير الحدود أو عدد أمثاله .
2. في مسائل المصفوفات (Matrices) من السعة (m,n) : يكون البعد بدلالة أبعاد المصفوفة مثل : max(m,n) أو m+n أو m.n
3. في مسائل البيانات (Graphs) : يكون البعد بدلالة عدد العقد (Vertices) أو عدد الأضلاع (Edges) أو مجموعهما .
4. في مسائل الترتيب (Sorting): يكون البعد بدلالة عدد العناصر المراد ترتيبها .
5. في مسائل التحليل القواعدي (Syntax Analysis) : يكون البعد بدلالة طول الكلمة .

- اختيار نوع أو عدة أنواع من العمليات ، بحيث يتناسب زمن تنفيذ الخوارزمية مع عدد هذه العمليات . يعتمد هذا الخيار على زمن تنفيذ هذه العمليات ، إذ تهمل العمليات البسيطة أمام العمليات التي هي أكثر كلفة .

أمثلة:

1. في خوارزمية البحث عن عنصر ضمن قائمة يتم التركيز على عدد عمليات المقارنة (و هي الأكثر كلفة في هذه الحالة) بين العنصر الذي نبحث عنه و عناصر القائمة.
2. في خوارزمية ترتيب العناصر في قائمة يتم التركيز على عدد عمليات المقارنة بين العناصر و عدد عمليات التبديل بين المواقع .
3. في خوارزمية ضرب مصفوفتين يتم التركيز على عدد عمليات ضرب و جمع الأعداد.

بعد تحديد أنواع العمليات الأساسية لحساب التعقيد الزمني يحسب عدد العمليات من كل نوع .

<u>ملاحظات مفيدة في حساب التعقيد الزمني لخوارزمية :</u>

- تحسب عدد العمليات في التعليقات و العبارات الإعلانية مثل : int, struct, #include, class, etc., بصفر خطوة،
- تحسب عدد العمليات في عبارة التخصيص و التي لا تستدعي أي برامج أخرى بخطوة واحدة .

- عند وجود العمليات في متتالية من التعليمات فإن عددها الكلي هو مجموع عدد العمليات في كل تعليمة. مثلا : إذا كان P(X) عدد العمليات الأساسية للتعليمة X فإن: $P(X1;X2)= P(X1)+P(X2)$
- عند وجود العملية الشرطية (if then else) فإن عدد العمليات يعطى كحد أعلى :

$$P(\text{if A then B else C}) \le P(A)+\max(P(B), P(C))$$

- عند وجود حلقات فإن عدد العمليات هو: $\sum_i P(i)$ حيث i هو المتحول الذي يتحكم بالحلقة و P(i) عدد العمليات الأساسية المتعلقة بالمرور رقم i في الحلقة .
- أما فيما يتعلق بالبرامج الجزئية (الدوال و الإجراءات) التي لا تستخدم العودية ، يكون عدد العمليات الأساسية الموافقة لاستدعاء برنامج جزئي هو عدد العمليات الأساسية الموجودة في هذا البرنامج الجزئي .

- لحساب عدد العمليات اللازمة لتنفيذ برنامج جزئي عودي ، يجب إيجاد طريقة لحل معادلة عودية. حيث يتم التعبير عن العمليات في الاستدعاء العودي لإجرائيـــة مـــع قيمة n و ليكن T(n) بدلالة T(k) حيث k<n فمثلا" في خوارزمية إيجاد العاملي لعدد صحيح موجب ، إذا اخترنا عملية ضرب عددين صحيحين كعمليـــة أساســـية فإن:

$$T(0)=0$$
$$T(n)=T(n-1)+1 \quad \text{for } n>=1$$

كما أن حل هذه العلاقة العودية هو $T(n)=n$

أمثلة :

مثال: أوجد التعقيد الزمني لخوارزمية تقوم بجمع n عدد حقيقي في الحالتين :

1- بشكل تكراري

2- بشكل عودي .

