تعقيد الخوارزمية

م. سعيد سريحيني

بسم الله الرحمن الرحيم

نبدأ سوياً اليوم بالمحاضرة الأولى عملي لمادة الخوارزميات والتي سنتناول فيها عن تطبيقات عملية وأمثلة لما نتعلمه من أساسيات في النظري وسنتعرف على بنى المعطيات بشكل أوضح وبتطبيق أوسع فنسأل الله منا القبول ونتمنى التوفيق لنا ولكم ❤.

سنتعلم في هذه المحاضرة:

1. مفهوم تعقيد الخوارزمية وأنواعه.
2. معرفة التعقيد الزمني والزمن اللازم لتنفيذ التعليمات.
3. قواعد لكتابة التعقيد.
4. سنحل بعض التمارين لمعرفة تعقيدها.

تعقيد الخوارزمية:

هو الزمن اللازم لتنفيذ خطوات الخوارزمية والحجم الذي ستقوم بحجزه في الذاكرة أثناء تنفيذها وتعتبر الذاكرة والزمن هما معياران أساسيان للمقارنة بين البرامج (الخوارزمية الأفضل).

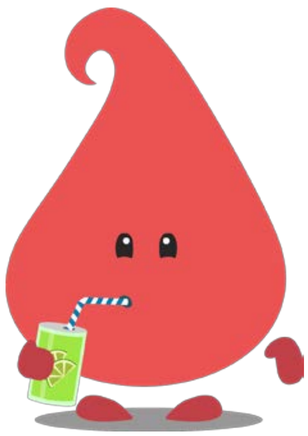
☺ أنواع التعقيد:

- التعقيد الزمني: هو الذي يهتم بمدة (الزمن) الذي يستغرقه البرنامج ليقوم بالتنفيذ وسنركز على هذه النوع في دراستنا لما فيه من حالات خاصة.
- التعقيد الذاكري: هو الذي يهتم بحجم استهلاك البرنامج من الذاكرة ويمكننا معرفة هذا التعقيد بسهولة من خلال جمع المتحولات.

لمعرفة التعقيد الزمني لخوارزمية ما (أي الزمن اللازم لتنفيذها):

لا بد من معرفة العمليات التي ستقوم بتنفيذها والتي تصنف وفق:

1. عمليات مكلفة للوقت.
2. عمليات غير مكلفة للوقت.



العمليات المكلفة للوقت	العمليات الغير مكلفة للوقت
الضرب والقسمة والجمع والعمليات الحسابية	عملية الإسناد
الحلقات بأشكالها (for, while)	عملية زيادة عداد الحلقات for (int i = 0 ; i <n ; i++)
المقارنات المنطقية (if, Switch)	
سنقوم بعدّ هذه العمليات لأن لها تأثير كبير	سنقوم بإهمال عدد هذه العمليات لأنها لا تؤثر على وقت تنفيذ الخوارزمية.

- مثال: أي الخوارزميات التالية هي الأفضل والأقل تعقيداً؟

1-
```
for(int i = 0 ; i< n ; i++)
      for(int j = 0 ; j< n ; j++)
            // Statements
```

2-
```
for(int i = 0 ; i< n ; i++){
      // Statements
  int n[m];
}
```

في هاتين الخوارزميتين نرى أنه من الواضح أن الخوارزمية الثانية هي الأفضل لإنها ستنفذ n مرة فقط.
أما الخوارزمية الأولى يوجد فيها حلقتان for متداخلتان حيث أن الحلقة الداخلية ستنفذ n مرة وأيضاً ما هو داخل الحلقة الداخلية سينفذ n مرة أي أنه بالتالي ستنفذ $n \times n = n^2$ مرة.

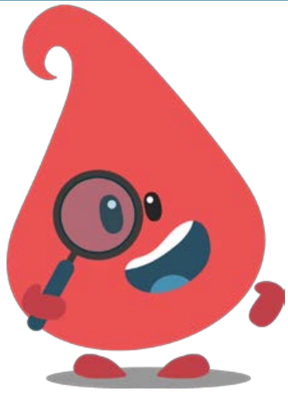
نعبر عن التعقيد بالرمز O(...) وهذا الرمز يستخدم في التحليل للدلالة على غمر التابع وهنا يقصد به العمومية والتعقيد الأشمل.

