Ficha 2

① Contagem

1.

→ Identificar MC e PC em termos do nº de comparações entre elems. do array e em termos do nº trocas efetuadas.

→ Calcule nº comparações entre elems do array efetuadas nesses casos identificados.

a)
```
void bubbleSort (int v[], int N){
    int i, j;
    for (i=N-1; i>0; i--)
        for(j=0; j<i; j++)
            if(v[j] > v[j+1])  swap (v, j, j+1);
}
```

> ñ tem MC ou PC pois são sempre feitas o mesmo nº de comparações

$T_>(N) = \sum_{i=1}^{N-1} \sum_{j=0}^{i-1} 1$ → comparação entre elems. do array, tem tempo constante

$= \sum_{i=1}^{N-1} i - \cancel{1} - 0 + \cancel{1} = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \Theta(N^2)$

Swap já tem MC e PC pois depende da condição $v[j] > v[j+1]$

quando array está ordenado crescente, ou seja, $v[j] < v[j+1]$ (if sempre False)

quando array está ordenado decrescente, ou seja, $v[j] > v[j+1]$ (if sempre True)

$T_{swap}^{MC}(N) = 0 = \Theta(1)$

$T_{swap}^{PC}(N) = \sum_{i=1}^{N-1} \sum_{j=0}^{i-1} 1 = \frac{N(N-1)}{2} = \Theta(N^2)$

b) void iSort (int v[], int N){
```
int i, j;
for (i=1; i<N; i++)
    for (j=i; j>0 && v[j-1] > v[j]; j--)
        Swap (v, j, j-1);
```

> agora tem MC e PC pois está dentro do for, logo:

$$T_{>}^{MC}(N) = \sum_{i=1}^{N-1} 1 = N-1-\cancel{1}+\cancel{1} = \underline{\Theta(N)}$$

$$T_{>}^{PC}(N) = \sum_{i=1}^{N-1}\sum_{j=1}^{i} 1 = \sum_{i=1}^{N-1} i-\cancel{1}+\cancel{1} = \frac{(N-1)N}{2} = \underline{\Theta(N^2)}$$

Swap tem MC e PC também:

quando $v[j-1] <= v[j]$

quando $v[j-1] > v[j]$

$$T_{Swap}^{MC}(N) = 0 = \underline{\Theta(1)}$$

$$T_{Swap}^{PC}(N) = \sum_{i=1}^{N-1}\sum_{j=1}^{i} 1 = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \underline{\Theta(N^2)}$$

2.

```
int mult1(int x, int y){
    // pre: x>=0
    int a=x, b=y, r=0;
    while (a>0){
        r = r+b;
        a = a-1;
    }
    //pos: r == x*y
    return r;
}
```

```
int mult2(int x, int y) {
    // pre: x>=0
    int a=x, b=y, r=0;
    while (a>0){
        if (a%2 == 1) r = r+b;
        a = a/2; b = b*2;
    }
    // pos: r == x*y
    return r;
}
```

$\boxed{X}$ sempre feito

3* $\boxed{?}$

+ nº ocorrências ímpares

Contar nº de vezes que as operações primitivas (+ - *2 /2 %2) contadas no corpo do ciclo são executadas no pior caso.

Tamanho do input é o nº bits necessários para representar nºs inteiros, como argumento.

N=4 bits

| 1 | 1 | 1 | 1 | $= 2^4-1 = 15$

| 1 | 0 | 0 | 0 | $= 2^3 = 8$

$\{2^{N-1}, 2^N-1\}$

$\boxed{\text{mult1}}$

→ tanto para + como para - ñ existe PC e MC pois só depende de a>0 para ser executado o corpo do ciclo

nº comparações de + e - é de $\underline{X}$

$T_-^{MC}(N) = T_+^{MC}(N) = 2^{N-1}$

$T_-^{PC}(N) = T_+^{PC}(N) = 2^N-1$

→ assimptoticamente

$T_+(N) = \Theta(2^N)$

$\boxed{\text{mult2}}$

nº comparações $= \boxed{X}$

→ if(False) sempre até $\frac{1}{1}=1$

$T_+^{MC}(N) = 1$

| 0 | 0 | 0 | 0 | 1 |

$T_+^{PC}(N) = N = \Theta(N)$

N-1 ... 0

| 0 | 1 | 1 | 1 | $\xrightarrow{a=\frac{a}{2}}$ | 0 | 0 | 1 | 1 |

faz shift para a direita

bit + significativo

Logo vamos ter N shifts para a direita

Como * faz o mesmo nº de execuções de /
então $T_*(N) = T_/(N) = N = \Theta(N)$

3.

