PROVA IN ITINERE 18/11/2021 A

n° 1

```
algoritmo mystery(array A[1,...,n], int i, int n) → int n
  if (i ≥ n-2) then                                   ] PASSO BASE
    if (n = 1) or (A[i] > A[i+1]) then                ] O(1)
      return A[i]                                     ] O(1)
    else                                              ] O(1)
      retur A[i+1]                                    ] O(1)
    endif
  endif
  int n ← mystery(A, i+1, n)                          ] CHIAMATA RICORSIVA ] T(n)
  if A[i] > n then                                    ] O(1)
    return A[i]                                       ] O(1)
  else                                                ] O(1)
    return n                                          ] O(1)
  endif
```

- IL PASSO BASE E': $(i \geq n-2)$ perché a quel punto non ci interessa neanche effettuare la ricorsione perché abbiamo già scoperto che l'elemento che stiamo cercando è dato o dal contenuto di A[i] o A[i+1] o se n=1, quindi l'array contiene un solo elemento, restituisce quell'elemento
- LA CHIAMATA RICORSIVA E': int n ← mystery(A, i+1, n), la chiamata ricorsiva scorre l'array da sinistra verso destra e restituisce il massimo elemento compreso tra i+1 e n

METODO ITERATIVO

$$\begin{cases} T(n) + O(1) & \text{se } i = n-2 \\ 1 & \text{se } n = 1 \end{cases}$$

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

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

$$T(n-2) = T(n-2) + O(1)$$

$$\vdots$$

$$T(n) = T(n-k) + \overbrace{O(1) + O(1) + O(1)}^{k \cdot O(1)}$$

$n-k=1$; $k=n-1$

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

$T(n) = O(n)$

ES. 2

$$T(n) = \begin{cases} 1 & \text{se} \quad n \leq 1 \\ 1 + 2T\left(\frac{n}{2}\right) & \text{se} \quad n > 1 \end{cases}$$

CASO I MASTER

$\left(n^{\log_2 2 - \varepsilon}\right)$, $\varepsilon = 1$

$(n^{1-1}) = n^0 = 1$

$\Theta(n)$

ITERAZIONE

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

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

$T(n) = 1 + 2\left(1 + 2T\left(\frac{n}{4}\right)\right) \Rightarrow 3 + 4T\left(\frac{n}{4}\right)$

$T\left(\frac{n}{4}\right) = 1 + 2T\left(\frac{n}{8}\right)$

$T(n) = 3 + 4\left(1 + 2T\left(\frac{n}{8}\right)\right) \Rightarrow 7 + 8T\left(\frac{n}{8}\right)$

$\vdots$

$T(n) = 2^k\left(\frac{n}{2^k}\right) + \sum_{j=0}^{k-1} 2^j$

$\frac{n}{2^k} = 1$; $k = \log_2 n$

$$T(n) = 2^{\log_2 n}\left(1\right) + \frac{2^k - 1}{2-1} = n(1) + \frac{2^{\log_2 n} - 1}{1};$$

$T(n) = n + n - 1$; $T(n) = \Theta(n)$

PROVA IN ITINERE 18/11/2021 PROVA B

```
algoritmo mystery (array A[1,...,n], int i, int n) → int
    if i ≥ n then                          ] PASSO BASE ] O(1)
      return A[i]                          ]
    endif                                  ] O(1)
    int m ← (i+n)/2                        ]
    int v1 ← mystery (A, i, m)             ] 2T(n/2)
    int v2 ← mystery (A, m+1, n)           ]
    if v1 ≥ v2 then                        ]
      return v1                            ] O(1)
    else                                   ]
      return v2                            ]
    endif                                  ]
```

$$\begin{cases} c_1 + 2T\left(\frac{n}{2}\right) & \text{se } n < 1 \\ c_2 & \text{se } n = 1 \end{cases}$$