الحل: 1- الإجرائية بشكل تكراري:

```
1    float Sum(float a[], int n)
2      {
3        float s=0.0;
4          for (int i=1; i<=n; i++)
5            s+=a[i];
6         return s;
7         }
```

لدينا أولا" حجم المعطيات يساوي n ، و كذلك نعتبر أن عملية التخصيص و عبارة return عمليات أساسية ، توجد عبارة تخصيص واحدة في السطر 3 ، عدد العمليات المنفذة في الحلقة for في السطر الرابع يساوي n+1 كما أنه توجد n عملية إسناد و جمع في السطر 5 و كذلك توجد خطوة لتعليمة الاسترجاع في السطر6 . ولذا :

$$T_{Sum}(n)=1+n+1+n+1=2n+3$$

2- الإجرائية بشكل عودي:

```
1   float RSum(float a[], int n)
2       {
3        if(n==0)
4            return (0.0);
5          else
6          return (RSum(a,n-1)+a[n]);
7          }
```

نلاحظ عندما n==0 توجد عمليتان أساسيتان هما عملية المقارنة وعملية الاسترجاع (في السطرين 3 و 4) . أما إذا كان n>0 فعندئذ يوجد $T_{RSum\ (n-1)}$ عملية أساسية لاستدعاء الدالة RSum(a,n-1) ، إضافة لذلك يوجد عمليتان أساسيتان : الجمع و الاسترجاع في السطر السادس . و بذلك يكون زمن التنفيذ للدالة RSum :

$$T_{RSum}(n) = \begin{cases} 2 & if\ n = 0 \\ 2 + T_{RSum}(n-1) & if\ \ n > 0 \end{cases}$$

يشار إلى العلاقة السابقة بعلاقة عودية. تًستخدم طريقة التعويض لحل العلاقة العودية السابقة بالشكل:

$$\begin{aligned} T_{RSum}(n) &= 2 + T_{RSum}(n-1) \\ &= 2 + 2 + T_{RSum}(n-2) \\ &= 2(2) + T_{RSum}(n-2) \\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ &= n(2) + T_{RSum}(0) \\ &= 2n + 2 \qquad n \geq 0 \end{aligned}$$

مثال: لتكن a, b مصفوفتين من نفس السعة (m,n) أوجد التعقيد الزمني لخوارزمية جمع مصفوفتين ؟

الحل: لنعبر عن خوارزمية جمع مصفوفتين بالدالة الإجرائية التالية :

```
void add(Type a[][size], Type b[][size], Type c[][size], int m,  int n)
        {
          for (int i=1; i<=m; i++)
              for (int j=1; j<=n; j++)
            c[i][j] = a[i][j] + b[i][j];
                }
```

حجم معطيات المسألة هو m.n . كما يحتوي البرنامج على حلقتين متداخلتين ، عدد العمليات الأساسية في الحلقة الأولى هو : m+1 و عدد العمليات الأساسية في الحلقة الثانية هو m(n+1) كما أنه يوجد mn عملية جمع و إسناد في السطر الثالث من البرنامج و بالتالي التعقيد الزمني يعطى بالعلاقة:

$$T(mn)=m+1+m(n+1)+mn=2mn+2m+1$$

مثال: أوجد التعقيد الزمني لخوارزمية تقوم بقراءة أي عدد صحيح غير سالب n وتطبع العدد الموافق fn من متتالية فيبوناتشي (Fibonacci) :

الحل: نعبر عن الخوارزمية بالإجرائية التالية:

```
1     void Fibonacci(int n)
2        // Compute the nth Fibonacci number
3        {
4            if (n <=1)
5                cout <<n<<endl;
6            else {
7                int fnm2 = 0, fnm1 =1, fn;
8                for (int i = 2; i <=n; i++) {
9                    fn = fnm1 + fnm2;
10                   fnm2=fnm1; fnm1 = fn;
11                   }
12           cout << fn << endl;
13               }
14           }
```

حجم المعطيات هو n ، لدينا حالتان الأولى عندما n=0 أو n = 1 ، و الثانية n > 1 عندما n=1 أو n=0 فإن كلاً من السطرين 4 و 5 سينفذ بزمن يساوي الواحد . وعندما يكون n > 1 عندئذ ينفذ السطر 7 بزمن يساوي 2 ، والسطر 8 بزمن يساوي n ، والسطر

9 بزمن يساوي n-1 ، والسطر 10 بزمن يساوي 2n-2 ، و ينفذ السطر12 بزمن يساوي الواحد. وبالتالي زمن تنفيذ البرنامج هو :

$$T(n)=1+1+2+n+n-1+2n-2+1 = 4n+2$$

مثال: لتكن a[1:n] متجهة تتضمن n عنصرا" . و نريد حساب التعقيد الزمنـي لخوارزميـة الفرز بالفقاعات (Bubble Sort) .