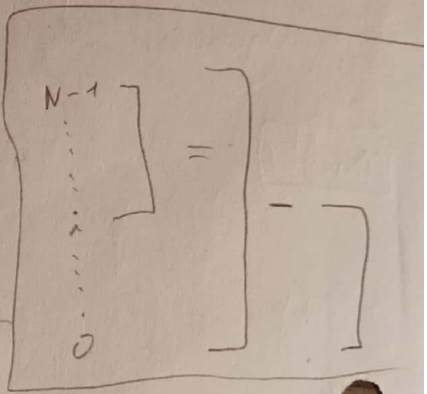
```c
int maxSoma (int v[], int N){
    int i, j, r=0, m;
    for (i=0; i<N; i++)
        for (j=i; j<N; j++){
            m = Soma(v,i,j);
            if (m>r) r=m;
        }
    return r;
}
```

```c
int Soma (int v[], int a,
          int b) {
    int r=0, i;
    for (i=a; i<=b; i++)
        r = r+v[i];
    return r;
}
```

a) $\boxed{n^2 \text{ acessos ao array argumento}}$

Soma não tem MC ou PC

$$T_{Soma}(a,b) = \sum_{i=a}^{b} 1 = b-a+1$$



$$T_{maxSoma}(N) = \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} T_{Soma}(i,j)$$

$$= \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} (j-i+1)$$

$$= \sum_{i=0}^{N-1} \left(\sum_{j=i}^{N-1} j - \sum_{j=i}^{N-1} i-1 \right)$$

$$= \sum_{i=0}^{N-1} \left(\sum_{j=1}^{N-1} j - \sum_{j=1}^{i} j - \sum_{j=i}^{N-1} i-1 \right)$$

$$= \sum_{i=0}^{N-1} \left(\frac{(N-1)N}{2} - \frac{(i-1+1)(i-1)}{2} - (i-1) \sum_{j=i}^{N-1} 1 \right)$$

$$= \sum_{i=0}^{N-1} \left(\frac{N^2-N}{2} - \frac{i^2+i}{2} - (i-1)(N-\cancel{1}-i+\cancel{1}) \right)$$

$$= \sum_{i=0}^{N-1} \left(\frac{N^2-N}{2} - \frac{i^2+i}{2} - (i-1)(N-i) \right) \quad -iN + i^2 + N - i$$

$$= \sum_{i=0}^{N-1} \left(\frac{N^2-N}{2}\right) - \sum_{i=0}^{N-1} \frac{i^2-i}{2} - \sum_{i=0}^{N-1} iN + \sum_{i=0}^{N-1} i^2 + \sum_{i=0}^{N-1} N - \sum_{i=0}^{N-1} i$$

$$= \frac{N^2-N}{2} \cdot \underbrace{\sum_{i=0}^{N-1} 1}_{N-0+1} - \frac{1}{2} \cdot \underbrace{\sum_{i=0}^{N-1} i^2}_{?} + \frac{1}{2} \cdot \sum_{i=0}^{N-1} i - N \cdot \sum_{i=0}^{N-1} i + \underbrace{\sum_{i=0}^{N-1} i^2}_{?} + N \cdot \underbrace{\sum_{i=0}^{N-1} 1}_{N} - \underbrace{\sum_{i=0}^{N-1} i}_{\frac{(N-1)N}{2}}$$

$$\frac{(N^2-N)N}{2} = \frac{N^3-N^2}{2}$$

$$= \frac{N^2-N}{2} \cdot N - \frac{1}{2} \sum_{i=0}^{N-1} i^2 + \frac{1}{2} N - N \cdot \frac{(N-1) \cdot N}{2} + \sum_{i=0}^{N-1} i^2 + N - \frac{(N-1)N}{2}$$

$$= \frac{N^3-N^2}{2} + \frac{1}{2} \sum_{i=0}^{N-1} i^2 + \frac{1}{2} N - \frac{N^3+N^2}{2} + N - \frac{N^2-N}{2}$$

$$= \cancel{\frac{N^3}{2}} - \cancel{\frac{N^2}{2}} + \cancel{\frac{1}{2} N} - \cancel{\frac{N^3}{2}} + \cancel{\frac{N^2}{2}} + N - \frac{N^2}{2} - \cancel{\frac{N}{2}} + \frac{1}{2} \sum_{i=0}^{N-1} i^2$$

