Diviser pour régner

Principes généraux

Pour résoudre un problème de taille N, un **algorithme récursif** fonctionne en général de la manière suivante :

- Extraire ou construire à partir de notre entrée un problème de taille N-1
- Résoudre le problème de taille N-1 (récursivité)
- Utiliser cette solution pour résoudre le problème initial

La méthode ***diviser pour régner*** (*Divide and conquer ou Divide and rule* en anglais) consiste, pour résoudre un problème de taille N, à :

- (**Diviser**) : partager le problème en sous-problèmes (par exemple de taille N/2)
- (**Régner**) : résoudre ces différents sous-problèmes (généralement récursivement)
- (**Combiner**) : fusionner les solutions pour obtenir la solution du problème initial

---

On obtient en général moins d'appels récursifs. Dans certains cas la méthode diviser pour régner donne un algorithme de résolution plus rapide.

Un exemple : Exponentiation rapide

Approche itérative :

x^n est le `resultat`, initialement égal à 1, de l'opération `resultat*x` qui est répétée `n` fois.

Approche récursive :

On peut définir x^n de façon récursive :

- $x^0 = 1$
- $x^n = x * x^{n-1}$

La complexité (en regardant le nombre de multiplication) de cet algorithme est en O(n), il faudra donc faire 100 multiplications pour calculer x^{100} .

Écrire la fonction `puissance_recursive(x,n)`.

À partir de quelle valeur de *n* on dépasse les capacités de la pile ?

Une autre approche : Diviser pour régner

On peut définir x^n d'une autre façon :

- $x^0 = 1$
- Si n est pair : $x^n = (x^{n/2})^2$
- Si n est impair : $x^n = x*(x^{(n-1)/2})^2$

La complexité (en regardant le nombre de multiplication) de cet algorithme est en $O(\log_2(n))$, il faudra donc faire 7 multiplications pour calculer x^{128} .

Écrire la fonction `puissance_dpr(x,n)`.

Vérifier que la différence des temps d'exécutions des deux programmes est évidente pour par exemple le calcul de 2^{500}

Vérifier également que le second programme permet de calculer 2^{1000}