المحاضرة العاشرة .   / /

أعداد موجبة $c>0$

$$\frac{|f(x)|}{x^m} \le |a_m| + \dots + |a_1| + |a_0|$$

$$\Rightarrow \frac{|f(x)|}{x^m} \le c \Rightarrow |f(x)| \le c \cdot x^m$$

$$\# \; f(x) = O(x^m) \quad \Leftarrow$$

كيف نستفيد منها $T(n) = 2n^2 = O(n^2)$ // مثال سابق

لو كانت $T(n) = 1000n^3 + 60n$

$$\Rightarrow T(n) = O(n^3)$$
$c \cdot g(n)$

انتهت المحاضرة ٩

المحاضرة العاشرة

كتابة ضرب مصفوفتين. $A_{n\times m} \times B_{m\times r} = C_{n\times r}$ {3 حلقات for}

for   for   for

⇐ عدد عمليات الضرب n^3 مرة $\{n=m=r\}$

ودرجة تعقيده $O(n^3)$.

// هامش : طريقة ستراسن لضرب مصفوفات ، مثل $O(n^{2.7})$ درجة تعقيدها بطريقة ستراسن % (عدد n أي n كبيرة جداً)

مثال : ليكن لدينا

```
p = 1 ;
while (n > 1) {
  p = p x n ;
  n = n - 1 ;
}
```

```
p = 1 ;
for i = 1 to n do   (n)
  p = p x n ;
  n = n - 1 ;
```

في for سيكون $T(n) = n$.

, المطلوب الكلفة بالنسبة لعدد عمليات الضرب .

هامش // الكلفة : عدد مرات الدخول x عدد مرات التكرار //   الحل : بفرض i عدد مرات الدخول الحلقة

المتحول n هو عنوان في الذاكرة وقيمته قيمة آنية

$\Rightarrow$ n (أول دخول) , $n-1$ (ثاني دخول) , $n-2$, ---- $n-i$

لم يتحقق شرط while $\Rightarrow n-i \le 1$

(لا يتحقق الشرط) $\Rightarrow n-i=1 \Rightarrow i=n-1 \Leftarrow$

$$T(n) = n-1$$

الكلفة تساوي $\{1 \times i$ تكرارات

مثال:

```
p = 1
while (n > 1) do
    p = p x n
    n = n / 2
```

احسب كلفة الخوارزمية بالنسبة لعدد عمليات الضرب، وما هي قيمة المقول p بعد انتهاء الخوارزمية ؟

الحل: ليكن i عدد مرات الدخول

$\frac{n}{2^0}, \frac{n}{2^1}, \frac{n}{2^2}, \frac{n}{2^3} \dots \frac{n}{2^{i-1}}$

$\Rightarrow \frac{n}{2^{i-1}} \leq 1 \Rightarrow \frac{n}{2^{i-1}} = 1 \Rightarrow n = 2^{i-1}$

$\Rightarrow i - 1 = \log_2 n \Rightarrow \boxed{i = \log_2 n + 1}$

قيمة المقول p $\quad p = \prod_{j=0}^{i-1} 2^j$

الخوارزميات العودية Recurssive Algo

عامش: العودية: تكرار النفس الظاهرة //.

الهدف العودية: نقول عن هدف ما أو شيء ما إنه عودي إذا عُرف بدلالة نفسه كلياً أو جزئياً.

مثال: الأعداد الطبيعية.

1. الصفر هو عدد طبيعي
2. كل عدد يلي عدد طبيعي هو عدد طبيعي

} عودية كلية

مثال: العاملي،

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

} جزئية.

تكرارية

```
long fi(int n) {
    long r = 1;
    for (int i=2; i<=n, i++)
        r *= i;
    return r;
```

$T(n) = n$

```
long fr (int n) {
if (n==0)    return 1;
else
return (n x fr(n-1));
}.
```

حجم الذاكرة اللازم للخوارزمية

$S(n) = Size\ of\ (long + int)$

$= 32 + 16 = 48\ bit$

$= \frac{48}{8} = 6\ byt$

لو كان n=4 مثلاً فإن

21 = fr(4)

24 = 4 x fr(3)

6 = 3 x fr(2)

2 = 2 x fr(1)

1 = 1 x fr(0)

= 1

هبوط :
وهي مرحلة استدعاءات عودية حتى الوصول إلى القيمة التي نعرف جوابها فوراً
وهي حالة توقف

Stoping case

حالة توقف: هي القيمة التي أعرف الجواب عندها مباشرة

Recurssif Depth

العمق العودي: هو عدد مرات استدعاء التابع العودي لنفسه (خوارزمية عودية لنفسها).

- العمق العودي للمثال السابق هو 4

- أما العمق العودي للدالة fr(n) هو n.

