Corrigé du TD du 2 février 2006

1 Exponentiation rapide

Algorithme séquentiel de base

```
mult1:=proc(x,n)
   local i,res;
      res:=1;
      for i to n do
         res:=res*x;
      end do;
end proc;
```

Ici on fait n multiplications.

Algorithme récursif de base

```
mult2:=proc(x,n)
   if n=0 then 1
      else x*mult2(x,n-1)
   end if;
end proc;
```

Ici, on fait aussi n multiplications.

Algorithme rapide

```
mult3:=proc(x,n);
   if n=1 then x
      elif n mod 2=0 then mult3(x^2,n/2)
      else x*mult3(x^2,(n-1)/2)
   end if;
end proc;
```

Ici on a en gros $\log_2(n)$ passages dans l'algorithme et à chaque fois, on fait 1 ou 2 multiplications. La complexité est donc en $O(\log_{10}(n))$. Par exemple, calculer $x^{1000,000,000}$ demande seulement 41 multiplications à l'algorithme. Essayez `mult3(2,1000000)` puis `mult1(2,1000000)`.

2 Multiplication rapide

On décompose le programme en deux parties pour avoir une meilleure présentation. `Karatsuba` calcule `l`, la plus petite puissance de 2 supérieure à la longueur du plus grand multiplicande et appelle la procédure récursive `Kara` en utilisant cette information supplémentaire. L'idée de considérer `n1` et `n2` comme des nombres de `l` chiffres (en rajoutant des 0 virtuels à leur gauche) que l'on peut donc toujours couper en deux...

```
   Karatsuba:=proc(n1,n2)
   local l;
      l:=2^ceil(ln(max(length(n1),length(n2)))/ln(2));
      Kara(n1,n2,l);
end proc;

Kara:=proc(n1,n2,l)
   local a,b,c,d,x,y,z;
      if l=0 then 0
      elif l=1 then n1*n2;
      else b:=n1 mod 10^(l/2); a:=(n1-b)/10^(l/2);
           d:=n2 mod 10^(l/2); c:=(n2-d)/10^(l/2);
           x:=Kara(a,c,l/2); y:=Kara(b,d,l/2); z:=Kara(a-b,c-d,l/2);
           x*10^l+(x+y-z)*10^(l/2)+y;
      end if;
end proc;
```