

Exercicis ADA

Anàlisi i Disseny d'Algoritmes

Tema 1. Anàlisi d'Algorismes

Demostrar: $\sum_{i=1}^k i2^{-i} = 2 - k2^{-k} - 2^{1-k}$

Per a $k=1$,

$$2^{-1} = 2 - 2^{-1} - 2^{1-1}$$

$$2^{-1} = 2 - 2^{-1} - 1$$

$$2^{-1} = 1 - 2^{-1}$$

$$2^{-1} = 2^{-1}$$

Per a $k \geq 1$,

Hipòtesi d'inducció

$$\underline{2^{-1} + 2 \cdot 2^{-2} + 3 \cdot 2^{-3} + 4 \cdot 2^{-4} + \dots + k \cdot 2^{-k}} = \underline{\underline{2 - k \cdot 2^{-k} - 2^{1-k}}}$$

Demostració

$$\begin{aligned} \underline{2^{-1} + 2 \cdot 2^{-2} + 3 \cdot 2^{-3} + 4 \cdot 2^{-4} + \dots + k \cdot 2^{-k} + (k+1) \cdot 2^{-k+1}} &= 2 - (k+1) \cdot 2^{-k+1} - 2^{1-(k+1)} \\ \underline{\underline{2 - k \cdot 2^{-k} - 2^{1-k}} + (k+1) \cdot 2^{-k+1}} &= 2 - (k+1) \cdot 2^{-k+1} - 2^{1-(k+1)} \\ -k \cdot 2^{-k} - 2^{1-k} + k \cdot 2^{-k-1} + 2^{-k-1} &= -k \cdot 2^{-k-1} - 2^{-k-1} - 2^{-k} \\ -k \cdot 2^{-k} - 2^{1-k} + 2^1 \cdot k \cdot 2^{-k-1} + 2^1 \cdot 2^{-k-1} &= -2^{-k} \\ -k \cdot 2^{-k} - 2^{1-k} + k \cdot 2^{-k} + 2^{-k} &= -2^{-k} \\ 2^{-k} (-k - 2 + k + 1) &= -2^{-k} \\ 2^{-k} (-1) &= -2^{-k} \\ -2^{-k} &= -2^{-k} \end{aligned}$$

1.3

Demostrar: $\sum_{i=0}^n i = \frac{n(n+1)}{2}$

Per a $n=1$,

$$1 = \frac{2 \cdot 1}{2}$$
$$1 = 1$$

Per a $n \geq 1$,

Hipòtesi d'inducció

$$\underline{1 + 2 + 3 + 4 + 5 + \dots + n} = \underline{\underline{\frac{n(n+1)}{2}}}$$

Demostració

$$\underline{1 + 2 + 3 + 4 + 5 + \dots + n + (n+1)} = \frac{(n+1)(n+2)}{2}$$
$$\underline{\underline{\frac{n(n+1)}{2} + n + 1}} = \frac{n^2}{2} + \frac{2 \cdot n}{2} + \frac{n}{2} + 1$$
$$\frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$
$$\frac{n(n+1)}{2} = \frac{n(n+1)}{2}$$

1.4

Demostrar: $\sum_{i=0}^n 2^i = 2^{n+1} - 1$

Per a $n=0$,

$$2^0 = 2^1 - 1$$

$$1 = 2 - 1$$

$$1 = 1$$

Per a $n \geq 0$,

Hipòtesi d'inducció

$$\underline{2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^n} = \underline{\underline{2^{n+1} - 1}}$$

Demostració

$$\begin{aligned} \underline{2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^n} + 2^{n+1} &= 2^{n+2} - 1 \\ \underline{\underline{2^{n+1} - 1}} + 2^{n+1} &= 2^{n+2} - 1 \\ 2^{n+1} + 2^{n+1} &= 2^{n+2} \\ 2^{n+2} &= 2^{n+2} \end{aligned}$$

1.27

$$\bullet T(n) = T(n-1) + \Phi(1)$$

Per substitució repetida:

$$\begin{aligned} T(n) &= T(n-1) + k2 \\ &= T(n-2) + k2 + k2 \\ &= \dots \\ &= T(n-n) + \underbrace{k2 + \dots + k2}_n \\ &= T(0) + n \cdot k2 \end{aligned}$$

$$\begin{aligned} T(n) &= \Phi(1) + n \cdot \Phi(1) \\ &= (n+1)\Phi(1) \\ &= \Phi(n) \end{aligned}$$

Per teorema Màster I:

$$\Phi(n^{k+1}) = \Phi(n^1) = \Phi(n)$$

$$\bullet T(n) = T(n-2) + \Phi(1)$$

Per substitució repetida:

$$\begin{aligned} T(n) &= T(n-2) + k2 \\ &= T(n-4) + k2 + k2 \\ &= \dots \\ &= T(n-n) + \underbrace{k2 + \dots + k2}_{n/2} \\ &= T(0) + n/2 \cdot k2 \end{aligned}$$

$$\begin{aligned} T(n) &= \Phi(1) + n/2 \cdot \Phi(1) \\ &= (n/2 + 1)\Phi(1) \\ &= \Phi(n) \end{aligned}$$