CASO I TEOREMA MASTER

$n^{\log_2 2} = n^1$; $n^{\log_2 2 - \varepsilon} = n^0 = 1$, $\varepsilon = 1$

$\Theta\left(n^{\log_2 2}\right) = \Theta(n)$

METODO ITERAZIONE

$T(n) = c_1 + 2T\left(\frac{n}{2}\right)$

$T\left(\frac{n}{2}\right) = c_1 + 2T\left(\frac{n}{4}\right)$

$T(n) = c_1 + 2\left(c_1 + 2T\left(\frac{n}{4}\right)\right) \Rightarrow 3c_1 + 4T\left(\frac{n}{4}\right)$

$$T\left(\frac{n}{4}\right) = c_1 + 2T\left(\frac{n}{8}\right)$$

$$T(n) = 3c_1 + 4\left(c_1 + 2T\left(\frac{n}{8}\right)\right) \Rightarrow 7c_1 + 8T\left(\frac{n}{8}\right)$$

$$\vdots$$

$$T(n) = k \cdot c_1 + 2^k\left(\frac{n}{2^k}\right)$$

$$\frac{n}{2^k} = 1 \; ; \; k = \log_2 n$$

$$T(n) = \log_2 n \cdot c_1 + 2^{\log_2 n} T(1) \; ; \; T(n) = \Theta(n)$$

$n \cdot 2$

$$T(n) = \begin{cases} 1 & \text{se } n \leq 1 \\ 4T\left(\frac{n}{2}\right) + n & \text{se } n > 1 \end{cases}$$

$n^{\log_2 4} = n^2$ CASO I TEOREMA MASTER

$n^{\log_2 4 - \varepsilon} \Rightarrow n' = n, \; \varepsilon = 2$

$T(n) = \Theta(n^2)$

METODO ITERATIVO

$$T(n) = n + 4T\left(\frac{n}{2}\right)$$

$$T\left(\frac{n}{2}\right) = \frac{n}{2} + 4T\left(\frac{n}{4}\right)$$

$$T(n) = n + 4\left(\frac{n}{2} + 4T\left(\frac{n}{4}\right)\right) \; ; \; n + 2n + 4^2 T\left(\frac{n}{4}\right)$$

$$T\left(\frac{n}{4}\right) = \frac{n}{4} + 4T\left(\frac{n}{8}\right)$$

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

$$\vdots$$

$k-1$

$$2^k + \left(\frac{n}{2^k}\right) \sum_{j=0} 2^j$$

$$\frac{n}{2^k} = 1 \; ; \; k = \log_2 n$$

$$4^{\log_2 n}(1) \cdot \frac{2^{\log_2 n} - 1}{2-1} \; ; \; n^2 + n - 1$$

ES. 1

```
algoritmo Alg1(int n) -> int
    if (n<=1) then
        return n;
    else
        int a = 1;
        int j = 1;
        while (j<n)
            a = a/2;
        end while
        return a+3 (Alg1(n/2))
```

`if (n<=1) then` `return n;`] $O(1)$ sta solo effettuando un controllo

`int a = 1;` `int j = 1;` inizializzazione dei variabili, quindi costo 1 ma all'interno dell'else troviamo un ciclo while che effettuerà un controllo per n volte all'incirca, quindi $O(n)$

`while (j<n)` `a = a/2;` `end while`] $O(n)$

$O(1)$

`return a+3 (Alg1(n/2))`] $T(n/2)$ ritorna 3 volte l'algoritmo sulla metà dei n più un intero a

$$\begin{cases} O(n) + T(n/2) & \text{se } n > 1 \\ O(1) & \text{se } n = 1 \end{cases}$$

$f(n) = n$

$b = 2$

$a = 1$

$n^{\log_2 1} = n^0 = 1$

CASO 3 TEOREMA MASTER

$n^{\log_2 1 + \varepsilon} = n^1 = n, \; \varepsilon = 1$ } rivedere

$\Theta(n)$ l'algoritmo ha un costo $\Theta(n)$ }

RIPROVO CON L' ITERAZIONE

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

$$T\left(\frac{n}{2}\right) = \frac{n}{2} + T\left(\frac{n}{4}\right)$$

$$T(n) = n + \left(\frac{n}{2} + T\left(\frac{n}{4}\right)\right)$$

$$T\left(\frac{n}{4}\right) = \frac{n}{4} + T\left(\frac{n}{8}\right)$$

$$T(n) = n + \frac{n}{2} + \left(\frac{n}{4} + T\left(\frac{n}{8}\right)\right)$$

$$T(n) = \underbrace{n + \frac{n}{2} + \frac{n}{4} + \dots}_{\approx 2n} T\left(\frac{n}{2^k}\right)$$

$$\frac{n}{2^k} = 1 \;;\; k = \log_2 n$$

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

$$T(n) = \Theta(n)$$

Es. 2

$$T(n) = 3T(2n) + c$$

18/11/2021

```
algoritmo mystery(array A[1, ..., n], int i, int n) → int
    if i ≥ (n-2) then → PASSO BASE
        if (n=1) or (A[i] > A[i+1]) then     ] O(1)
            return A[i]                                      ] O(1)
        else                                 ] O(1)
            return A[i+1]
        end if
    endif
    int h ← mystery(A, i+1, n)]  T(n-1)    (i+1: -1 dall'istanza)
```

```
if A[i] > h then
    return A[i]
else
    return h
endif
```

