Méthode *diviser pour régner*

Dernière mise à jour le : 27/10/2023

En anglais, on dit *divide and conquer*.

Définition

On appelle **diviser pour régner** une méthode algorithmique de résolution d'un problème consistant à :

1. DIVISER : découper le problème initial en sous-problèmes ;
2. REGNER : résoudre les sous-problèmes (récursivement ou directement s'ils sont assez petits) ;
3. COMBINER : calculer une solution au problème initial à partir des solutions des sous-problèmes.

Exemples

Premier exemple

On dispose d'une fonction efficace pour traduire une phrase et on veut traduire tout un texte : on décompose (phase "Diviser") alors le texte en phrases (terminées par certains signes de ponctuation) que l'on traduit avec la fonction connue (phase "Régner"), puis on juxtapose les phrases traduites (phase "Combiner").

La recherche dichotomique

La recherche dichotomique dans un tableau trié est un autre exemple d'application de la méthode *diviser pour régner*. En effet, l'idée de la recherche dichotomique est de comparer la valeur `v` cherchée à l'élément central et, selon le cas, on a trouvé `v` ou on poursuit la recherche dans la moitié de gauche ou de droite. On réduit ainsi le problème initial (recherche dans le tableau tout entier) à un problème plus simple (recherche dans une portion du tableau dont la taille est divisée par deux), jusqu'à trouver un cas simple (en trouvant la valeur `v` ou en arrivant à la dernière valeur du tableau sans la trouver).

En classe de Première, l'algorithme de recherche dichotomique a été écrit avec une boucle `while` mais il s'écrit également de manière récursive assez naturellement.

```
# Recherche récursive de v dans le tableau T[g..d]
def recherche(T, v, g, d):
    """Renvoie une position de v dans T[g..d], ou None si v ne s'y trouve pas"""
    if g > d:
        return None
    m = (g + d) // 2
    if T[m] > v:
        return recherche(T, v, g, m - 1)
    elif T[m] < v :
        return recherche(T, v, m + 1, d)
    else:
        return m

# Recherche dichotomique en commençant la recherche sur le tableau entier
def recherche_dichotomique(T, v):
    """Renvoie une position de v dans le tableau T, ou None si v ne s'y trouve pas"""
    return recherche(T, v, 0, len(T) - 1)
```

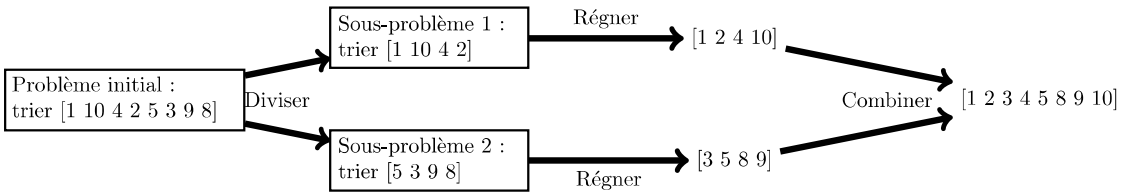
On peut vérifier que tout fonctionne :

```
>>> tab = [1, 1, 2, 2, 3, 4, 4, 6, 7, 8, 8, 9, 10]
>>> recherche_dichotomique(tab, 3)
4
>>> recherche_dichotomique(tab, 5)  # renvoie None
```

Autres exemples

Les algorithmes mettant en jeu la méthode *diviser pour régner* sont assez nombreux, certains seront abordés dans les activités. Citons par exemple :

- Recherche de la première occurrence d'un élément dans un tableau ;
- Le *tri fusion*, plus efficace que ceux abordés en classe de Première (tri par sélection, tri par insertion) ;



Les trois étapes illustrées avec l'algorithme du tri fusion

Crédits : Fschwarzentruber, CC BY-SA 4.0, via Wikimedia Commons

- Le *tri rapide* (hors programme) ;
- Multiplication et exponentiation rapide ;
- Rotation d'une image d'un quart de tour ;
- etc.

Bilan

- La méthode *diviser pour régner* consiste à décomposer un problème en plusieurs **sous-problèmes** de même nature mais plus petits (étape DIVISER), à résoudre ensuite chacun des sous-problèmes (étape REGNER) et enfin à combiner les résultats des sous-problèmes pour obtenir le résultat du problème initial (étape COMBINER).
- La résolution des sous-problèmes se fait en général récursivement en les décomposant à leur tour en problèmes plus petits encore jusqu'à arriver au(x) cas de base.
- Cette méthode permet d'écrire des algorithmes souvent plus efficaces mais elle reste parfois limitée par le nombre d'appels récursifs

---

Références :

- Equipe pédagogique DIU EIL, Université de Nantes.

Germain BECKER, Lycée Mounier, ANGERS

Voir en ligne : info-mounier.fr/terminale_nsi/algorithmique/diviser-pour-regner