"*truc*" est une procedure dont le "temps d'exécution" $T(truc, n)$ dépend d'un entier n.
Prouver l'existence de (a, b) tel que $\forall n,\ a \ln n \leq T(truc, n) \leq b \ln n$ (temps logarithmique) (ou bien $\forall n,\ a\, n \leq T(truc, n) \leq b\, n$ (temps linéaire) ou bien $\forall n,\ an^2 \leq T(truc, n) \leq bn^2$ (temps quadratique),...)
Exemples :

1. Pour une procedure non récursive : cf "Calculer un booléen", exemple 2.

```
chose :=proc(i)
        global n ;
local k,iItere ;
(1)  iItere :=i ;
(2)  for k from 0 to n-1 do
(3)          if iItere>15 then
(4)                  return "vrai"
(5)          fi ;
(6)          iItere :=t[iItere]
(7)  od ;
(8)  return "faux" ;
     end ;
```

$$T(chose, n) = \left| \begin{array}{ll} 2 & \text{ligne (1) :1 accès,1 écriture} \\ +n\times & \text{ligne (2) : nombre de "tours de boucle"} \\ (2+2) & \text{ligne (3) :1 accès, 1 evaluation-boolean, ligne (6) :1 accès, 1 écriture} \end{array} \right. = 2n{+}2 \in [2n, 3n]$$

si $n \geq N_0$: complexité linéaire.

```
truc :=proc(t)
global n ;
local i ;
(1)  for i from 0 to n-1 do
(2)          if chose(i)="faux" then
(3)                  return "faux"
(4)          fi ;
(5)  od ;
(6)  return "vrai" ;
     end ;
```

$$T(truc, n) = \left| \begin{array}{ll} n\times & \text{ligne (1) : nombre de "tours de boucle"} \\ (1 + T(chose, n) + 1) & \text{ligne (2) :1 appel, 1 evaluation-boolean} \end{array} \right. = n(2{+}(2n{+}2)) = 2n^2{+}4n \in [2n^2, 3n^2]$$

si $n \geq N_0$: complexité quadratique.

2. Pour une procedure récursive :
   (a) Algorithme de Hörner :

```
     truc :=proc(a,x,i)
(1)  if i=n-1 then
(2)          return a[i]
(3)  else
(4)          return a[i]+truc(a,x,i+1)*x
(5)  fi ;
     end ;
```

   n est fixé et i varie de 0 à $n-1$.
   $T(truc, n-1) = 1$ (ligne(2))
   Si $i \leq n-2$, alors $T(truc, i) = 1 + T(truc, i+1) + 3$ (1 accès, 3 calculs) $= 4 + T(truc, i+1)$.
   $T(truc, 0) = T(truc, n-1) + \sum_{i=0}^{n-2} 4 = 4n - 4 \in [\![\, 3n, 4n \,]\!]$ pour $n \geq N_0$: complexité linéaire.

   (b) Exponentiation rapide :

```
     truc :=proc(n,a)
     local reste,quotient ;
(1)  if n=0 then
(2)          return 1
(3)  else
(4)          quotient :=iquo(n,2) ;
(5)          reste :=irem(n,2) ;
```

```
(6)          if reste=0 then
(7)                  return truc(quotient,a*a)
(8)          else
(9)                  return a*truc(quotient,a*a)
(10)         fi
(11)   fi;
       end;
```

$T(truc, 0) = 0$ (ligne(2))

Si $n \geq 1$, alors $T(truc, n) = \left| \begin{array}{ll} 2 & \text{ligne (4)} \\ +2 & \text{ligne (5)} \\ +T(truc, n/2) & \text{ligne (7)} \\ +2 & \text{ligne (7) :1 accès, 1 calcul} \end{array} \right. = 6 + T(truc, n/2)$ si n est pair, $T(truc, n) = \left| \begin{array}{ll} 2 & \text{ligne (4)} \\ +2 & \text{ligne (5)} \\ +T(truc, (n-1)/2) & \text{ligne (9)} \\ +3 & \text{ligne (7) :1 accès, 2 calcul} \end{array} \right. = 7 + T(truc, (n-1)/2)$

si n est impair.

Supposons que $n = \sum_{k=0}^{p} a_k 2^k$ avec $a_p \neq 0$ et $\forall k$, $a_p \in \{0, 1\}$ (ie $n = a_p ... a_1\, a_0$ en binaire et $a_p = 1$) ;

iquo(n,2)= $a_p ... a_1$ en binaire ou $\sum_{k=0}^{p-1} a_{k+1} 2^k$, ie on a "perdu" un chiffre en binaire.

Soit $m(p), M(p)$ tels que $m(p) \leq T(truc, n) \leq M(p)$ pour tout n ayant $(p+1)$ chiffres en binaires : on a $6 + m(p-1) \leq m(p) \leq M(p) \leq 7 + M(p-1)$ et $m(0) = M(0) = 0$ donc $m(p) = 6p$, $M(p) = 7p$ est une solution.

$2^p \leq n \leq \sum_{k=0^p} 2^k = 2^{p+1} - 1$ donc $\dfrac{\ln(n+1)}{\ln 2} - 1 \leq p \leq \dfrac{\ln n}{\ln 2}$ et

$$5\frac{\ln n}{\ln 2} \leq 6\left(\frac{\ln(n+1)}{\ln 2} - 1\right) \leq T(truc, n) \leq 7\frac{\ln n}{\ln 2} \text{ (si } n \geq N_0)$$

donc la complexité est logarithmique.

(c) Tri-fusion :

```
       tri :=proc(t,p)
       local tScinde,t1,t2,t1Trie,t2Trie,
(1)    if p=0 then
(2)          return t
(3)    else
(4)          tScinde :=scinde(t,p) ;
(5)          t1 :=tScinde[1] ;
(6)          t2 :=tScinde[2] ;
(7)          t1Trie :=tri(t1,p-1) ;
(8)          t2Trie :=tri(t2,p-1) ;
(9)          return fusion(t1Trie,t2Trie,p)
(10)   fi;
       end;
```

La taille de la liste triée est $n = 2^p$.

$T(tri, 1) = 0$ (ligne(2))

Si $p \geq 1$, alors $T(tri, 2^p) = \left| \begin{array}{ll} T(scinde, p) & \text{ligne (4)} \\ +2 \times 2^{p-1} & \text{ligne (5) et (6) :copies des 2 listes} \\ +2 \times T(tri, 2^{p-1}) & \text{ligne (7) et (8)} \\ +T(fusion, p) & \text{ligne (9)} \end{array} \right.$.

$T(scinde, p) = 2 \times 2^p$ et $T(fusion, p) \in [a.2^p, b.2^p]$ donc $(3+a)2^p + 2.T(tri, 2^{p-1} \leq T(tri, 2^p) \leq (3+b)2^p + 2.T(tri, 2^{p-1}$ et

$$\frac{3+a}{\ln 2} n \ln n \leq T(tri, n) \leq \frac{3+b}{\ln 2} n \ln n$$

(complexité en $n \ln n$)