O(1) (if-then) + O(1) (else) = O(1)

$$T(n) = \begin{cases} c' & \text{se } n = 1 \\ c + T(n-1) & \text{se } n < 1 \end{cases}$$

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

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

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

$T(n-2) = T(n-2-1) + c$

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

$T(n) = T(n-k) + c \sum_{j=0}^{k-1} 2^j$

$n - k = 1, \quad k = n-1$

$T(n) = \underbrace{T(1)}_{c'} + c(2^n - 1)$

$T(n) = \Theta(n)$

//

ESAME 22/01/2021

```
algoritmo Alg1 (int n) → int
    if (n ≤ 1) then
        return n;
    else
        int a = 1
        int j = 1
        while (j ≤ n)
```

O(1) PASSO BASE

O(n) + O(1) = O(n)

```
        a = a/2
    endwhile
    return a + 3 * Alg1(n/2)        T(n/2)
endif
```

$$T(n) = \begin{cases} C' & \text{se } n = 1 \\ O(n) + T\left(\frac{n}{2}\right) & \text{se } n > 1 \end{cases}$$

METODO RICORRENZE

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

$$T(n/2) = O\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right)$$

$$T(n) = O(n) + \left(O\left(\frac{n}{2}\right) + T(n/4)\right)$$

$$T\left(\frac{n}{4}\right) = O(n) + T(n/8)$$

$$T(n) = O(n) + \left(O\left(\frac{n}{2^2}\right) + T\left(\frac{n}{8}\right)\right)$$

$$T(n) = O(n) + O\left(\frac{n}{2^k}\right) + T\left(\frac{n}{2^k}\right)$$

$\frac{n}{2^k} = 1$; $k = \log_2 n$

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

TEOREMA MASTER III CASO

$b = 2 \quad a = 1 \quad f(n) = n$

$n^{\log_2 1} = n^0 = 1 \qquad f(n) = n > n^{0+\epsilon}$ (2)

$a f\left(\frac{n}{b}\right) \leq c f(n) \qquad \frac{n}{2} \leq cn \; ; \; c = \frac{1}{2}$

$T(n) = \Theta(n)$

Esercizio 2

$T(n) = 3T(2n) + c$

$T(2n) = 3T(4n) + c$

$T(n) = 3\left(3T(4n) + 3^0 c\right) + c$

$T(4n) = 3T\left(2^4 n\right) + c$

$T(n) = 3\left(3^2 T\left(2^4 n\right) + 3^1 c\right) + c$

$T(n) = 3^k T\left(2^k n\right) + c \sum_{j=0}^{k-1} 3^j$

$2^k n = 1$

$\sum_{j=0}^{k-1} \frac{3^k - 1}{3 - 1}$ → 3 qualunque base sia

Generalizzando

$2^k n = 1 \; ; \; k = -\log_2 n$

$T(n) = 3^{-\log_2 n} T(1) + c\left(\frac{3^{-\log_2 n} - 1}{2}\right)$

$T(n) = n^{-\log_2 3} + c\left(\frac{n^{-\log_2 3} - 1}{2}\right)$

$T(n) = \Theta\left(n^{-\log_2 3}\right)$

// Esame 19/11/2020 PROVA B

```
algoritmo Alg1 (Array A, int n) → int
    if (n==1) then
        return n;                          ] PASSO BASE, O(1)
    else
        int a = 1
        int m = 1                          ] O(1)
        int k = a * m
        for j = 1 to k                     ]
            M = a + n/2                    ]
        endfor
        return Alg1(A, n/2) * Alg1(A, n/2) ] 2T(n/2)
    endif
```

$$\begin{cases} 2T(n/2) + \overbrace{O(1)}^{c} & \text{se } n > 1 \\ c' & \text{se } n = 1 \end{cases}$$