قواعد لكتابة التعقيد:

1. $O(C.n) \rightarrow O(n)$
   يهمل الثابت عند حساب التعقيد لأنه من الصعب حسابه بدقة.
2. $O(C+n) \rightarrow O(n)$
3. $O(n^x + n^y) \rightarrow O\left(n^{\max(x,y)}\right)$
   نقوم بالمقارنة بين x و y ونضع الأكبر مثلاً $O(n^2 + n^3) \rightarrow O(n^3)$

- سؤال: أوجد تعقيد التمارين التالية:

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



الشرح:

في الحلقة الخارجية:
نبدأ من $i = 0$ نقارنها مع ال n إذا أكبر تكسر الحلقة وإذا أصغر تتابع التعليمات
في الحلقة الداخلية:
نبدأ من $j = 0$ ونقارنها مع ال i وفي كل مرة يزداد العداد j إلى أن تصبح أكبر أو تساوي i لتخرج من الحلقة وتعود للحلقة الخارجية، أي في كل مرة ستنفذ وتكسر الحلقة لتعود للحلقة الخارجية من جديد.

عدد العمليات	قيمة j عند كل i (نقصد بإشارة × كسر شرط الحلقة)	قيمة i
0	$j \to\times$	$i = 0$
1	$j = 0 \to\times$	$i = 1$
2	$j = 0 \to 1 \to\times$	$i = 2$
3	$j = 0 \to 1 \to 2 \to\times$	$i = 3$
$\vdots$	$\vdots$	$\vdots$
$n - 1$	$j = 0 \to 1 \to \cdots \to (n-2) \to\times$	$i = n - 1$

نلاحظ أن عدد العمليات هو n عملية وبالتالي يمكن حساب تعقيد هذه الخوارزمية عن طريق جمع عدد العمليات
ونلاحظ أنه يمكن حساب التعقيد وفق القانون لحساب متسلسلة حسابية:

$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{n}{2}$$

فيكون التعقيد بعد إهمال الثوابت والأخذ بالعدد ذو الأس الأكبر يكون التعقيد النهائي هو $\underline{O(n^2)}$

2-

```
int sum = 0;
int  j = 0;
 for(int i = 0 ; i<n ; i++){
    for (   ;  j< i ; j++)
          sum +=j;
    }
```

التمرين الثاني:

أيضاً سنبدأ من الحلقة الخارجية:
وسنقارن 1 مع n ثم إلى الحلقة الداخلية لكن هنا ننتبه أننا عرفنا j خارج الحلقة أي ستحتفظ بقيمتها (ولن تعود إلى قيمتها الابتدائية =0)
وعندما تصبح $i > j$ فيكسر شرط التكرار، لذلك تعتبر عملية واحدة أي بالمقارنة مع مبدأ الجدول السابق يكون لدينا:

عدد العمليات	قيمة j عند كل i	قيمة i
0	$j \to\times$	$i = 0$
1	$j = 0 \to 1 \to\times$	$i = 1$
1	$j = 1 \to 2 \to\times$	$i = 2$
1	$j = 2 \to 3 \to\times$	$i = 3$
$\vdots$	$\vdots$	$\vdots$
1	$j = n - 2 \to n - 1 \to\times$	$i = n - 1$

n(مرة) $\longrightarrow$ فالتعقيد $O(n)$

- سؤال يخطر الأذهان وهو ما الفرق بين التمرين الأول والثاني؟ من حيث تأثير مكان التعليمة $j = 0$ على قيمة i؟

في المثال الأول: يتم إعادة تعيين $j = 0$ في كل مرة تبدأ فيها حلقة for الداخلية ففي المرة الأولى تأخذ j قيم من $(n-1)$,... ,0,1,2,3 في الدخول الأول وعندما تكسر الحلقة يتم إعادة تعيين j=0.

أما في المثال الثاني: فلا يتم إعادة تعيين قيمة j مطلقاً (لا يتم إعادة تعيين قيمتها في كل مرة تبدأ فيها الحلقة for) لأن قيمة j ثبتت خارج الحلقة فتأخذ القيم $0,1,\dots,i$ وعند الدخول مرة ثانية للحلقة لا يتم تعيين j (أنها تساوي الصفر) بل عند أخر قيمة لها عند كسر الحلقة عنده.

