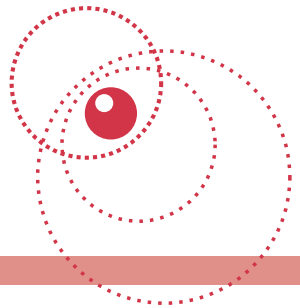

Entraînement au concours ACM-ICPC

Récursion, mémoïsation, programmation dynamique

4 mai 2015

Pierre Senellart

TELECOM ParisTech

Plan

Quand utiliser la récursion ?

Un problème P peut être résolu par récursion quand :

- Il peut être paramétré par un ou plusieurs entiers $P(n_1 \dots n_k)$
- La solution de $P(n_1 \dots n_k)$ peut être calculée à partir de la solution de $P(n_1^{(1)} \dots n_k^{(1)}) \dots P(n_1^{(\ell)} \dots n_k^{(\ell)})$ pour un certain $\ell \geqslant 1$ avec pour tout $1 \leqslant i \leqslant \ell$, chaque $n_j^{(i)} \leqslant n_j$ pour $1 \leqslant j \leqslant k$, l'une de ces inégalités étant stricte ($\exists 1 \leqslant j \leqslant k, n_j^{(i)} < n_j$) : cas récursif
- $P(0 \dots 0)$ (ou $P(n_1 \dots n_k)$ avec n_i « petit » suivant les cas) facile à calculer : cas de base

Exemple 1 : Factorielle

- $P(n) = n!$
- $k = 1, \ell = 1$
- $P(n) = n \times P(n-1)$ pour $n \geqslant 1$
- $P(0) = 1$

Exemple 2 : Fibonacci

- $P(n)$ n-ième nombre de Fibonacci
- $k = 1$, $\ell = 2$
- $P(n) = P(n-1) + P(n-2)$ pour $n \geqslant 1$
- $P(0) = 1$, $P(1) = 1$

Exemple 3 : distance d'édition de Levenshtein

$d(s, s')$ distance d'édition (nombre minimal de caractères à ajouter, enlever, modifier) pour passer de s à s'.

- $P(n_1, n_2)$ distance d'édition entre le préfixe de longueur n_1 de s et le préfixe de longueur n_2 de s'
- $k = 2$, $\ell = 3$
- $P(n_1, n_2) =$ $\min\left(P(n_1 - 1, n_2) + 1, P(n_1, n_2 - 1) + 1, P(n_1 - 1, n_2 - 1) + \mathbb{I}_{s_{n_1} \neq s'_{n_2}}\right)$ pour $n_1 \geqslant 1$, $n_2 \geqslant 1$
- $P(n_1, 0) = n_1$ pour $n_1 \geqslant 0$; $P(0, n_2) = n_2$ pour $n_2 \geqslant 0$.

$\mathbb{I}_b$ vaut 1 si b est vrai, 0 sinon

Implémentation

- Fonction récursive

function $P(n_1, \ldots, n_k)$;
if *cas de base* **then**
    ...;
    return ...;
else
    // cas récursif
    ...;
    // ℓ appels à P
    return ...;
end

- On dit que la récursion est terminale si $\ell = 1$ et si l'appel à P est la dernière instruction appelée du cas récursif (**return** $P(n'_1, \ldots, n'_k)$)
- NB : Dans l'example ci-dessus le **else** peut être omis.

Complexité

- Bornes supérieures (parfois meilleures, par exemple si l'appel récursif se fait sur $\frac{n}{2}$ et pas sur $n-1$) :
  - Si $\ell = 1$: $O(n_1 \times \cdots \times n_k)$: souvent acceptable
  - Si $\ell > 1$: $O(\ell^{n_1 \times \cdots \times n_k})$: souvent non raisonnable
- Attention à l'espace mémoire sur la pile : $O(\prod_{i=1}^{k} n_i)$ (en général l'espace allouée à la pile va de quelques centaines de kilo-octets à quelques méga-octets).
- En cas de récursion terminale, le compilateur (Java, C++) peut éliminer la récursion, et donc l'espace sur la pile requis