Per teorema Màster I:

$$\Phi(n^{k+1}) = \Phi(n^1) = \Phi(n)$$

a=1 k=0

$$\bullet T(n) = T(n-2) + \Phi(n)$$

Per substitució repetida:

$$\begin{aligned} T(n) &= T(n-1) + k2 \\ &= T(n-2) + k2 + k2 \\ &= \dots \\ &= T(n-n) + \underbrace{k2 + \dots + k2}_n \\ &= T(0) + n \cdot k2 \end{aligned}$$

$$\begin{aligned} T(n) &= \Phi(1) + n \cdot \Phi(n) \\ &= n\Phi(n) + \Phi(1) \\ &= \Phi(n^2) \end{aligned}$$

Per teorema Màster I:

$$\Phi(n^{k+1}) = \Phi(n^{1+1}) = \Phi(n^2)$$

$$\bullet T(n) = 2T(n-1) + \Phi(1)$$

Per substitució repetida:

$$\begin{aligned} T(n) &= 2T(n-1) + k2 \\ &= 4T(n-2) + k2 + k2 \\ &= \dots \\ &= 2^n T(n-n) + \underbrace{k2 + \dots + k2}_n \\ &= 2^n T(0) + n \cdot k2 \end{aligned}$$

$$\begin{aligned} T(n) &= 2^n \Phi(1) + n \cdot \Phi(1) \\ &= \Phi(2^n) + \Phi(n) \\ &= \Phi(2^n) \end{aligned}$$

Per teorema Màster I:

$$\begin{aligned} &a=2 \quad k=0 \quad c=1 \\ \Phi(2^{n/c}) &= \Phi(n^{n/1}) = \Phi(2^n) \end{aligned}$$

1.28

$$\bullet T(n) = 2T(n/2) + \Phi(1)$$

Per substitució repetida:

$$\begin{aligned} T(n) &= 2T(n/2) + k^2 \\ &= 2(2T(n/4) + k^2) + k^2 \\ &= 4T(n/4) + 2k + k \\ &= 4(2T(n/8) + k^2) + 3k^2 \\ &= 8T(n/8) + k^2 + 3k^2 \\ &= \dots \\ &= 2^{\log_2(n)} T(n/n) + (n-1)k^2 \\ &= nT(1) + (n-1)k^2 \\ &= \Phi(n) \end{aligned}$$

$$\begin{aligned} T(n) &= \Phi(1) + n/2 \cdot \Phi(1) \\ &= (n/2 + 1)\Phi(1) \\ &= \Phi(n) \end{aligned}$$

Per teorema Màster II:

$$\begin{aligned} a=2 \quad b=2 \quad k=0 \\ \Phi(n^{\log_2 2}) = \Phi(n) \end{aligned}$$

$$\bullet T(n) = 2T(n/2) + \Phi(n^2)$$

Per teorema Màster II:

$$\begin{aligned} a=2 \quad b=2 \quad k=2 \\ \Phi(n^2) \end{aligned}$$

$$\bullet T(n) = 2T(n/2) + O(n^2)$$

Per teorema Màster II:

$$\begin{aligned} a=2 \quad b=2 \quad k=1 \\ O(n \cdot \log(n)) \end{aligned}$$

$$\bullet T(n) = 4T(n/2) + n^2$$

Per teorema Màster II:

$$a=4 \quad b=2 \quad k=2 \\ \Phi(n^2 \cdot \log(n))$$

$$\bullet T(n) = T(9n/10) + \Phi(n)$$

Per teorema Màster II:

$$a=1 \quad b=10/9 \quad k=1 \\ \Phi(n)$$

1.37

$n^{\log_4 x} < n^{\log_2 7}$ per tant $\log_4 x = \log_2 7$ només si $x < 4^{\log_2 7}$
 $x = 49$, però ens demanen el mínim amb la qual cosa hem d'agafar el 48

Tema 2. Divideix i Venceràs

2.5

```
int busctern(a:vector, int prim, int ult, int x)
{
    int terc;
    if (prim>=ult)
    {
        return (a[ult]=x);
    }
    terc:=(ult-prim+1)/3;
    if (x=a[prim+terc])
    {
        return true;
    }
    else if (x<a[prim+terc])
    {
        return busctern(a, prim, prim+terc-1, x);
    }
    else if (x=a[ult-terc])
    {
        return true;
    }
    else if (x<a[ult-terc])
    {
        return busctern(a, prim+terc+1, ult-terc-1, x);
    }
    else
    {
        return busctern(a, ult-terc+1, ult, x);
    }
}
```

Asimptòticament té el mateix cost que l'algorisme de cerca dicotòmica, ja que fa tres crides recursives, per tant enlloc de base 2, tindrem base 3.

2.29

```
function maxsum(A, l, r)
{
    if (l>r)
    {
        return 0;
    }
    else if (l=r)
    {
        return (A[l]>0)?A[l]:0
    }
    else
    {
        m:=(l+r)/2;
        lmax:=0;
        sum:=0;
        for (i:=m downto l)
        {
            sum:=sum+A[i];
            if (sum>lmax)
            {
                lmax:=sum;
            }
        }
        rmax:=0;
        sum:=0;
        for (i:=m+1 to r)
        {
            sum:=sum+A[i];
            if (sum>rmax)
            {
                rmax:=sum;
            }
        }
        return max(maxsum(l, m), maxsum(m+1, r), lmax+rmax);
    }
}
```