- التمرين الثالث:

التابع RAND هو تابع يعطي قيم عشوائية.
وفي مثالنا الشرط أن تكون هذه القيم زوجية حتى يكسر الحلقة.

```
3-
int sum = 0;
int j = 0
for (int i = 0 ; i<n; i++) {
   for (   ; j<n; j++){
      if(rand( )%2==0)
         break ;
      sum +=I ; } }
```

- سنناقش ثلاث حالات:

1. في حال التابع rand أعطى جميع القيم زوجية فسيتم هنا كسر الحلقة في كل مرة.
وبحسب الحلقة الخارجية فستنفذ العملية n مرة فالتعقيد سيكون $O(n)$.
2. في حال التابع rand أعطى جميع القيم فردية لن تنكسر الحلقة أبداً ويزداد العداد j ليصل إلى n وبعدها لن يدخل إلى هذه الحلقة ثانية، فالتعقيد $O(n)$.
3. الحالة العامة وهي أن يعطي التابع قيم زوجية وقيم فردية مثلاً لنفترض أنه يعطي 5 قيم فردية وواحدة زوجية فالـ i يتم إعادة تعيّنها (0) بينما الـ $j = 5$ تحافظ على قيمتها والحالة الأسوأ مثلاً أنه في الحلقة الثانية يعطي 10 قيم فردية وبعدها زوجية هنا قيمة i تصفّر بينما j تزداد لتصل إلى n وهنا تنكسر الحلقة الداخلية دائماً.
أي سينفذ البرنامج n مرة ثم لن يعد ينفذ فالتعقيد $O(n)$.

- التمرين الرابع:

```
4-
int i = 1;
while (i<n){
   i=i*2;
   sum+=i;}
```

التعقيد $O(\log n)$

يا ترى ما هو تعقيد هذه الخوارزمية؟
ولكن كيف يكون التعقيد $\log n$
عندما نقول عن $\log(\text{عدد}) =$ كم bit في هذا العدد
مثلاً $\log_2 1000 = 10$ والـ 1000 عند تحويلها إلى النظام الثنائي سيكون فيها 10 خانات

عشري	2	× 2	4	3	× 2	6
ثنائي	10		100	11		110

نلاحظ أن الضرب بـ 2 يضيف 0 بالنظام الثنائي وأيضاً ينطبق هذا على النظام العشري فعندما نضرب بـ 10 نزيد صفر وعندما نقسم على 10 نحذف خانة ولذلك فأن أي حلقة فيها ضرب أو قسمة على 2 أو أي عدد تعتبر log.

التمرين الخامس

```
5-
for(int i = 0;i<n;i++){
   int j = 0 ;
    while (j<n){
        j*=2;
        for (int k=n; k>1; k=k/2)
            sum+=k;
}
```

التعقيد $O(\log^2{}_2 n)$

التمرين في هذه الحالة تعقيده سيكون $O(\infty)$ لأن $j = 0$ وأي عدد ضرب 0 هو 0 وبالتالي الحلقة لن تنكسر أبداً وسيكون لا نهائية.

ولكن في حال كانت $j = 1$:

في هذه الحالة سنبدأ بحلقة *for* الداخلية

وتعقيدها $O(log\ n)$

حلقة *while* أيضاً تعقيدها $O(logn)$

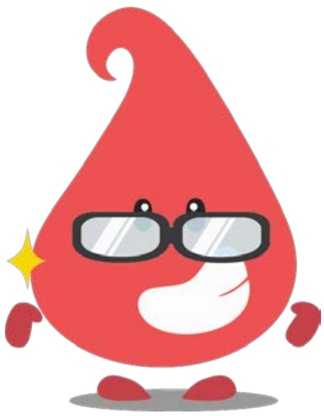
حلقة *for* الخارجية سيتم تنفيذها *n* مرة

التعقيد النهائي سيكون جداء تعقيد الحلقات السابقة

فيكون تعقيد البرنامج النهائي $O(\log_2 n \times \log_2 n \times n) = O(n\ \log^2{}_2 n)$

ملاحظة:

- عندما يكون البرنامج مكون من أكثر من حلقة متداخلة نبدأ بالحلقة الداخلية ونوجد تعقيدها ونستعيض عنها بالتعقيد، ثم ننتقل إلى الحلقة الأشمل وأيضاً نوجد تعقيدها لنصل إلى الحلقة الخارجية.



انتهت المحاضرة...

Once you stop learning...

You Start dying.

Albert Einstein