الحل: تعطى الإجرائية التي تنفذ خوارزمية الفرز بالفقاعات بالشكل التالي :

```
void Bubble Sort (int a[], int n)
    {
      int i, j;
   for(i=n-1; i>=1; i--)
      for (j=1; j<=i; j++)
        if (a[j+1]<a[j]{
          int t=a[j]; a[j]=a[j+1]; a[j+1]=t;
            }
                  }
```

الحل: بعد المسألة هو n و يحتوي البرنامج على حلقتين متداخلتين الأولـى تتضـمن n-1 مرور في الحلقة الخارجية . في المرور n-i نجري i عملية مقارنة و i عملية تبديل مواقع و لـذا :

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{i} 2 = \sum_{i=1}^{n-1} 2i = n(n-1) = n^2 - n$$

5-1 التعقيد الزمني الوسطي - في أسوأ الأحوال – في أحسن الأحوال

يتعلق زمن تنفيذ خوارزمية ما بالمعطيات التي تعالجها . هذه المعطيات تتغير وفق منحيين: تغير في الحجم و تغير في المحتوى ، ففي خوارزمية البحث التسلسلي عن عنصر ضمن قائمة ، يمكن أن يتغير كل من عدد عناصر القائمة (حجم القائمة) و محتواها .

لنرمز بـــ Dn إلى معطيات المسألة ذات الحجم n . و بـــ TA(d) إلى التعقيد الزمني للخوارزمية A من أجل معطيات d .

- نسمي التعقيد الزمني في أحسن الأحوال للخوارزمية A المقدار:

$$MinA(n)=Min\{TA(d)\ ;\ d\in Dn\}$$

- نسمي التعقيد الزمني في أسوأ الأحوال للخوارزمية A المقدار:

$$MaxA(n)=Max\{TA(d)\ ;\ d\in Dn\}$$

- نسمي التعقيد الزمني الوسطي للخوارزمية A المقدار:

$$Average\quad A(n) = \sum_{d\in D_n} P(d)T_A(d)$$

حيث P(d) احتمال أن تكون معطاة الخوارزمية هي d ، و عندما تكون جميع المعطيات متساوية الاحتمال تصبح العلاقة السابقة:

$$Average\ _A(n) = \frac{1}{|D_n|}\sum_{d\in D_n} T_A(d)$$

حيث |Dn| عدد المعطيات ذات الحجم n .

مثال: أوجد عدد المقارنات اللازمة للبحث التسلسلي عن عنصر ضـــمن قائمـــة تحـــوي n عنصرا" ؟

الحل: من الواضح أن : MaxA(n)=n و MinA(n)=1 لحساب AverageA(n) يجب معرفة بعض الاحتمالات حول القائمة L و العنصر x :

- ليكن q احتمال أن يكون العنصر x موجودا" في القائمة L .
- نفترض أنه إذا وجد x في القائمة فإن المواضع كلها متساوية الاحتمال .

من أجل $1 \le i \le n$ سنرمز بـــ $D_{n,i}$ إلى مجموعة كل القوائم التي طولها n و التي يظهر فيها x أول مرة في الموقع رقم i ، و سنرمز بـــ $D_{n,o}$ إلى مجموعة كل القوائم التي لا يظهر فيها x .

بحسب الفرضيات السابقة:

$$P(D_{n,o}) = 1 - q$$

$$P(D_{n,i}) = \frac{q}{n}$$

من تحليل الخوارزمية نستنتج أن: $T(Dn,o)=n$, $T(Dn,i)=i$

ومن ثم :

$$Average_A(n) = \frac{q}{n}\sum_{i=1}^{n} i + (1-q)n = \frac{qn(1+n)}{n\ 2} + (1-q)n =$$

$$\frac{q(1+n)}{2} + (1-q)n$$

فإذا كنا نعلم سلفا" ان x موجود ضمن L فإن : $Average_A(n) = \frac{n(1+n)}{2}$

وإذا كان احتمال وجود x ضمن هو 1/2 فإن : $Average_A(n) = \frac{3(1+n)}{4}$

1–6 مقارنة الخوارزميات

بعد تحديد تعقيد خوارزمية كتابع لحجم المعطيات ، يمكن دراسة سرعة تزايد هذا التابع عندما يزداد حجم المعطيات . تفيد هذه الدراسة في تحديد فعالية الخوارزمية من أجل معالجة معطيات كبيرة الحجم ، إذ يمكن في بعض الحالات أن نجد فروقا هائلة بين خوارزميتين من حيث التعقيد الزمني .

