♥ Java ♥

23/10/2017 الاثنين المحاضرة الثامنة

اكتب خوارزمية عودية لحساب x^n حيث $x, n \in \mathbb{N}$

$x^n = x.x.x.....x$ (مرة n)

$x^n = x.x^{n-1} = x.(x.x^{n-2})$

⋮

$x^n = x.x.x.....x.(1) \rightarrow x^0$

```
int power1 (int x, int n){
    if(n==0) return (1);
else return (x* power1(x,n-1);
                    }
```

حساب كلفة الخوارزمية بالنسبة لعدد عمليات الضرب

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

$= T(n-2) + 2 = T(n-3) + 3 = ...$

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

$T(0) = 0$ لا يوجد أي عملية ضرب حينما $n=0$

العمق العودي = n كون لدينا في كل عملية ضرب واحدة

حل الخوارزمية السابقة بالطريقة التكرارية:

```
int R=1;
for(i=1;i<=n;i++)
    R= R* x
```

الكلفة: $T(n) = n = O(n)$

عدنا إلى نفس المشكلة لم تتغير في الخوارزمية العودية الكلفة نفسها في الخوارزمية التكرارية مع زيادة في حجم الذاكرة المحجوزة ويعود ذلك لأننا رأينا العودية بشكل خاطئ.

إذا أردنا اختصار الحل فسيكون بالشكل الرياضي للعودية

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & : n \text{ زوجي} \\ x^{(n-1)/2} \cdot x^{(n-1)/2} \cdot x & : n \text{ فردي} \\ 1 & : n = 0 \end{cases}$$

برنامج سي:

```
int power2(int x, int n){
 if(n==0) return(1);
   else if(n%2 ==0){
  return(power2(x,n/2)*power2(x,n/2));
                    }
else{return(power2(x,(n-1)/2)*power2(x,(n-1)/2)*x);
   }
                    }
```

حساب الكلفة:

$$T(n) = 2T(\frac{n}{2}) + 1$$
$$= 2(2T(\frac{n}{4}) + 1) + 1 = 4T(\frac{n}{4}) + 3$$
$$= 4(2T(\frac{n}{2^3}) + 1) + 3 = 2^3 T(\frac{n}{2^3}) + 7$$
$$\vdots$$
$$= 2^i T(\frac{n}{2^i}) + 2^i - 1$$

$$\begin{cases} T(\frac{n}{2^i}) = T(1) \\ \frac{n}{2^i} = 1 \Rightarrow n = 2^i \\ i = \log_2 n \end{cases}$$

$$T(n) = 2^{\log_2(n)} T(1) + 2^{\log_2(n)} - 1 = nT(1) + n - 1$$
$$= 2n - 1 = O(n)$$

حسبنا كلفة المشكلة ولم نستفد أي شيء

برنامج ثنيي

```
int power2(int x, int n){
if(n==0) return(1);
else if(n%2 ==0){ int a=power2(x,n/2);
                  return(a*a);}
else{int a=power2(x,(n-1)/2);
     return(a*a*x);}}
```

حساب الكلفة:

$$T(n) = T(\frac{n}{2}) + 1$$
$$= T(\frac{n}{2^2}) + 2$$
$$= T(\frac{n}{2^3}) + 3 \quad \dots\dots$$
$$= T(\frac{n}{2^i}) + i$$

$\left\{ T(\frac{n}{2^i}) = T(1) \Rightarrow \frac{n}{2^i} = 1 \right.$

$i = \log_2 n$

$$= T(1) + \log_2 n$$
$$T(n) = \log_2 n + 1 = O(\log n)$$

العمق العودي هو $\log n$ وبالتالي حجم الذاكرة هو $\log n$

مثال: اكتب خوارزمية عودية لحساب $n!$ بحيث لا تتجاوز كلفتها بالسبة لعدد عمليات الضرب $n-2$.

```
int fact(int n) {
  if(n==0) return(1);
  else return(n*fact(n-1));
}
```

$T(n) = n$

كي تكون $T(n) = n-2$ تكون الدالة بالشكل:

```
int fact(int n){
  if(n==0 || n==1) return(1);
   else (n==2) return(2);
    else return(n*fact(n-1));
}
```

وظيفة: اكتب خوارزمية عودية لحساب $n!$ بحيث لا يتجاوز العمق العودي $n-5$

انتهت المحاضرة