Analyse du calcul récursif de F_n.

Soit f_n une majoration du coût du calcul d'un F_i pour $i \leqslant n$ et m_n une majoration de celui nécessaire pour multiplier deux entiers inférieurs à $F_{\lfloor n \rfloor}$. Les équations

$$F_{2n} = F_{n-1}^2 + F_n^2 \quad \text{et} \quad F_{2n-1} = F_{n-1}(2F_n - F_{n-1})$$

impliquent, que

$$f_n \ll 2f_{n/2} + 2m_{n/2} \ll 2^k f_{n/2^k} + 2\sum_{i=1}^{k} 2^i m_{n/2^i},$$

pour tout $k \geqslant 1$. Comme $F_n \sim \frac{1}{\sqrt{5}}\phi^n$, il a $O(n)$ chiffres, et la multiplication naïve donne $m_n = O(n^2)$. En choisissant $k \sim \log_2(n)$, on obtient $f_n = O(n^2)$.

Plus généralement, sous l'hypothèse $m_{n/a} = O(m_n/a)$ pour $a \geqslant 1$, on obtient $f_n = O(m_n)$ [*il faut voir que* $n = O(m_n)$].

Une variante : si on pose $T_n := [F_{n-1}, F_n]$, alors les relations de récurrences ci-dessus montrent que T_n se calcule facilement en fonction de $T_{\lfloor n/2 \rfloor}$:

$$\left\{\begin{array}{rcl} F_{2n} & = & F_{n-1}^2 + F_n^2 \\ F_{2n-1} & = & F_{n-1}(2F_n - F_{n-1}) \end{array}\right. \quad \text{et} \quad \left\{\begin{array}{rcl} F_{2n+1} & = & F_n(2F_{n-1} + F_n) \\ F_{2n} & = & F_{n-1}^2 + F_n^2 \end{array}\right.$$

Ce qui donne le programme suivant, ne contenant plus qu'un seul appel récursif :

```
T := proc(n)
  q := iquo(n, 2, 'r') ;
  t := T(q) ; a := t[1] ; b := t[2] ;
  if (r = 1) then [a^2 + b^2, b*(2*a + b)] ;
             else [a*(2*b - a), a^2 + b^2] ;
  fi ;
end ;
T(0) := [0,1] ;
T(1) := [1,1] ;
```

Si t_n majore le coût du calcul de $T_{\lfloor n \rfloor}$, on a $t_n \leqslant 3m_{n/2} + t_{n/2}$ (la multiplication par 2 est une addition).

Gros avantage : l'option `remember` n'est plus nécessaire, puisque aucun résultat n'est calculé plusieurs fois. Le coût en mémoire est bien moindre.