في معظم الحالات نكتفي بتقريب بسيط لتابع التعقيد الزمني ، لمعرفة فعالية الخوارزمية و لمقارنة خوارزميتين . فمثلا عندما تكون n كبيرة من غير المهم أن نعرف أتحتاج خوارزمية معينة إلى n أم إلى n+5 عملية . ويمكن في معظم الأحوال إهمال الثوابت الضربية في تابع التعقيد . فمثلا إذا كنا نريد مقارنة الخوارزمية A التي تعقيدها الزمني $T_A(n) = n^2$ مع الخوارزمية B التي تعقيدها الزمني $T_B(n)=4n$ فإن الخوارزمية A أفضل من الخوارزمية B من أجل n>4 .

تؤول التقريبات السابقة إلى إيجاد ما يسمى مرتبة كبر التابع ، و تجرى مقارنة الخوارزميات على أساس مرتبة الكبر لتوابع التعقيد الزمني .

ولأجل إضفاء منهجية صورية على عمليات مقارنة الخوارزميات في حجوم معطيات كبيرة ، لذلك لابد من التطرق إلى المفاهيم الآتية:

7-1 حدوديات التقارب $[O, \Omega, \Theta, o, w]$

تعريف المجموعة [big -oh] O(g) : لتكن g دالة من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعداد الحقيقة الموجبة . عندئذ O(g) هي مجموعة من الدوال f ، أيضا من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعداد الحقيقة الموجبة، بحيث أنه يمكن إيجاد ثابت حقيقي موجب تماما c ، و يوجد عدد صحيح غير سالب n_0 ، يكون $f(n) \le cg(n)$ من أجل كل $n \ge n_0$

مثال: 3n+2 = (is)O(n) لان $3n+2 \le 4n$ من أجل كل $n \ge 2$. 100n+6=O(n) لان $100n+6 \le 101n$ من أجل كل $n \ge 6$. $10n^2+4n+2=O(n^2)$ لان $10n^2+4n+2 \le 11n^2$ من أجل كل $n \ge 5$. $10n^2+4n+2 \ne O(n)$

ملاحظات:

الكتابة f(n) = O(g(n)) تكافئ $f \in g(n)$

O(1) يشير إلى أن زمن تنفيذ الخوارزمية ثابت ، O(n) يشير إلى أن زمن تنفيذ الخوارزمية خطي ،$O(n^2)$ يشير إلى أن زمن تنفيذ الخوارزمية تربيعي ، $O(n^3)$ يشير إلى أن زمن تنفيذ الخوارزمية تكعيبي ، و $O(2^n)$ يشير إلى أن زمن تنفيذ الخوارزمية أسي .

إذا أخذت خوارزمية لمسألة ما زمن تنفيذ $O(\log_2 n)$ تكون أسرع (من أجل قيم كبيرة بشكل كاف لـ n) من خوارزمية لنفس المسألة إذا أخذت زمن تنفيذ O(n) . وبشكل مشابه نجد أن: زمن التنفيذ $O(n \log_2 n)$ يكون أفضل من $O(n^2)$.

توجد سبعة أزمنة تنفيذ و هي كما يلي على الترتيب:

$$O(1),\ O(\log_2 n),\ O(n),\ O(n\log_2 n),\ O(n^2),\ O(n^3),\ O(2^n)$$

لتكن A, B خوارزميتين تحلان نفس المسألة و لنفترض أننا استطعنا إيجاد تابعين f, g بحيث : $T_B(n)=O(g(n))$; $T_A(n)=O(f(n))$ عندئذ نستطيع مقارنة الخوارزميتين A, B بمقارنة التابعين f, g

نظرية: إذا كان $f(n) = a_m n^m + ... + a_1 n + a_0$ كثير حدود من الدرجة m عندئذ $f \in O(n^m)$