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

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + c$$

$$T(n) = 2\left(2c + 2^2 T\left(\frac{n}{4}\right)\right) + c$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + c$$

$$T(n) = 2\left(2^1 c + 2^1 T\left(\frac{n}{4}\right)\right) + c$$

Generalizzando

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + c \sum_{j=0}^{k-1} 2^j$$

$$\frac{n}{2^k} = 1 \; ; \; k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + c\left(2^{\log_2 n} - 1\right)$$

$$T(n) = \Theta(n)$$

Teorema Master

$$f(n) = O(1) \qquad a = 2, \; b = 2$$

$$f\left(n^{\log_2 2}\right) = n^1$$

Teorema master caso I

$$f\left(n^{\log_2 2 - \varepsilon}\right) = \Theta\left(n^{\log_2 2}\right) = \Theta(n)$$

Recap Master

CASO I

$$f(n^{\log_b a - \varepsilon}) = \Theta(n^{\log_b a})$$

CASO II

$$f(n^{\log_b a}) = \Theta(n^{\log_b a} \log_2 n)$$

CASO III

$$f(n^{\log_b a + \varepsilon}) \quad \varepsilon > 0 \quad c \in 1$$

Es. 2

$$T(n) = 4T(n/2) + n$$

$$T(n/2) = 4T\left(\frac{n}{4}\right) + \frac{n}{2}$$

$$T(n) = 4\left(4T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$T(n) = 4^2 T\left(\frac{n}{4}\right) + 2n + n$$

$$T\left(\frac{n}{4}\right) = 4T\left(\frac{n}{8}\right) + \frac{n}{4}$$

$$T(n) = 4^2\left(4T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n + n$$

$$T(n) = 4^3 T\left(\frac{n}{8}\right) + 4n + 2n + n$$

Generalizzo

$$T(n) = 4^k T\left(\frac{n}{2^k}\right) + n \sum_{j=0}^{k-1} 2^j$$

$$\frac{n}{2^k} = 1 \;, \quad k = \log_2 n$$

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

$T(n) = O(n^2)T(1) + O(n^2)$

$T(n) = O(n^2)$ ✓

19/11 PROVA A

```
algoritmo Alg1 (array A, int n) → int
    if (n == 1) then          } passo base
        return n              } O(1)
    else
        int b = 1             }
        int k = b * n         } O(1)
        for j = 1 to k        }
            a = b + n/2       } O(n)
        endfor
        return Alg1 (A, k/2) + Alg1 (A, n/2)   ] 2T(n/2)
    endif
```

ITERAZIONE

$$\begin{cases} C' & \text{se } n = 1 \\ O(n) + 2T\left(\frac{n}{2}\right) & \text{se } n > 1 \end{cases}$$

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

$T\left(\frac{n}{2}\right) = O\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right)$

$T(n) = O(n) + 2\left(O\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right)\right)$

$T\left(\frac{n}{4}\right) = O\left(\frac{n}{4}\right) + 2T\left(\frac{n}{8}\right)$

$T(n) = O(n) + 2\left(O\left(\frac{n}{2}\right) + 2\left(O\left(\frac{n}{4}\right) + 2T\left(\frac{n}{8}\right)\right)\right)$

Generalizzando

$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$

$$\frac{n}{2^k} = 1 \; ; \; k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + \log_2 n \cdot n$$

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

$$T(n) = \Theta(n \log_2 n)$$

Teorema Master

$$f(n) = n^{\log_2 2} \Rightarrow n$$

Teorema master CASO II

$$\Theta\left(n^{\log_2 2} \cdot \log_2 n\right) = \Theta(n \log_2 n)$$

PROVA A

```
function Function(A, i, j)
    while i ≤ j do
        newBeginIdx ← j
        newEndIdx ← i
        for ii = i to j do
            if A(ii) > A(ii+1) then
                SWAP(A(ii), A(ii+1))
                newEndIdx ← ii
            endif
        endfor
        j ← newEndIdx - 1
        for ii ← j downto i do
            if A(ii) > A(ii+1) then
                SWAP(A(ii), A(ii+1))
                newBeginIdx ← ii
            endif
        endfor
        i ← newBeginIdx ← ii
    endwhile
    return A
endfunction
```

O(1) O(n) O(1) O(n) O(1) O(n)