- حساب كلفة الخوارزمية العودية:

عدد عمليات في الدخول الواحد X العمق العودي. ((T(n) = n مثال))

- في حال لم يعلم العمق العودي: return(n x fr(n-1))

$T(n) = 1 + T(n-1)$ // معادلات تراجعية //

$= 1 + [1 + T(n-2)] = 2 + T(n-2) = 2 + [1 + T(n-3)]$

$= 3 + T(n-3) = \text{------} = n + T(n-n)$

$= T(0) + n = 0 + n = n$

n=0 عدد عمليات الضرب عندما

$T(n) = n$

انتهت - ١٥.

/ /

مثال: اكتب الخوارزمية التكرارية لحساب هذه الدالة a^n ; $n, a \in N^*$

```
long int f1 ( int a, int n ) {
        long int r = 1;
      for (int i = 1 ; i <= n ; i++);
      r = r * a ;
        return r;
       }
```

$T(n) = n$ $O(n)$ كلفة الخوارزمية بالنسبة لعدد عمليات الضرب

$S(n) = Size\ of\ (long\ int + int) =$

- لو أردنا حساب خوارزمية العودية لـ a^n حيث $a, n \in N^*$ سمي التابع العودي $f(a,n) = a^n$.

$a^n = a^{n-1} * a$, $f(a,n) = f(a, n-1) * a$

الخوارزمية هي:

```
long int f2 (int a, int n) {
 if (n == 1) return a;
 else return (a * f2 (a, n-1));
}
```

كلفة الخوارزمية بالنسبة لعدد عملياته الضرب

$T(n) = 1 + T(n-1) = 2 + T(n-2) = \dots = n-1 + T(n-(n-1))$

$= T(1) + n - 1 = 0 + n - 1 = n - 1$

عدد عمليات الضرب

$\Rightarrow T(n) = n - 1$ $O(n)$

// $S(n) = (n-1)\ Siz\ of\ (int, long\ int)$ // هام جدا

يمكن أن نكتب a^n طريقة ثانية وأسرع من السابقة (كما أراد المبرجين العوديين)

$$a^n = \begin{cases} a^{\frac{n}{2}} \cdot a^{\frac{n}{2}} & : n \text{ زوجي} \\ a^{\frac{n-1}{2}} \cdot a^{\frac{n-1}{2}} \cdot a & : n \text{ فردي} \\ a & : n = 1 \end{cases}$$

```
long int f3(int a, int n){
  if(n==1) return a;
  else{ long int r;
  if (n%2==0){   r=f3(a, n/2);
                 return(r*r); return(f3(a,n/2),f3(a,n/2))
                                  ↑ هذا الكلام خاطئ
              }
  else {
        r = f3(a, (n-1)/2);
        return (r*r*a);
  }//else      }//else      }//
```

خوارزمية

كلفتها لعدد عمليات الضرب.

عمليات الضرب

$$T(n) = 2 + T\left(\frac{n-1}{2}\right) = 2 + [2 + T\left(\frac{n-3}{4}\right)]$$ (-1 ثم ÷2)

$$= 4 + T\left(\frac{n-3}{4}\right) = 4 + [2 + T\left(\frac{n-7}{8}\right)]$$

$$= 6 + T\left(\frac{n-7}{8}\right) \dots = 2i + T\left(\frac{n-(2^i-1)}{2^i}\right)$$

لازم تساوي 1

نجوز $2i+1$

$$\frac{n-(2^i-1)}{2^i} = 1 \Rightarrow 2^i = n-(2^i-1) \Rightarrow 2^{i+1} = n+1$$

$$\Rightarrow i+1 = \log_2(n+1) \Rightarrow i = \log_2(n+1) - 1$$

$$\Rightarrow T(n) = T(1) + 2\log_2(n+1) - 2 = 2\log_2(n+1) - 2$$

$O(\log_2(n))$

// a^{1000} بكارتيبها 1000 عملية ضرب

// a^{1000} بالعودي اذا كان $a=2$ فقط و بيكون بدها و مرات تقريباً //

مثال: اكتب خوارزمية عودية لحساب $n!$ حيث n عدد طبيعي سدخل بحيث لا تتجاوز كلفتها بالنسبة لعدد عمليات الضرب $(n-2)$ //لاحظ//

```
long int fact(int n){
if (n == 0) return 1;
else if (n == 1) return 1;
```

```
else if (n == 2) return 2;
else return (n x fact(n-1));
}
```

$T(n) = 1 + T(n-1) = \dots = n-2 + T(n-(n-2))$

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

مثال:

جد الزمنية متكونة من عودية وتكرارية.

```
long int fact (int n, int s) {
    if (n <= s) {
        long int r = 1;
        for (int i = 1; i <= n; i++)
            r *= i;
        return r;
    }
    else return (n x fact(n-1, s));
}
```

عملية ضرب

عملية ضرب n-s

$\uparrow T(n)$

$S(n) = -(n-s)$ long int $+ 1$ long int $+$ int