الجمهورية العربية السورية

جامعة طرطوس

كلية العلوم قسم الرياضيات

السنة الأولى

المادة: خوارزميات_ عملي

المحاضرة الأولى

/تحليل الخوارزميات/

*مفهوم تحليل الخوارزميات:

- يدرس كفاءة الخوارزميات من ناحية الوقت والمساحة(الذاكرة) التي يحتاجها تنفيذ الخوارزمية.
- الهدف من تحليل الخوارزميات هو دراسة تعقيد الوقت وتعقيد المساحة.

*الفرق بين العمليات الأولية والعمليات الأساسية:

- العملية الأولية: هي أي خطوة يكون وقت تنفيذها دائماً محدد بمقدار ثابت بغض النظر عن بيانات الإدخال والخوارزميات المستخدمة ويكون زمنها 1s.

  مثال: عملية الإسناد(=) و العبارات العلائقية(> , < ,=!, == ,=< ,=>)

  العمليات الحسابية(+, /, -, *, %).

  عملية الطباعة والإدخال(coutوcin)
- العمليات الأساسية: هي العملية الأولية الأكثر تكراراً من بين جميع العمليات الأولية الأخرى.

*حساب معدل النمو:

هو معدل تغيير وقت أو مساحة الخوارزمية مع تغيير حجم المدخلات.

أمثلة: قم بحساب زمن تنفيذ الخوارزمية:

مثال1:

طريقة الحل نحسب مجموع الخطوات(الزمن) لكل عملية أولية.

فيكون زمن التنفيذ:

T(n)=1+1+1+1+1=5s

Int b=0;	عملية إسناد 1s
Int a=4;	عملية إسناد 1s
Int z=b+a;	عملية إسناد1s وعملية جمع 1s
Cout <<z;	عملية طباعة1s

ملاحظة: بالنسبة لعملية (الزيادة السابقة أو اللاحقة _ النقصان السابق واللاحق) تعتبر خطوتين لأنها تحتوي عمليتين أساسيتين جمع واسناد كما يلي

$$x++ \iff x=x+1$$

حساب عدد الخطوات ضمن حلقة for:

For(القيمة الابتدائية للعداد ; شرط التوقف ; مقدار التغيير في العداد)

{ التنفيذ }

القيمة الابتدائية للعداد هي عبارة عن عملية أولية (عملية اسناد) لذلك تنفذ بخطوة واحدة أي **(1s)**.

شرط التوقف أو مقدار المقارنة: هي عملية أولية عبارة عن (< أو > أو <= أو >=) تنفذ بخطوة واحدة (1s) ولكنها تتكرر عدد من الخطوات ضمن الحلقة لمعرفة عدد هذه الخطوات يجب أن ننظر للبداية والنهاية وهنا لدينا بعض الأمثلة.

```
For (int i=0; i<n; i++)
```

بفرض كانت n=3 والبداية هنا i=0 والشرط i<n

تكون القيم التي سوف تتم مقارنتها كالتالي

0
1
2
3 ← هذه القيمة لا تحقق الشرط ولكن يتم المرور عليها ومقارنتها

نلاحظ أن عدد القيم التي مررنا عليها هي
4
أي عدد الخطوات
N+1

```
For (int i=0; i<=n; i++)
```

بفرض كانت n=3 والبداية هنا i=0 والشرط i<=n

تكون القيم التي سوف تتم مقارنتها كالتالي

0
1
2
3
4 ← هذه القيمة لا تحقق الشرط ولكن يتم المرور عليها ومقارنتها

نلاحظ أن عدد القيم التي مررنا عليها هي
5
أي عدد الخطوات
N+2

```
For (int i=1; i<n; i++)
```

بفرض كانت n=3 والبداية هنا i=1 والشرط i<n

تكون القيم التي سوف تتم مقارنتها كالتالي

1
2
3 ← هذه القيمة لا تحقق الشرط ولكن يتم المرور عليها ومقارنتها

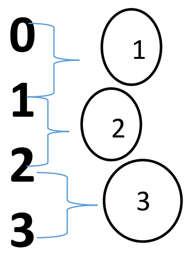
نلاحظ أن عدد القيم التي مررنا عليها هي
3
أي عدد الخطوات
N

مقدار التغيير في العداد

يحتوي عملية حسابية وإسناد أي تحتاج خطوتين (2s) ولكن عملية التغيير تتم في كل مرة يتحقق فيها شرط الحلقة

لاحظنا في الأمثلة السابقة لعملية المقارنة أن أخر قيمة غير محققة أي أن خطوات تغيير العداد سوف تنقص دائماً عن عملية المقارنة بمقدار 1

مثال توضيحي



نلاحظ أن عدد

خطوات المقارنه 4 أي

N+1

وعدد مرات زيادة العداد 3

أي عدد الخطوات للتغيير

N

فيكون الزمن 2مضروبة بعدد الخطوات

التنفيذ

هي العمليات التي يتم تكرارها ضمن حلقة for وهي تنفذ عند تحقق الشرط فقط

أي نفس عدد خطوات مقدار التغير

مثال: احسب زمن تنفيذ خوارزمية إيجاد مجموع الأعداد بين 0وn:

```
Int sum =0;                 1s
For (int i=o; i<n;i++)      1s   N+1   2n
{sum=sum + i;}              2n
cout<<i;}                   1s
```

$$T(n)=1+1+(n+1)+2n+2n+1=5n+4$$

مثال:

```
Int a,b,c;
For (int i=o; i<n;i++)      1s   N+1   2n
{a=a+2;
b=b*3;                      6n
c=c+1;}
}
```

$$T(n)=1+(n+1)+2n+6n=9n+2$$

زمن التعقيد للخوارزمية Big-o-

هو الحد الأعلى للتابع دون أمثال أي هو يمثل العملية الأكثر تكراراً.

أمثلة:

$$T(n) = 5n + 4 \qquad \Rightarrow o(n)$$

$$T(n) = 15\, n^2 + n + 1 \qquad \Rightarrow o(n^2)$$

ملاحظة: الثوابت زمن تعقيده o (1)

```
{ Int k=100;
For (int i=0; i<k; i++)
Cout <<i;}
```

O (1)

لأن ل k قيمة ثابته هي 100

في حلقات for:

بما أن زمن التعقيد هو الحد الأعلى دون ثوابت أي ننظر إلى العملية الأكثر تكراراً (أكثر عدد خطوات) بالشكل التالي:

```
For (int i=0; i<n; i++)
{Statement;}
```

نلاحظ هنا أن أكثر عدد خطوات هو المقارنة وهي n+1 مع تجاهل الثوابت تكون o(n)

```
For (int i=0; i<n*n; i++)
{Statement;}
```

نلاحظ هنا أن أكثر عدد خطوات هو المقارنة وهي $n^2 + 1$ مع تجاهل الثوابت تكون $o(n^2)$

```
For (int i=0; i*i<n; i++)
{Statement;}
```

$i^2 < n \Rightarrow i < \sqrt{n}$

فيكون $o(\sqrt{n})$

ملاحظة: لو كان مقدار التغير في العداد هو ضرب فإن big_o_ هو لوغاريتم.

```
For (int i=0; i<n; i*=2)
{Statement;}
```

فيكون $o(\log_2 n)$