الإثبات:

$$
\begin{aligned}
f(n) &\le \sum_{i=0}^{m} |a_i| n^i \\
&\le n^m \sum_{i=0}^{m} |a_i| n^{i-m} \\
&\le n^m \sum_{i=0}^{m} |a_i|, \quad for \quad n \ge 1
\end{aligned}
$$

و بالتالي فإن $f \in O(n^m)$

تعريف المجموعة $\Omega(g)$ [big-Omega]

ليكن g دالة من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعداد الحقيقة الموجبة عندئذ. $\Omega(g)$ هو مجموعة من الدوال f ، أيضا من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعداد الحقيقة الموجبة، بحيث أن يوجد ثابت حقيقي موجب تماما c، يوجد عدد صحيح غير سالب n0 ، يكون $f(n) \ge cg(n)$ من أجل كل $n \ge n_0$

مثال: $3n+2 = \Omega(n)$ لأن $3n+2 \ge 3n$ من أجل أي $n \ge 1$.
$100n+6 = \Omega(n)$ لأن $100n + 6 \ge 100n$ من أجل أي $n \ge 1$.
$10n^2+4n+2 = \Omega(n^2)$ لأن $10n^2 + 4n + 2 \ge n^2$ من أجل أي $n \ge 1$.
$6*2^n + n^2 = \Omega(2^n)$ لأن $6*2^n + n^2 \ge 2^n$ من أجل أي $n \ge 1$.

نظرية: إذا كان $f(n) = a_m n^m + ... + a_1 n + a_0$ و $a_m > 0$ عندئذ $f \in \Omega(n^m)$

تعريف المجموعة $\Theta(g)$ **[Theta]** : لتكن g دالة من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعداد الحقيقية الموجبة . عندئذ $\Theta(g) = O(g) \cap \Omega(g)$، أي أن ، مجموعة التوابع التي تنتمي إلى كل من $O(g)$ و $\Omega(g)$. أو يمكن تعريف $f \in \Theta(g)$ إذا وفقط إذا وجد ثابتان حقيقيان موجبان c_1, c_2 وعدد صحيح غير سالب n_0 بحيث يتحقق $c_1 g(n) \le f(n) \le c_2 g(n)$ من أجل كل $n \ge n_0$

مثال: $3n+2=\Theta(n)$ لأن $3n+2 \ge 3n$ من أجل أي $n \ge 2$ و $3n+2 \le 4n$ من أجل أي $n \ge 2$ و بالتالي فإن $c_1=3,\ c_2=4$ و $n_0=2$. $10n^2+4n+2=\Theta(n^2)$. $6*2^n+n^2=\Theta(2^n)$.

نظرية: إذا كان $f(n) = a_m n^m + ... + a_1 n + a_0$ و $a_m > 0$ عندئذ $f \in \Theta(n^m)$

تعريف المجموعة $o(g)$ **[Little "oh"]** : تكون الدالة *f(n) = o(g(n))* إذا وفقط إذا كان

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

مثال: $3n+2=o(n^2)$ لأن $\lim_{n \to \infty} \frac{3n+2}{n^2} = 0$ و كذلك $3n+2=o(n\log_2 n)$. كما أن $6*2^n+n^2=o(3^n)$ بينما $6*2^n+n^2 \ne o(2^n)$.

تعريف المجموعة $w(g)$ **[Little omega]** : تكون الدالة *f(n) = w(g(n))* إذا وفقط إذا كان . $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$.

8-1 خواص حدوديات التقارب

ليكن f, g, h ثلاث دوال من مجموعة الأعداد الصحيحة غير السالبة إلى مجموعة الأعـــداد الحقيقية الموجبة عندئذ حدوديات التقارب تحقق الخواص التالية :

1- خاصية التعدي Transitivity :

- $f \in \Theta(g)$ and $g \in \Theta(h)$ then $f \in \Theta(h)$
- $f \in O(g)$ and $g \in O(h)$ then $f \in O(h)$
- $f \in \Omega(g)$ and $g \in \Omega(h)$ then $f \in \Omega(h)$
- $f \in o(g)$ and $g \in o(h)$ then $f \in o(h)$
- $f \in w(g)$ and $g \in w(h)$ then $f \in w(h)$

2- الخاصية الانعكاسية Reflexivity

- $f \in \Theta(f)$
- $f \in O(f)$
- $f \in \Omega(f)$

3-الخاصية التناظرية Symmetry :

