0.1 Complessità asintotica

```
procedure Esempio_1 (n : integer);
int k:=0;
 while (k ≤ n) {
    for (int j = 1; j ≤ 3^k; j++) {
        t++;k++;}
                }
for (int r = 1; r ≤ k; r++)
   for (int s = 1; s ≤ k; s++)
       A[r,s]:=s;
 }
```

$T_1(n) = \sum_{k=0}^{n} \sum_{j=1}^{3^k} O(1) + O(k^2) = \sum_{k=0}^{n} O(3^k) + k^2 = c\frac{3^{n+1}-1}{3-1} + k^2 = O(3^n) + O(n^2) = O(3^n)$

```
procedure Esempio_2 (n : integer);
int k:=0;
 while (k ≤ n) {
    for (int j = 1; j ≤ 3^k; j++) {t++};
    for (int r = 1; r ≤ k; r++)
       for (int s = 1; s ≤ k; s++)
             A[r,s]:=s;
    k++;
 }
```

$T_2(n) = \sum_{k=0}^{n} \left(\sum_{j=1}^{3^k} O(1) + \sum_{r=1}^{k} \sum_{s=1}^{k} O(1) \right) = \sum_{k=0}^{n} \left(O(3^k) + k^2 \right) = O(3^n) + O(n^3) = O(3^n)$

```
procedure Esempio_3 (n : integer);
int k:=1; i:=1;
 while (k ≤ n) {
    for (int j = 1; j ≤ n; j++) {t:=4;}
    k := k * 2;
    }
 }
```

$T_3(n) = \sum_{t=1}^{\lfloor \log n \rfloor} n \leq \sum_{t=1}^{\log n} n = O(n \log n) = O(n^{1+\epsilon})$ per ogni $\epsilon > 0$. Si noti che il `while` esterno viene eseguito fino a che k assume il valore della potenza di 2 massima minore o uguale a n. Cioè, fino a che $k = 2^t \leq n$, ossia $t \leq \log_2(n)$, che equivale a $t \leq \lfloor \log_2(n) \rfloor$ perchè il numero di iterazioni t può assumere solo valori interi.

```
procedure Esempio_4 (n : integer);
int k:=1; i:=1;
 while (k ≤ n) {
    for (int j = 1; j ≤ n; j++) {
        t:=4;
        k := k * 2}
    }
 }
```

$T_4(n) = \sum_{t=0}^{0} O(n) = O(n)$

```
{procedure Esempio_5 (n : integer);}
int k:=1; i:=1;
 while (k ≤ n) {
```

```
    for (int j = 1; j ≤ k; j++) {
        w:=4}
    k:=k*9;
    }
}
```

$T_5(n) = \sum_{t=0}^{\lfloor \log_9 n \rfloor} 9^t \leq 9^{\log_9 n+1} \in O(n)$. Cosa sarebbe successo se k fosse stato inizializzato a 0?

```
{procedure Esempio_6 (n : integer);}
int k:=1; int i:=0;
 while (k ≤ n) {
    for (int j = 1; j ≤ 3^i; j++) {
        w:=4}
    k:=k*9; i++;
    }
}
```

Si noti che i conta quante volte è stato già eseguito il comando `while`.Pertanto $i = t$, con $i \geq 0$. $T_6(n) = \sum_{t=0}^{\lfloor \log_9 n \rfloor} O(3^i) = \sum_{t=0}^{\lfloor \log_9 n \rfloor} O(3^t) \leq c\frac{3^{\log_9 n+1}-1}{3-1}$ dove c rappresenta la costante nascosta in $O(3^t)$. Pertanto $T_6(n) \leq c'3^{\log_9 n} \leq c'3^{\frac{\log_3 n}{\log_3 9}} = cn^{\frac{1}{2}} \in O(\sqrt{(n)})$.

```
procedure Esempio_7 (n : integer);
int k:=1; i:=1;
 while (k ≤ n^2) {
    for (int j = 0; j ≤ log_2 k; j++) {
        w:=4}
    k:=k*2;
    }
}
```

Notando che $\log_2 k = t$, si ha $T_7(n) = \sum_{t=0}^{\lfloor 2\log n \rfloor} t + 1 \leq \log_2(n)(2\log_2 n + 1) \in O(\log^2 n)$

```
{procedure Esempio_8 }(n : integer);
int k:=1; i:=1;
 while (k ≤ n^2) {
    for (int j = 0; j ≤ log_2 k; j++) {w:=4}
    k:=k*4;
    }
}
```