$$= N - \frac{N^2}{2} + \underbrace{\frac{1}{2} \sum_{i=0}^{N-1} i^2}_{\underbrace{\frac{1}{2} \cdot \left(\sum_{i=1}^{N-1} i^2 + \sum_{i=0}^{1} i^2 \right)}_{\underbrace{\frac{1}{2} \cdot \frac{(N-1)N(2(N-1)+1)}{6} + 1}_{\frac{1}{12} \cdot (N^2-N)(2N-1) = \frac{1}{12} \cdot 2N^3 - N^2 - 2N^2 + N}}}$$

$$= N - \frac{N^2}{2} + \frac{2N^3 - 3N^2 + N}{12} = \Theta(N^3)$$

④

```c
int crescente (int v[], int N){
    int i;
    for (i=1; i<N; i++)
        if ( v[i] < v[i-1]) break;
    return i;
}
```

```c
int maxCresc (int v[], int N) {
    int r = 1, i=0, m;
    while (i<N-1) {
        m = crescente (v+i, N-i);
        if (m > r) r=m;
        i++;
    }
    return r;
}
```

MC e PC de maxCresc em termos de $\boxed{\text{nº de Comp. entre elems. do array}}$

Vamos 1º analisar a função crescente → mais específico v[1] e v[0]

no MC (v[i] < v[i-1] e faz break) ⟶ $\overline{T}^{MC}_{crescente}(N) = 1$

no PC (v[i] >= v[i-1] sempre) ⟶ $\overline{T}^{PC}_{crescente}(N) = \sum_{i=1}^{N-1} 1 = N-1\cancel{+1}\cancel{-1}$

ou

v[i] >= v[i-1] com i=[0... N-2]

a última comparação é feita à mesma em caso de falhar.

$$\overline{T}^{MC}_{maxCres}(N) = \sum_{i=0}^{N-2} 1 = N-2-0+1 = N-1$$

$$\overline{T}^{PC}_{maxCres}(N) = \sum_{i=0}^{N-2} N-1 = N\sum_{i=0}^{N-2} 1 - \sum_{i=0}^{N-2} 1$$

$$= N.(N-1) - N-1$$

$$= N^2 - 2N - 1 = \underline{\Theta(N^2)}$$

② Definições Recursivas

resolução ensinada pelo jbb

1. Árvores de Recorrência

$T(0)$ = constante

a) $T(n) = k + T(n-1)$ com k constante

$$T(n) = \begin{cases} a & \Leftarrow n < 1 \\ k + T(n-1) & \Leftarrow n \geq 1 \end{cases}$$

$T(0) = a$

$T(1) = k + a$

$T(2) = k + (k+a)$

$T(3) = k + (k + k + a)$

$\vdots$

$T(n) = k + ((n-1)k + a) = \underline{\underline{nk + a}}$

	i
$T(n) - k$	0
$T(n-1) - k$	1
$T(n-2) - k$	2
$\boxed{T(n-i) - k}$	i
$T(1) - k$	u
$T(0) - a$	N

$1 = n - u$
$\Leftrightarrow u = n-1$

$$T(n) = \sum_{i=0}^{u} k + a$$

$$= \sum_{i=0}^{n-1} k + a = \underline{\underline{nk + a}} = \underline{\Theta(n)}$$

b) $T(n) = k + T\left(\frac{n}{2}\right)$ com k constante

$$T(n) = \begin{cases} a & \Leftarrow n < 1 \\ k + T\left(\frac{n}{2}\right) & \Leftarrow n \geq 1 \end{cases}$$

$T(0) = a$

$\frac{1}{2} = 0{,}5 = 0$ $\quad T(1) = k + a$

$T(2) = k + k + a = 2k + a$

$T(4) = k + 2k + a = 3k + a$

$T(8) = k + 3k + a = 4k + a$

$T(16) = k + 4k + a = 5k + a$

$\vdots$

	i
$T(N) - k$	0
$T\left(\frac{N}{2}\right) - k$	1
$T\left(\frac{N}{4}\right) - k$	2
$T\left(\frac{N}{8}\right) - k$	3
$\boxed{T\left(\frac{N}{2^i}\right) - k}$	i
$T(1) - k$	u
$T(0) - a$	N