$f \in \Theta(g)$ if and only if $g \in \Theta(f)$

4-الخاصية تناظرية المنقول Transpose Symmetry :

- $f \in O(g)$ if and only if $g \in \Omega(f)$
- $f \in o(g)$ if and only if $g \in w(f)$

ملاحظات

- يكون زمن التنفيذ لخوارزمية $\Theta(g(n))$ إذا وفقط إذا كان زمن تنفيذها في أسوأ الأحوال $O(g(n))$ و زمن تنفيذها في أحسن الأحوال $\Omega(g(n))$.
- المجموعة $o(g(n)) \cap w(g(n))$ هي مجموعة خالية
- يوجد تشابه بين المقارنة بين عددين حقيقيين a, b و المقارنة بين الدوال بالشكل :

$$
\begin{array}{lcl}
f(n)=O(g(n)) & \approx & a \le b \\
f(n)=\Omega(g(n)) & \approx & a \ge b \\
f(n)=\Theta(g(n)) & \approx & a = b \\
f(n)=o(g(n)) & \approx & a < b \\
f(n)=w(g(n)) & \approx & a > b
\end{array}
$$

- يمكننا تمديد تعريف حدوديات التقارب على الدوال بمتغيرين بالشكل التالي :

يكون $f(n, m)=O(g(n,m))$ إذا وفقط إذا وجد ثوابت موجبة $c,\ n_0,\ m_0$ بحيث تتحقق العلاقة $f(n,m) \le cg(n,m)$ من أجل $n \ge n_0,\quad m \ge m_0$. و بشكل مشابه يمكن تعريف المجموعتين $\Theta(g(n, m))$, $\Omega(g(n,m))$.

- يمكننا استخدام حدوديات التقارب في المعادلات الرياضية لحذف التفاصيل غير الأساسية من المعادلات . على سبيل المثال : تشير المعادلة $2n^2+3n+1=2n^2+\Theta(n)$ توجد دالة $f \in \Theta(n)$ بحيث $2n^2+3n+1=2n^2+f(n)$ من أجل أي قيمة لـــ n ، في هذه الحالة يكون $f(n) = 3n+1$. في بعض الحالات تظهر حدوديات التقارب في الطرف الأيسر و الطرف الأيمن للمعادلة مثل : $2n^2+\Theta(n)=\Theta(n^2)$ و التي تشير إلى أنه من أجل أي دالة $f \in \Theta(n)$ توجد دالة $g \in \Theta(n^2)$ بحيث $2n^2+f(n)=g(n)$ من أجل أي قيمة لـــ n .

1-9 التوابع الرياضية و السلاسل الشهيرة

لنذكر ببعض التوابع الرياضية و السلاسل الشهيرة التي تساعد في استخدام حدوديات التقارب

توابع القاع و السقف Floors &Ceilings

من أجل أي عدد حقيقي x ، قاع x هو أكبر عدد صحيح أصغر أو يساوي x و يرمز له بـــ $\lfloor x \rfloor$. أما سقف x فهو أصغر عدد صحيح أكبر أو يساوي x و يرمز له بـــ $\lceil x \rceil$.

خواص توابع القاع و السقف

1- من أجل أي عدد حقيقي x يكون : $x-1<\lfloor x \rfloor \le x \le \lceil x \rceil < x+1$

2- من أجل أي عدد صحيح موجب n يكون: $\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil = n$

3- من أجل أي عددين صحيحين a, b غير صفريين ، فمن أي عدد صحيح n يكون :

$$\left\lfloor \frac{\left\lfloor \frac{n}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor \quad \text{و} \quad \left\lceil \frac{\left\lceil \frac{n}{a} \right\rceil}{b} \right\rceil = \left\lceil \frac{n}{ab} \right\rceil$$

خواص التوابع الآسية Exponentials

- من أجل ثلاثة أعداد حقيقية $a \neq 0$، m و n لدينا المتطابقات التالية :

$$a^0 = 1$$
$$a^1 = a$$
$$a^{-1} = \frac{1}{a}$$
$$(a^m)^n = a^{mn}$$

$$a^m a^n = a^{m+n}$$

- معدل نمو التابع الآسي الموجب يكون أسرع من معدل نمو أي كثير حدود لأن : من أجل الثابتين الحقيقيين a>1 و b لدينا $\lim_{n \to \infty} \frac{n^b}{a^n} = 0$ أي أن $n^b = o(a^n)$.

