IFT2125 - Introduction à l'algorithmique

Programmation dynamique (B&B chapitre 8)

Pierre McKenzie

DIRO, Université de Montréal

Automne 2017

Deux usages inefficaces de diviser-pour-régner

1) Calcul des coefficients binomiaux, B&B Section 8.1.1

<u>COEFFICIENT BINOMIAL</u>

DONNÉE: entiers $0 \leq k \leq n$

CALCULER: $\binom{n}{k}$

Voici un algo diviser-pour-régner pour COEFFICIENT BINOMIAL :

function $C(n, k)$
  if $k = 0$ **or** $k = n$ **then return** 1
  else return $C(n-1, k-1) + C(n-1, k)$

Deux usages inefficaces de diviser-pour-régner

2) Probabilité de l'emporter en série mondiale, B&B Section 8.1.2

SÉRIE MONDIALE

DONNÉE: probabilité $0 \leq p \leq 1$ que A batte B lors d'un seul match, entiers $n > 0$ et $0 \leq i, j < i + j < 2n$

CALCULER: probabilité $P(i, j)$ que A gagne n matchs avant B, sachant qu'il manque à A i victoires et à B j victoires

Quel serait un algo diviser-pour-régner ?

Deux usages inefficaces de diviser-pour-régner

2) Probabilité de l'emporter en série mondiale, B&B Section 8.1.2

<u>SÉRIE MONDIALE</u>

DONNÉE: probabilité $0 \leq p \leq 1$ que A batte B lors d'un seul match, entiers $n > 0$ et $0 \leq i, j < i + j < 2n$

CALCULER: probabilité $P(i, j)$ que A gagne n matchs avant B, sachant qu'il manque à A i victoires et à B j victoires

Quel serait un algo diviser-pour-régner ?

function $P(i, j)$
  if $i = 0$ **then return** 1
  else if $j = 0$ **then return** 0
  else return $pP(i-1, j) + qP(i, j-1)$

Bien meilleure solution pour Série mondiale

Par "programmation dynamique"

$$\underbrace{P(i,j) = pP(i-1,j) + qP(i,j-1)}_{\text{suggère de remplir un tableau une diagonale à la fois de haut en bas}}$$

SÉRIE MONDIALE par programmation dynamique

B&B Section 8.1.2

function *series*(n, p)
    array $P[0..n, 0..n]$
    $q \leftarrow 1 - p$
    {Fill from top left to main diagonal}
    for $s \leftarrow 1$ **to** n **do**
        $P[0, s] \leftarrow 1$; $P[s, 0] \leftarrow 0$
        for $k \leftarrow 1$ **to** $s - 1$ **do**
            $P[k, s-k] \leftarrow pP[k-1, s-k] + qP[k, s-k-1]$
    {Fill from below main diagonal to bottom right}
    for $s \leftarrow 1$ **to** n **do**
        for $k \leftarrow 0$ **to** $n - s$ **do**
            $P[s+k, n-k] \leftarrow pP[s+k-1, n-k] + qP[s+k, n-k-1]$
    return $P[n, n]$

Rendre la monnaie

B&B Section 8.2

- Rappel : bien qu'efficace, l'approche vorace parfois ratait une solution

- L'approche programmation dynamique fonctionne, quelles que soient les valeurs des pièces :
  - démo du 15 novembre.
  - L'idée :
    $c[i, j]$ = nombre min de pièces pour rendre j en i dénominations
  - Alors :
    $$\underbrace{c[i, j] = \min(c[i-1, j], 1 + c[i, j - \text{dénom}[i]])}_{\text{suggère à nouveau de remplir par diagonales, de haut en bas}}$$

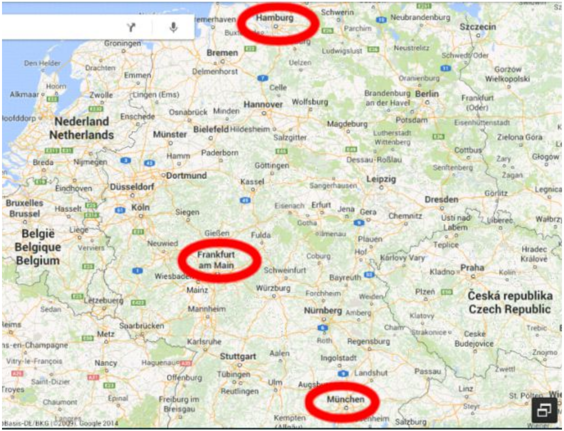
Principe d'optimalité

B&B Section 8.3

La programmation dynamique est à privilégier lorsque

- le problème à résoudre se décompose en sous-problèmes semblables
- ces sous-problèmes ont tendance à se chevaucher
- le principe d'optimalité s'applique : chaque sous-séquence d'une séquence de choix optimale est optimale

Principe d'optimalité
Exemples



- chemin le plus court : oui
- chemin le plus rapide : non
- chemin simple le plus long : non, ex : graphe complet

<u>SAC À DOS</u>

DONNÉE: capacité $W \in \mathbb{R}^{\geq 0}$ et objets $1, 2, \ldots, n$ de poids $w_1, \ldots, w_n \in \mathbb{R}^{\geq 0}$ et de valeurs $v_1, \ldots, v_n \in \mathbb{R}^{\geq 0}$

CALCULER: objets de valeur maximale et de poids n'excédant pas W

- Rappel : bien qu'efficace, l'approche vorace ne parvenait à résoudre que SAC À DOS FRACTIONNAIRE

- L'approche programmation dynamique résoud SAC À DOS
  Comment ?

Sac à dos
(suite)

- L'idée :
  $V[i,j] =$ valeur max avec objets $\{1, 2, \ldots, i\}$ et capacité $\leq j$
- On cherche : $V[n, W]$

- Alors :

Sac à dos
(suite)

- L'idée :
  $V[i,j] =$ valeur max avec objets $\{1,2,\ldots,i\}$ et capacité $\leq j$
- On cherche : $V[n,W]$

- Alors :
  $\underbrace{V[i,j] = \max(V[i-1,j], v_i + V[i-1,j-w_i])}_{\text{suffit donc de remplir ligne par ligne, de haut en bas}}$
- L'algorithme détaillé coûtera $\Theta(nW)$ opérations (accès au tableau, sommes, comparaisons)