$1 = \frac{N}{2^u} \Leftrightarrow 2^u = N$
$\Leftrightarrow u = \log_2 N$

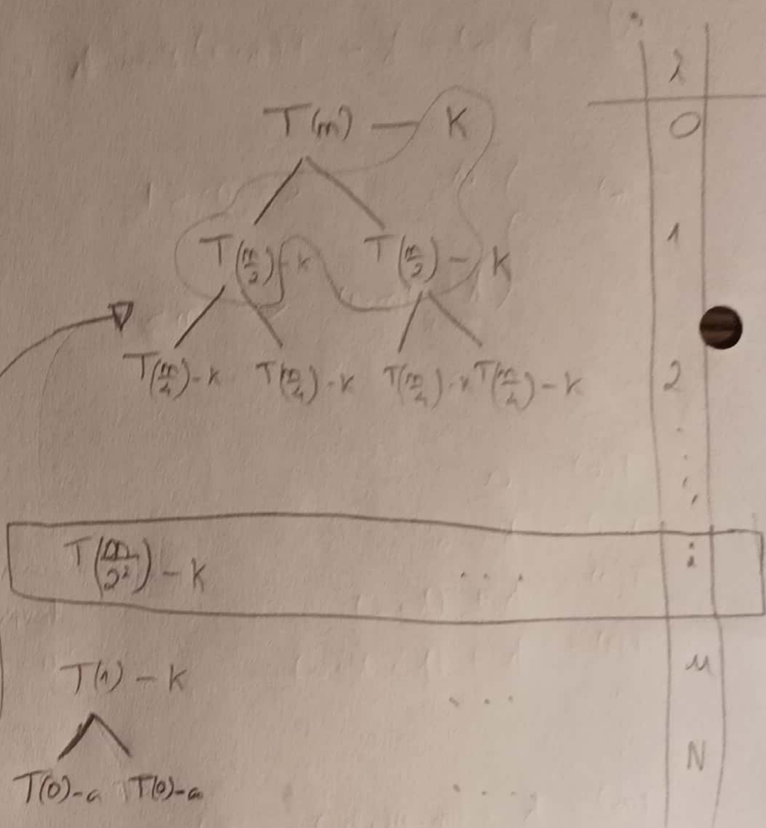
$$T(n) = \sum_{i=0}^{u} k + a = \sum_{i=0}^{\log_2 N} k + a = (*)$$

$$\sum_{i=0}^{\log_2 N} k + a \overset{(+)}{=} k \sum_{i=0}^{\log_2 N} 1 + a = \underline{k(\log_2 N + 1) + a}$$

$$= k(\log_2 N) + k + a = \Theta(\log_2 N)$$

c) $T(n) = k + 2 * T(\frac{n}{2})$ com k constante

$$T(n) = \begin{cases} a & \Leftarrow n < 1 \\ k + 2 * T(\frac{n}{2}) & \Leftarrow n \geq 1 \end{cases}$$



$T(0) = a$

$T(1) = k + 2a$

$T(2) = k + 2(k + 2a) = 3k + 4a$

$T(4) = k + 2(3k + 4a) = 7k + 8a$

$\vdots$

$T(n) = (2n-1)k + 2na$??

por intuição, vamos testar pela árvore de recorrência

tava certa a minha intuição ✓

$$T(n) = \sum_{i=0}^{u} 2^i k + 2 \times 2^u \times a = k \sum_{i=0}^{\log_2 n} 2^i + 2 \times 2^{\log_2 n} \times a$$

$$= \underline{k \times (2n-1) + 2na} = \Theta(n)$$

2^i folhas em cada nível

logo, se em u temos 2^u folhas, em $N = u+1$ nível temos 2^{u+1} folhas

Soma dos k (logo de i's por nível)

$i=0$	$1k = 2^0 k$	
$i=1$	$2k = 2^1 k$	
$i=2$	$4k = 2^2 k$	
$i=3$	$8k = 2^3 k$	
$\vdots$	$\vdots$	
$i=i$	$2^i k$	
$i=u$	$2^u k$	
$i=N$	$2^{u+1} k = 2^{u+1} a$	pois só é com k

$$\boxed{\sum_{i=0}^{\log_2 N} 2^i = \frac{2^{\log_2 N + 1} - 1}{2 - 1} = 2 \cdot 2^{\log_2 N} - 1 = 2N - 1}$$

$1 = \left(\frac{n}{2^u}\right) \Leftrightarrow 2^u = n$

$\Leftrightarrow u = \log_2 n$

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

$T(0) = a$

$T(1) = 1 + a$ $\quad +1$

$T(2) = 2 + 1 + a = 3 + a$ $\quad +2$

$T(3) = 3 + 3 + a = 6 + a$ $\quad +3$

$T(4) = 4 + 6 + a = 10 + a$ $\quad +4$

$\vdots$

$T(n) = ?$

		i
$T(n)$	n	0
$T(n-1)$	$n-1$	1
$T(n-2)$	$n-2$	2
$T(n-3)$	$n-3$	3
$\vdots$		
$T(n-i)$	$n-i$	i
$T(1)$	1	u
$T(0)$	a	N