Si noti che il ciclo `while` esterno è ripetuto fino a che $4^t \leq n^2$, ossia $t \leq 2\log_4 n = 2\frac{\log_2 n}{\log_2 4} = \log_2 n$. Pertanto: $T_8(n) = \sum_{t=0}^{\lfloor \log_2 n \rfloor} 2t + 1 \leq \in O(\log^2 n)$. Poichè $T_8(n) \geq \sum_{t=0}^{\lfloor \log_2 n \rfloor} t \in \Omega(\log^2 n)$, vale $T_8(n) \in \Theta(\log^2 n)$.

```
procedure Esempio_9 (n : integer);
 i=1;
 for (int k = 1; k ≤ n - 1; k++) {
    for (int j = ⌈n/k⌉; j ≤ ⌈n/(k+1)⌉ + 2; j--) {
            A[k,j]=i;
        i=i+2;
    }
}
```

Ricordando che $x \leq \lceil x \rceil < x+1$, si ha: $T_9(n) = O(1) + \sum_{k=1}^{n-1} \sum_{j=\lceil \frac{n}{k+1} \rceil + 2}^{\lceil \frac{n}{k} \rceil} O(1) < \sum_{k=1}^{n-1} \left(\frac{n}{k} + 1 - (\frac{n}{k+1} + 2) + 1 \right) = \sum_{k=1}^{n-1} \left(\frac{n}{k} - \frac{n}{k+1} \right) = n \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) = n(1 - \frac{1}{n}) = O(n)$. La sommatoria $\sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) = \sum_{k=1}^{n-1} \frac{1}{k(k+1)}$ è una sommatoria telescopica il cui valore è $1 - \frac{1}{n}$.

```
{procedure Esempio_9B }(n : integer);
 i=1;
 for (int k=1; k≤n-1; k++) {
    for (int j=⌊n/k⌋-2; j≤⌊n/(k+1)⌋; j--) {
            A[k,j]=i;
        i=i+2;
    }
 }
```

Ricordando che $x - 1 < \lfloor x \rfloor \leq x$, si ha: $T_{9B}(n) = O(1) + \sum_{k=1}^{n-1} \sum_{j=\lfloor \frac{n}{k+1} \rfloor}^{\lfloor \frac{n}{k} \rfloor - 2} O(1) < \sum_{k=1}^{n-1} \left(\frac{n}{k} - 2 - (\frac{n}{k+1} - 1) + 1 \right) = \sum_{k=1}^{n-1} \left(\frac{n}{k} - \frac{n}{k+1} \right) = n \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) = n(1 - \frac{1}{n}) = O(n)$

```
{procedure Esempio_10 }(n : integer);
int k:=n; i:=1;
 while (k≥3) {
    for (int j=1; j≤n; j++) {
          for (int i=1; i≤n/(2*j-1); i++) {
        w:=4}
    k := k^(1/3);
    }
 }
```

Innanzitutto, alla fine della t-esima iterata, con t che parte da 0, k assume il valore $n^{\frac{1}{3^t}}$. Il ciclo `while` esterno è eseguito fino a che $n^{\frac{1}{3^t}} \geq 3$, equivalentemente $3^{\frac{\log_3 n}{3^t}} \geq 3$, ossia $\frac{\log_3 n}{3^t} \geq 1$, o $t \leq \log_3 \log_3 n$. Il massimo valore di t che soddisfa $t \leq \log_3 \log_3 n$ è $t = \lfloor \log_3 \log_3 n \rfloor$.

Ricordando che $\sum_{j=1}^{n} \frac{1}{j} = O(\log n)$ (*serie armonica*), il costo del corpo del `while` è $\sum_{j=1}^{n} \sum_{i=1}^{\frac{n}{2*j-1}} O(1) = c \sum_{j=1}^{n} \frac{n}{2*j-1} \leq c \sum_{j=1}^{n} \frac{n}{j} = nc \sum_{j=1}^{n} \frac{1}{j} \in O(n \log n)$. Poichè il costo del corpo del while è indipendente da k, il costo complessivo della procedura è $T_{10}(n) = O(n \log n \log \log n)$..

```
{procedure Esempio_11} (n : integer);
int k:=1; a:=0;
    for (int j=1; j≤n; j++) {
          for (int i=1; i≤2*j-1; i++) {
        A[j,i]:= a;
        a++;}
    }
```