Plan

Principe (1/3)

Mémoire : objet global (ou statique, ou membre de classe) qui se souvient des résultats de la fonction P d'un appel à l'autre

Principe (2/3)

function $P(n_1, \ldots, n_k)$;
if $M(n_1, \ldots, n_k)$ *est défini* **then**
    return $M(n_1, \ldots, n_k)$;
end
if *cas de base* **then**
    ...;
    $r \leftarrow$...;
else
    // cas récursif
    ...;
    // ℓ appels à P
    $r \leftarrow$...;
end
$M(n_1, \ldots, n_k) \leftarrow r$;
return r ;

Principe (3/3)

- Récursion terminale impossible !
- Structure de données : tableau multi-dimensionnel ou tableau associatif (map) implémenté comme un arbre binaire de recherche (TreeMap en Java, map en C++) ou comme une table de hachage (HashMap en Java, unordered_map en C++ 2011/TR1)...
- Choisir un tableau classique quand les paramètres sont des entiers à valeurs contiguës, choisir un tableau associatif sinon
- Les tableaux peuvent généralement être alloués statiquement quand on dispose d'un majorant a priori de la taille des paramètres

Complexité

- $O(n_1 \times \cdots \times n_k \times \ell)$ opérations
- $O(n_1 \times \cdots \times n_k)$ espace mémoire sur le tas (espace occupé dépend de la structure de données utilisée)
- $O(n_1 \times \cdots \times n_k)$ espace mémoire sur la pile
- Pas de calculs inutiles !

Plan

Récursion

Mémoïsation

Programmation dynamique

Principe

- Calcul itératif
- On construit une matrice (tableau multi-dimensionnel) $n_1 \times \cdots \times n_k$ qu'on remplit itérativement du bas vers le haut

function $P(n_1, \ldots, n_k)$;
$M \leftarrow \text{matrix}(n_1, \ldots, n_k)$;
// remplir M avec les cas de base
for $i_1 \leftarrow 1$ **to** n_1 **do**
    ...;
    for $i_k \leftarrow 1$ **to** n_k **do**
        ...;
        $M(i_{,1}, \ldots, i_k) \leftarrow \ldots$;
        // utilise les valeurs calculées
    end
end
return $M(n_1 \ldots n_k)$;

Complexité

- $\blacksquare(n_1 \times \cdots \times n_k \times \ell)$ opérations
- $\blacksquare(n_1 \times \cdots \times n_k)$ espace mémoire sur le tas
- $O(1)$ espace mémoire sur la pile
- Parfois trop d'opérations !

- $g(n) = O(f(n))$: $g(n) \leqslant k \cdot f(n)$ pour un $k > 0$ et n suffisamment grand
- $g(n) = \blacksquare(f(n))$: $g(n) \geqslant k' \cdot f(n)$ pour un $k' > 0$ et n suffisamment grand
- $g(n) = \blacksquare(f(n))$: à la fois $g(n) = O(f(n))$ et $g(n) = \blacksquare(f(n))$

Mémoïsation vs Programmation dynamique

- Les deux techniques reviennent essentiellement au même
- Avantage de la mémoisation : seuls les calculs nécessaires sont effectués (on ne remplit pas une matrice complète)
- Inconvénient de la mémoïsation : les appels récursifs ont un léger surcoût (par rapport à un style impératif) et utilisent la pile d'appel (limitée en taille)
- On peut reproduire le comportement de la programmation dynamique à partir de la mémoïsation
- On peut dans certains cas complexes simuler les appels récursifs avec une pile maintenue à la main sur le tas

Licence de droits d'usage

Contexte public } avec modifications

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.
Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :
– le droit de reproduire tout ou partie du document sur support informatique ou papier,
– le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
– le droit de modifier la forme ou la présentation du document,
– le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
  – L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.
Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitepedago@telecom-paristech.fr