$$T(n) = \sum_{i=0}^{u} n-i + a$$

$$= n\sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i + a$$

$$= n\,n - \frac{(n-\cancel{1}+\cancel{1})(n-1+0)}{2} + a$$

$$= n^2 - \frac{n(n-1)}{2} + a \quad ? \quad \frac{n^2}{2} + \frac{n}{2} + a$$

$$= n^2 - \frac{n^2 - n}{2} + a = \Theta(n^2)$$

$1 = n - u$

$\iff u = n - 1$

e) $T(n) = n + T\left(\frac{n}{2}\right)$

$T(0) = a$

$T(1) = 1 + a$

$T(2) = 2 + 1 + a = 3 + a$

$T(4) = 4 + 3 + a = 7 + a$

$\vdots$

$T(n) = \underbrace{2n - 1 + a}$

intuição, Vamos verificar

árvore		i
$T(n)$	n	0
$T\left(\frac{n}{2}\right)$	$\frac{n}{2}$	1
$T\left(\frac{n}{4}\right)$	$\frac{n}{4}$	2
$T\left(\frac{n}{8}\right)$		3
$T\left(\frac{n}{2^i}\right)$		i
$T(1)$	1	u
$T(0)$	a	N

$$T(n) = \sum_{i=0}^{u} \frac{n}{2^i} + a$$

$$= n \sum_{i=0}^{\log_2 n} \frac{1}{2^i} + a$$

?? n sei se está certo

$$= n\left(2 + \frac{1}{n}\right) + a$$

$$= 2n + 1 + a = \Theta(n)$$

$1 = \frac{n}{2^u} \Leftrightarrow 2^u = n$

$\Leftrightarrow u = \log_2 n$

$$\underbrace{\underset{1}{\frac{1}{2^0}} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} \cdots}_{\simeq 2} + \frac{1}{n}$$

f) $T(n) = n + 2 * T(\frac{n}{2})$

$T(0) = a$

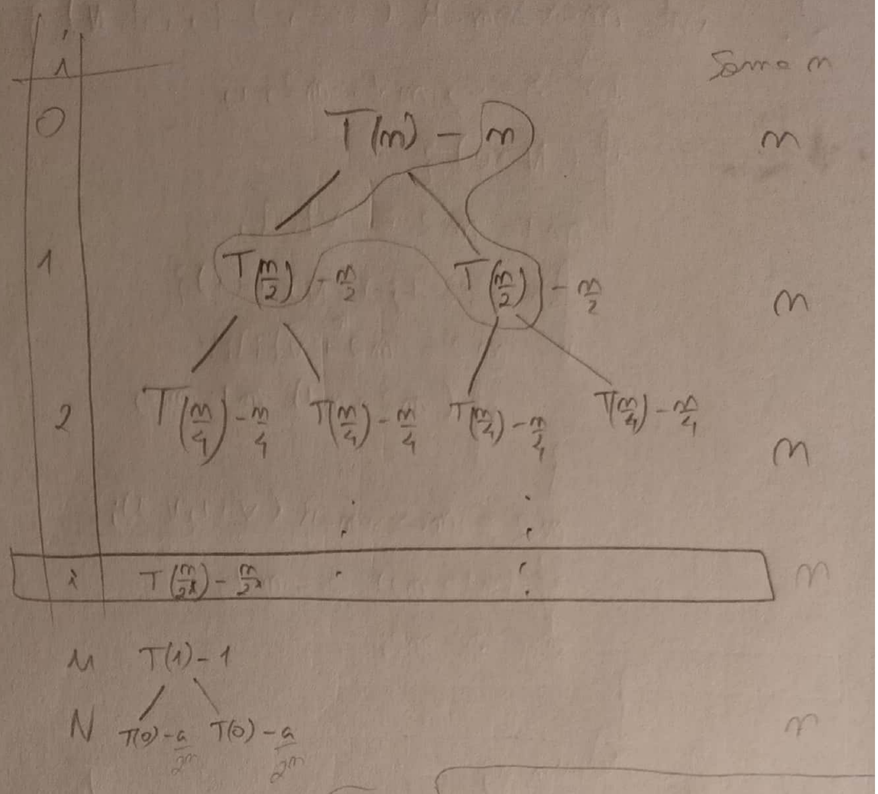
$T(1) = 1 + 2a$

$T(2) = 2 + 2 + 4a = 4 + 4a$

$T(4) = 4 + 8 + 8a = 12 + 8a$

$T(8) = 8 + 24 + 16a = 32 + 16a$

$\vdots$



2^n vezes

$2^n \times \frac{a}{2^n}$

$1 = \frac{n}{2^u} \Leftrightarrow 2^u = n$

$\Leftrightarrow u = \log_2 n$

$$T(n) = \sum_{i=0}^{\log_2 n} n + a$$

$$= n \sum_{i=0}^{\log_2 n} 1 + a$$

$$= n(\log_2 n + 1) + a = \Theta(n \log_2 n)$$

$$(= n\log_2 n + n + a)$$

2.
```c
int maxSomaR (int v[], int N) {
    int r=0, m1, m2, i;
    if (N>0) {
        m1 = m2 = v[0];
        for (i=1; i<N; i++){
            m2 = m2 + v[i];
            if (m2 > m1) m1 = m2;
        }
        m2 = maxSomaR (v+1, N-1);
        if (m1 > m2) r = m1;
        else r = m2;
    }
    return r;
}
```