- إذا اعتبرنا ....e = 2.71828 أساس التابع اللوغاريتم . عندئذ من أجل أي عدد حقيقي x لدينا:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^2}{2!} + \ldots\ldots = \sum_{i=1}^{\infty} \frac{x^i}{i!}$$

وبالتالي من أجل أي عدد حقيقي x ، تتحقق المتراجحة : $e^x \geq 1 + x$. تصبح المتراجحة السابقة مساواة عندما فقط يكون x = 0 أما إذا كان $|x| \leq 1$ يكون لدينا التقريب التالي :

$$1 + x \leq e^x \leq 1 + x + x^2$$

عندما تسعى x إلى اللانهاية يكون : $e^x = 1 + x + \Theta(x^2)$

خواص اللوغاريتميات Logarithms

- من أجل الأعداد الحقيقية a > 0, b > 0, c > 0 و n لدينا :

$$a = b^{\log_b a}$$

$$\log_c(ab) = \log_c a + \log_c b$$

$$\log_c(a^n) = n \log_c a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\log_b(\frac{1}{a}) = -\log_b a$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b n} = n^{\log_b a}$$

- يعطى منشور التابع *ln(1+x)* عندما |x| < 1 بالشكل :

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - ....$$

تابع العاملي Factorials

- ليكن n عددا" صحيحا" غير سالب ، نعرف !n بالشكل :

$$n! = \begin{cases} 1 & if \quad n = 0 \\ n.(n-1)! & if \quad n > 0 \end{cases}$$

أي أن　　n! = 1.2.3……n

- تعطى !n بحسب تقريب Stirling بالشكل التالي:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

أعداد فيبوناتشي Fibonacci Numbers

تعرف أعداد فيبوناتشي بالعلاقة العودية :

$$f_i = \begin{cases} 0 & if \ i = 0 \\ 1 & if \ i = 1 \\ f_{i-1} + f_{i-2} & if \ i > 1 \end{cases}$$

أي أن كل عدد من أعداد فيبوناتشي ينتج من مجموع العددين السابقين له ، و بالتالي السلسلة التالية تعرف أعداد فيبوناتشي : ..... , 55 , 34 , 21 , 13 , 8 , 5 , 3 , 2 , 1 , 1 , 0

كما أنه يمكن أن تعرف أعداد فيبوناتشي باستخدام الصيغة التالية :

$$f_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}} \quad , \quad i = 0,1,2,3,....... \qquad where$$

$$\phi = \frac{1+\sqrt{5}}{2} = 1.61803 \quad ....., \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2} = -0.61803 \quad .....$$

1-10 تمارين الفصل الأول

تمرين1: أوجد التعقيد الزمني للمقاطع البرمجية التالية :

```
for (int i=1; i<=n; i++)
    for (int j=1; j<=i; j++)
        for (int k=1; k<=j ; k++)
            x++ ;
```

(a)

```
int i=1;
while (i<=n){
x++;  i++;
 }
```

(b)

تمرين2: أوجد التعقيد الزمني للدوال التالية :

```
void D(int x[], int n)
    {
      int i = 1;
      do{
           x[i] += 2; i+=2; }
          }     while (i<= n )
           i = 1;
           while (i <=(n/2) {
                 x[i] += x[i+1]; i++;
              }
                }
```

```
void Transpose (int a[][] , int n)
        {
      for (int i =1; i<=n ; i++)
         for (int j = i+1; j<=n; j++){
           int t = a[i][j] ; a[i][j] = a[j][i]; a[j][i]=t;
              }
                    }
```

```
void mystery (int n)
```

```
              {
               int i, j, k;
               int r = 0;
              for(i=1; i<=n-1; i++)
                  for (j = i+1; j<=n; j++)
                      for (k=1; k<=j; k++)
                           r = r+1;
                                     }

void pesky (int n)
        {
        int i, j, k;
        int r = 0;
               for(i=1; i<=n; i++)
                   for (j = 1; j<= i; j++)
                       for (k=j; k<=i+j; k++)
                            r = r+1;
                                      }
void pestiferous (int n)
        {
        int i, j, k,l;
        int r = 0;
               for(i=1; i<=n; i++)
                   for (j = 1; j<= i; j++)
                       for (k=j; k<=j+i; k++)
                          for (l=1; k<= i+j-k; l++)
                               r = r+1;
                                      }
void conundrum (int n)
        {
        int i, j, k;
        int r = 0;
               for(i=1; i<=n; i++)
                   for (j = i+1; j<= n; j++)
                       for (k=i+j-1; k<= n; k++)
                             r = r+1;
                                      }
```