$O(n^2)$

PROVA A

```
algoritmo mystery(array A[1,...,n], int i, int n) → int
  if i ≥ n then
      return A[i]
  endif
  int m ← (i+n)/2
  int v1 ← mystery(A, i, m)
  int v2 ← mystery(A, m+1, n)
  if v1 ≥ v2 then
      return v1
  else
      return v2
  endif
```

Passo base $O(1)$

→ divido l'istanza

$2T\left(\frac{n}{2}\right)$ dove chiamate ricorsive su metà istanza

$O(1)$

$$T(n) = \begin{cases} c' & \text{se } n = 1 \\ 2T\left(\frac{n}{2}\right) + c & \text{se } n > 1 \end{cases}$$

Teorema Master

$b = 2 \quad a = 2 \quad f(n) = c$

$$f\left(n^{\log_b a}\right) = f\left(n^{\log_2 2}\right) = f(n^1) = f(n)$$

CASO I TEOREMA MASTER

$$f\left(n^{\log_2 2 - \varepsilon}\right)$$

$$\Theta\left(n^{\log_2 2}\right) ; \quad \Theta(n)$$

METODO DELL'ITERAZIONE

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

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + c$$

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + c\right) + c$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right)$$

$$T(n) = 2\left(2\left(2T\left(\frac{n}{8}\right) + c\right) + c\right) + c$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + c\sum_{i=0}^{k-1} 2^i$$

$$\frac{n}{2^k} = 1 \; ; \; k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + c\left(2^{k-1} - 1\right)$$

$$T(n) = n \cdot \underbrace{T(1)}_{c'} + c \cdot 2^{\log_2 n} - 1$$

$$T(n) = n \cdot c' + c(n-1)$$

$$T(n) = \Theta(n)$$

18/02/2019

```
algoritmo Alg1 (Array A, int n) → bool
    int a = 1
    int b = sizeof(A)                          } O(1)
    int c;
    while (A[(a+b)/2] != n) do                 } O(n)
        c = (a + b)/2
        if (A[c] > n) then                     } O(1)
            b = c - 1
        else
            a = c + 1
        endif
        if (a > b) then                        } O(1)
            return false;
        endif
    return true
```

$$T(n) = O(n)$$

//

$$2T(n/2) + O(n)$$

ITERAZIONE

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

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)$$

$$T(n) = O(n) + 2\left(2T\left(\frac{n}{4}\right) + O\left(\frac{n}{2}\right)\right)$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + O\left(\frac{n}{4}\right)$$

$$T(n) = O(n) + 2\left(2\left(2T\left(\left(\frac{n}{8}\right) + O\left(\frac{n}{4}\right)\right) + O\left(\frac{n}{2}\right)\right)\right)$$

Generalizzo

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\frac{n}{2^k} = 1 \quad ; \quad k = \log_2 n$$

$$T(n) = nT(1) + n\log_2 n$$

$$T(n) = \Theta(n \log_2 n)$$

TEOREMA MASTER

$$f(n) = \left(n^{\log_b a}\right) \qquad b = 2, \; a = 2$$

$$f(n) = (n^1) = n$$

CASO II

$$\Theta = \left(n^{\log_2 2} \log_2 n\right) \Rightarrow \Theta(n \log n)$$

//

$$2T\left(\frac{n}{2}\right) + c \qquad \text{se } n > 1$$

$$c' \qquad \text{se } n = 1$$

ITERAZIONE

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

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + c$$

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + c\right) + c$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + c$$

$$T(n) = 2\left(2\left(2T\left(\frac{n}{8} + c\right)\right) + c\right) + c$$

Generalizzato

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + c \sum_{i=0}^{k-1} 2^i$$

$$\frac{n}{2^k} = 1 \; ; \quad k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + c\left(2^{\log_2 n} - 1\right)$$

$$T(n) = n\,T(1) + c(n-1)$$

$$T(n) = O(n)$$

TEOREMA MASTER

$$f(n) = O(1)$$

$a = 2 \quad b = 2$

$$f\left(n^{\log_2 2}\right) = f(n) \quad (n^1)$$

CASO I TEOREMA MASTER

$$f\left(n^{\log_2 2 - \varepsilon}\right) = f\left(n^0\right) = 1$$

$\Theta(n)$

$O(n^2)$