(1)

(N-1)

Não tem PC ou MC
logo só existe 1
possível recorrência
do algoritmo

nº acessos array argumento

Como recorrência

$$T_{maxSoma}(N) = \begin{cases} 0 & \Leftarrow N == 0 \\ N + T_{maxSoma}(N-1) & \Leftarrow N > 0 \end{cases}$$

$T_{maxSoma}(N)$ — N	0
$T_{maxSoma}(N-1)$ — $N-1$	1
$T_{maxSoma}(N-2)$ — $N-2$	2
⋮	⋮
$\boxed{T_{maxSoma}(N-i) \text{ — } N-i}$	i
$T(1)$ — 1	u
$T(0)$ — 0	N

$1 = N - u$

$\Leftrightarrow u = N-1$

$$T(N) = \sum_{i=0}^{u} N-i + 0$$

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

$$= N(N-1-0+1) - \frac{N^2-N}{2} = N^2 - \frac{N^2-N}{2} = \frac{N^2-N}{2} = \Theta(N^2)$$

3.

```
void Hanoi (int nDiscos, int esquerda, int direita, int meio) {
    if (nDiscos>0) {
        Hanoi (nDiscos-1, esquerda, meio, direita);
        printf ("mover disco de %d para %d, esquerda, direita);
        Hanoi (nDiscos-1, meio, direita, esquerda);
    }
}
```

$\boxed{\text{nº linhas impressas}}$ → printf

Soma "0's" ultima linha árvore dá 0

$$T_{Hanoi}(N) = \sum_{i=0}^{u} 2^i + \overset{?}{0}$$

$$T_{Hanoi}(N) = \begin{cases} 0 & \Leftarrow N==0 \\ 1 + T_{Hanoi}(N-1) + T_{Hanoi}(N-1) & \Leftarrow N>0 \end{cases}$$

$$= \sum_{i=0}^{N-1} 2^i = \frac{1-2^{N-1+1}}{1-2} = -(1-2^N) = \underline{\underline{2^N - 1}} = \underset{\text{exponencial}}{\underline{\underline{\Theta(2^N)}}}$$

$T_{Hanoi}(N) - 1$

$T_{Hanoi}(N-1) - 1$ $T_{Hanoi}(N-1) - 1$

$T_{Hanoi}(N-2) - 1$ $T_{Hanoi}(N-2) - 1$ $T_{Hanoi}(N-2) - 1$ $T_{Hanoi}(N-2) - 1$

$\boxed{T_{Hanoi}(N-i) - 1}$

⋮

$T(1) - 1$

$T(0) - 0$ $T(0) - 0$

i	Soma "1's"
0	1
1	2
2	4
⋮	⋮
i	$\underline{2^i}$
⋮	
u	$2 \cdot 2^i = 2^{i+1} = 2^u$
N	

$\boxed{1 = N - u \Leftrightarrow u = N-1}$

4.
```
void msort(int v[], int N){
    int m = N/2;
    if (N>1){
        msort(v, m);
        msort(v+m, N-m);
        mergeH(v, N);
    }
}
```

→ em função do tamanho do vetor argumento
rel. recorrência do tempo de execução do msort.

$\boxed{T_{merge}(N) = 2 * N}$

$N - \frac{N}{2}$

$$T_{msort}(N) = \begin{cases} 0 & \Leftarrow N == 1 \\ T_{msort}\left(\frac{N}{2}\right) + T_{msort}\left(N-\frac{N}{2}\right) + T_{merge}(N) & \Leftarrow N > 1 \end{cases}$$

$$\boxed{T_{mSort}(N) = \begin{cases} 0 & \Leftarrow N == 1 \\ T_{mSort}\left(\frac{N}{2}\right) + T_{mSort}\left(\frac{N}{2}\right) + 2N & \Leftarrow N > 1 \end{cases}}$$