تمرين3: بيّن أن المعادلات التالية صحيحة :

1. $17 = \Theta(1)$
2. $5n^2 - 6n = \Theta(n^2)$
3. $n! = O(n^2)$
4. $2n^2 2^n + n \log_2 n = \Theta(n^2 2^n)$
5. $\sum_{i=0}^{n} i^k = \Theta(n^{k+1})$
6. $33n^3 + 4n^2 = \Omega(n^3)$

تمرين4: بيّن أن المعادلات التالية غير صحيحة :

(a) $10n^2 + 9 = O(n)$

(b) $n^2 \log_2 n = \Theta(n^2)$

(c) $\dfrac{n^2}{\log_2 n} = \Theta(n^2)$

تمرين5: رتب التوابع التالية بحسب معدل نموها (growth rate) :

$n,\ \sqrt{n},\ \log_2 n,\ \log_2 \log_2 n,\ \log^2{}_2 n,\ \dfrac{n}{\log_2 n},\ \sqrt{n}\log^2{}_2 n,\ (\dfrac{1}{3})^n,$

$(\dfrac{3}{2})^n,\ 17.$

تمرين 6 : أثبت مايلي :

1- إذا كان k عددا" موجبا" فإن $kf(n) = O(f(n))$

2- إذا كان $f(n) = O(h(n))$ و $g(n) = O(h(n))$ فإن $f(n)+g(n) = O(h(n))$

3- إذا كان $T1(n) = O(f(n))$ و $T2(n) = O(g(n))$ فإن :

$T1(n)T2(n) = O(f(n)g(n))$ و كذلك $T1(n)+T2(n) = O(max((f(n),g(n)))$

تمرين 7: بيّن إذا كان c عدد حقيقيا" موجبا" و $g(n)=1+c+c^2+......+c^n$ عندئذ يكون :

$g(n)=\Theta(1)$ إذا كان c<1 ، و كذلك

$g(n)=\Theta(n)$ إذا كان c=1 ، و أيضا" يكون

$g(n)=\Theta(c^n)$ إذا كان c>1 .

تمرين8: ليكن كثيرا الحدود A(x) و B(x) حيث :

$$A[x]=\sum_{i=0}^{n} a_i x^i \quad , \quad B[x]=\sum_{j=0}^{n} b_i x^i$$

المطلوب : 1- اكتب دالة إجرائية تقوم بحساب الجداء $P[x]=A[x]*B[x]$

2- أوجد التعقيد الزمني لهذه الدالة .

تمرين 9: اكتب دالة إجرائية تقوم بضرب مصفوفتين مربعتين و احسب زمن تنفيذها .

تمرين 10 : بفرض a عددا" (صحيحا" أو حقيقيا") و ليكن n عددا" صحيحا" موجبا" . اكتب دالة إجرائية لحساب القوة a^n و احسب زمن تنفيذها .

تمرين 11: المربع السحري (magic square) هو مصفوفة مربعة من المرتبة n عناصرها أعداد صحيحة تتراوح بين 1 و n^2 بحيث يكون مجموع عناصر أي سطر فيها يساوي مجموع عناصر أي عمود فيها و يساوي مجموع العناصر الواقعة على قطرها الرئيسي و يساوي أيضا" مجموع عناصر قطرها الثانوي . على سبيل المثال ، من أجل n=5 لدينا المربع السحري التالي :

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

اكتب دالة إجرائية تقوم بإنشاء مربع سحري مكون من n×n مدخل و أوجد زمن تنفيذها .

تمرين 12: العدد الأولي (prime number) هو عدد صحيح موجب أكبر من الواحد و قواسمه العدد واحد والعدد نفسه فقط . اكتب دالة إجرائية تختبر فيما إذا كان عددا" صحيحا" موجبا" معطى أوليا" أم لا و أوجد زمن تنفيذها.