$T_{11}(n) = \sum_{j=1}^{n} \sum_{i=1}^{2j-1} O(1) = \sum_{j=1}^{n} O(2j-1) = \Theta(n(n+1)) - n = \Theta(n^2)$.

```
{procedure Esempio_12} (n : integer);
int k:=n; a:=0;
    while (k≥1) {
          for (int i=1; i≤k; i++) {
          a:=a+1;}
```

```
    k := k/3;
    }
```

E' facile vedere che si eseguono $O(n + n/3 + n/9 + ...1$ passi, e quindi $T_{12}(n) \in O(n^2)$. Possiamo dare per una valutazione piú precisa della complessità ricordando che la serie geometrica di ragione $x < 1$ $\sum_{k=0}^{+\infty} x^k$ converge a $\frac{1}{1-x}$ Infatti, $T_{12}(n) = \sum_{t=0}^{\log_3 n} \sum_{i=1}^{3^{\log_3 n - t}} O(1) = c\sum_{t=0}^{\log_3 n} 3^{\log_3 n - t} = cn\sum_{t=0}^{\log_3 n} 3^{-t} = cn\sum_{t=0}^{\log_3 n}(\frac{1}{3})^t = cn\frac{1}{1-\frac{1}{3}} = O(n)$.

```
{procedure Esempio_13} (n : integer);
    for (int k=1; k≤n; i++){
        for (int i=1; i≤k*n/2^k; i++) {a:=a+1;}
    }
```

$T_{13}(n) = \sum_{k=1}^{n} \sum_{i=1}^{k*\frac{n}{2^k}} O(1) = \sum_{k=1}^{n} k * \frac{n}{2^k} = n\sum_{k=1}^{n} k(\frac{1}{2})^k = n\frac{\frac{1}{2}}{\frac{1}{4}} \in O(n)$ poichè $\sum_{k=1}^{n} kx^k = \frac{x}{(1-x)^2}$ quando $x < 1$.

Si noti inoltre che quando $x > 1$, $\sum_{k=1}^{n} kx^k \in O(nx^n)$.

```
procedure Esempio_14 (n : integer);
int k:=2^n; a:=0;
    while (k>1) {
        for (int i=1; i≤k; i++) {
        a:=i;}
    k:=k/2;
    }
```

$T_{13}(n) = \sum_{t=0}^{n-1} \sum_{i=1}^{2^{n-t}} O(1) = \sum_{t=0}^{n-1} 2^{n-t} = \sum_{t=1}^{n} 2^t = O(2^n)$

```
procedure Esempio_15 (n : integer);
int k:=0; a:=0;
    while (k<n) {
        for (int i=n; i>k; i--) {
        a:=i;}
    k:=k+5;
    }
```

$T_{15}(n) = \sum_{t=0}^{\lfloor n/5 \rfloor} \sum_{i=1}^{n-5*t} O(1) = \sum_{t=0}^{\lfloor n/5 \rfloor}(n - 5*t) = \Theta(n^2) - 5\sum_{t=0}^{\lfloor n/5 \rfloor} t = \Theta(n^2) - 5\frac{(n/5)(n/5+1)}{2} \in \Theta(n^2)$.

```
procedure Esempio_16 (n : integer);
int k:=0;
    for (int i:=1; i≤n; i++) {k:=k+2};
    for (int i=1; i≤2^k; i++) {
        a:=i;}
    k:=k+5;
    }
```

$T_{16}(n) = \sum_{i=1}^{n} c + \sum_{i=1}^{2^{2n}} c' = cn + c'2^{2n}) = \Theta(2^{2n})$. Si verifichi che $\Theta(2^{2n}) \in \Omega(2^n)$, mentre $\Theta(2^{2n}) \notin \emptyset(2^n)$.

```
{procedure Esempio_17 }(n : integer);
int k:=0;
    for (int i:=1; i≤n; i++) {k:=2+k};
    while (k≥1) {
```

```
        j:=n;
        while (j ≥ 3) j:=j −5;}
    k := ⌊log2(k)⌋;
    }
```

$T_{17}(n) = \sum_{i=1}^{n} c + \sum_{t=1}^{\log^*(2n)} cn = O(n) + O(n \log^* n) \sim O(n)$.

```
{procedure Esempio_18} (n : integer);
int k:=0;
    for (int i:=1; i ≤ n; i++) {k:=i∗k};
    while (k \ge 1) {
        j:=n;
        while (j ≤ n) { j:=j −5};
            k := log2(k);
    }
```

$T_{18}(n) = \sum_{i=1}^{n} c + \sum_{t=1}^{\log^*(n!)} cn = O(n) + O(n \log^* n)$ perchè $\log^*(n!) = \Theta(n \log n) \leq \log^* n + 2$