$T(N) = 2N(\log_2 N - 1) = \Theta(N \lg N)$

$$T(N) = \sum_{i=0}^{u} 2N + 0$$

$$= 2N \sum_{i=0}^{\log_2 N - 1} 1 = \underline{2N \cdot \log_2 N} = \underline{\Theta(N \log_2 N)}$$

$$2 = \frac{N}{2^u} \Leftrightarrow 2 \times 2^u = N$$
$$\Leftrightarrow \log_2(2^{u+1}) = \log_2 N$$
$$\Leftrightarrow u + 1 = \log_2 N$$
$$\Leftrightarrow u = \log_2 N - 1$$

$T_{mSort}(N) - 2N$

$T_{mSort}\left(\frac{N}{2}\right) - 2\frac{N}{2}$ $\quad$ $T_{mSort}\left(\frac{N}{2}\right) - 2\frac{N}{2}$

$T_{mSort}\left(\frac{N}{4}\right) - 2\frac{N}{4}$ $\quad$ $T_{mSort}\left(\frac{N}{4}\right) - 2\frac{N}{4}$ $\quad$ $T_{mSort}\left(\frac{N}{4}\right) - 2\frac{N}{4}$ $\quad$ $T_{mSort}\left(\frac{N}{4}\right) - 2\frac{N}{4}$

⋮

$\boxed{T_{mSort}\left(\frac{N}{2^i}\right) - 2N}$

$T(2) - 2N$

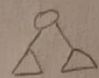
$T(1) - 0 \quad T(1) - 0$

i	Soma
0	$2N$
1	$2N$
2	$\frac{8}{4}N = 2N$
⋮	⋮
i	$2N$
u	$2N$

5.
```c
int altura (ABin a){
    int r=0;
    if (a!= NULL){
        r = 1 + max (altura (a->esq),
                     altura (a->dir));
    }
    return r;
}
```

nº chamadas da função max (por exemplo)

2 casos → ① Árvores equilibradas



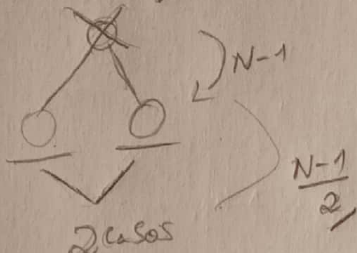
→ ② Árvores Lista

$T_{altura}(N) = \underline{\Theta(N)}$ //

nº nodos árvore

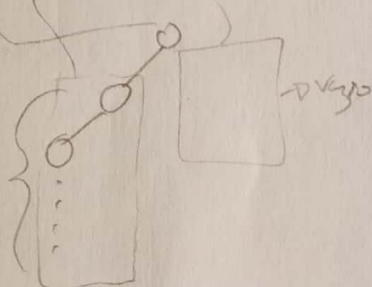
① $T_{altura}(N) = \begin{cases} 0 & \Leftarrow N==0 \\ 1 + 2 * T_{altura}\left(\frac{N-1}{2}\right) & \Leftarrow N>0 \end{cases}$

igual à alínea c) do exercício 1. (daí sabemos que é $\Theta(N)$)



② $T_{altura}(N) = \begin{cases} 0 & \Leftarrow N==0 \\ 1 + T(N-1) + T(0) & \Leftarrow N>0 \end{cases}$

	i
T(N) - 1	0
T(N-1) - 1	1
T(N-2) - 1	2
⋮	⋮
T(N-i)	i
	u
T(1) - 1	
T(0) - 0	N



$1 = N - u$
$\Leftrightarrow u = N-1$

$T_{altura}(N) = \sum_{i=0}^{u} 1 + 0 = \sum_{i=0}^{N-1} 1 = N - \cancel{1} + 0 + 1 = \underline{N} = \Theta(N)$ //

③ Análise de caso médio

1.
```c
int crescente (int v[], int N){
    int i;
    for(i = 1; i<N ; i++)
        if (v[i] < v[i-1]) break;
    return i;
}
```

prob $v[i] < v[i-1]$ é $\boxed{\frac{1}{2}}$

$$\overline{T_{Crescente}}(N) = \sum_{K=1}^{N-1} \frac{1}{2^i} \cdot i + \overbrace{\frac{1}{2^{N-1}} \cdot N-1}^{④}$$

K comparações em que $v[i] < v[i-1]$

K (comparações)	probabilidade
1	$\frac{1}{2}$
2	$(1-\frac{1}{2}) \times \frac{1}{2} = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$
3	$(1-\frac{1}{2}-\frac{1}{4}) \times \frac{1}{2} = \frac{1}{4} \times \frac{1}{2} = \frac{1}{8}$
⋮	⋮
i	$\frac{1}{2^i}$

A função crescente tem MC quando $v[1] < v[0]$

↳ | 20 | 10 | . . . . . |

e tem PC quando $v[i] >= v[i-1]$ nas N-1 1.as posições

↳ | 10 | 20 | 30 | 5 |

ou

| 10 | 20 | 30 | 40 |

④ faz o mesmo nº de comparações

3.
```
int strNdiff (char s1[], char s2[], int N){
    int i;
    for(i=0; i<N && s1[i] && s1[i] == s2[i]; i++);
    return i;
}
```

Caso médio → nº comp elems dos arrays

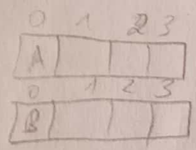
p(2 letras iguais) $= \frac{1}{26}$

k	prob
1	$\frac{1}{26}$
2	$\left(1-\frac{1}{26}\right) \times \frac{1}{26} = \frac{25}{26^2}$
3	$1-\frac{1}{26}-\frac{25}{26^2} = \dots$
⋮	
i	$\underbrace{\left(1-\frac{1}{26}\right)}_{\text{letras diferentes}} \underbrace{\left(\frac{1}{26}\right)^{i-1}}_{\text{letras iguais}}$

$$\overline{T_{strNdiff}}(N) = \sum_{k=1}^{N} \left(\frac{1}{26}\right)^{i-1} \times \left(\frac{1}{26}\right) \times i + \left(\frac{1}{26}\right)^{N} \cdot N$$

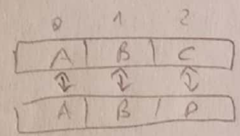
Nesta função temos MC e PC ao nível de comps.

MC → elems. diferentes logo na 1ª posição

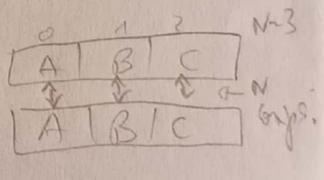


PC → elems iguais nas N-1 1ªs posições

ou



elems todos iguais



5.

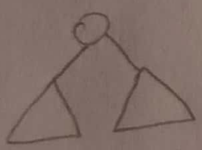
```
int elem (int x, ABin a){
    while (a!=NULL && a-> valor != x) {
        if (x < a->valor) a=a->esq;
        else a = a->dir;
    }
    return (a!=NULL);
}
```

Caso Médio relativo ao nº nodos consultados

Assumir que elem. existe e
tem igual probabilidade de
estar em qualquer nodo

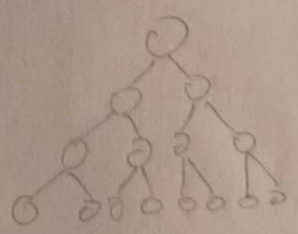
Logo $p = \frac{1}{N}$ (em N nodos)

① Árvore Equilibrada



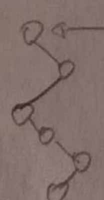
$$\overline{T}_{elem}(N) = \sum_{i=1}^{\log_2 N+1} \frac{1}{N} \cdot i$$

	i	prob
estar na raiz	1	$\frac{1}{N}$
	2	$\frac{1}{N}$
	3	
	⋮	
	⋮	
	λ	



Nível	Nodos
0	2^0
1	2^1
2	2^2
3	2^3

② Árvore Lista



MC $\rightarrow T^{MC}(N) = 1$

PC $\rightarrow T^{PC}(N) = N$

$$\overline{T}(N) = \sum_{i=1}^{N} \frac{1}{N} \cdot i = \frac{1}{N} \cdot \frac{N(N+1)}{2} = \frac{N+1}{2}$$

6.
```
int f(int a[], int N){
    int b = 0;
    for (int i=0; i<N; i++)
        if (a[i] == 1) b++;
    if (b==1) return g(a,N);
    else return 0;
}
```
$T_g(N) = \Theta(2^N)$

PC → é quando o array a contém apenas um 1

MC → " " " " " apenas "0's"

ou ter mais que um 1

tempo percorrer array para verificar se tem apenas um 1

$$T_f(N) = \underbrace{\frac{N}{2^N}}_{prob} \cdot \underbrace{(N + 2^N)}_{custo} + \left(1 - \frac{N}{2^N}\right) \cdot N = \Theta(N)$$

exemplo → $N=4$

| | | | |

0001
0010
0100
1000

→ probabilidade de ter apenas um 1 $= \frac{4}{16} = \frac{N}{2^N}$

e sem ter um 1 apenas? $1 - \frac{N}{2^N}$

$N=5$

| | | | | |

00001
00010
00100
01000
10000

→ $\frac{N}{2^N}$ possibilidades