

Programmation dynamique

Résumé de Cours

octobre 2002

C'est un "paradigme" de programmation qu'on peut voir comme une amélioration ou une adaptation du "diviser pour régner". Il a été introduit par Bellmann pour des problèmes d'optimisation (recherche opérationnelle) en 1955.

Quand? Si:

.La solution (optimale) d'un problème de taille n s'exprime en fonction de la solution (optimale) de problèmes de taille $< n$.

.Une implémentation récursive "naïve" conduit à calculer de nombreuses fois la solution de mêmes sous-problèmes.

Comment?

. On définit une table: à chaque élément correspondra la solution d'un et d'un seul problème intermédiaire. Il faut donc qu'on puisse définir chaque sous-problème qui sera traité au cours du calcul... Ensuite il faut remplir cette table; il y a deux approches, l'une itérative, l'autre récursive:

- Version itérative:

.on initialise les "cases" correspondant aux cas de base.

.on remplit ensuite la table selon un ordre bien précis (à déterminer): on commence par les problèmes de "taille" la plus petite possible, on termine par la solution du problème principal: **il faut bien sûr qu'à chaque calcul, on n'utilise que les solutions déjà calculées**. Le but est bien sr que chaque élément soit calculé une et une seule fois.

- Version récursive (tabulation, memoizing):

A chaque appel, on regarde dans la table si la valeur a déjà été calculée (donc une "case" correspond à un booléen et une valeur ou une seule valeur si on utilise une valeur "sentinelle"). Si oui, on ne la recalcule pas: on récupère la valeur mémorisée; si non, on la calcule et à la fin du calcul, on stocke la valeur correspondante.

Remarques:

. Le plus gros du travail réside dans l'expression d'une solution d'un problème en fonction de celles de problèmes "plus petits". Si on se rend compte qu'on est amené à recalculer plusieurs fois la solution de mmes problèmes, on est dans le cadre de la programmation dynamique. Le découpage du problème devrait naturellement conduire à la définition de la table (qui peut tre de dimension 1,2,3,...). Attention, le nombre de sous-problèmes peut être grand et l'algorithme obtenu n'est pas forcément polynomial. Ensuite, il faut encore réfléchir un petit peu ... pour savoir comment effectuer le remplissage de la table, tout du moins dans le cas itératif!

.La version récursive est souvent un peu plus simple à écrire et elle n'est pas forcément moins efficace que l'itérative: en effet, on ne calcule que les appels

réellement nécessaires, alors que dans le cas itératif, pour certains problèmes on calcule des valeurs inutiles.

. Le principe "la solution optimale d'un problème peut s'obtenir à partir des solutions optimales de sous-problèmes" est appelé dans la littérature *le principe d'optimalité (de Bellman)*. Bien que naturel, le principe n'est pas toujours applicable! Prenons l'exemple des chemins dans un graphe: le principe marche si on cherche le plus court chemin entre deux points (cf algo Floyd): si le chemin le plus court entre A et B passe par C, le tronçon de A à C (resp. de C à B) est le chemin le plus court de A à C (resp. de C à B); par contre ça ne marche plus si on cherche le plus long chemin sans boucle d'un point à un autre!

Programmation dynamique et Optimisation

La programmation dynamique est souvent utilisée dans le cadre des problèmes d'optimisation: on cherche une solution optimale par rapport à un certain cot (ou fonction objectif) et donc dans ce cas c'est non seulement le meilleur cot qu'on cherche, mais bien sûr aussi une solution qui corresponde à ce cot. Or souvent, la table construite dans le cadre de la programmation dynamique contient des cots et non des solutions. Dans une première phase, on calcule donc seulement le cot d'une solution optimale; il est cependant souvent facile ensuite de récupérer la (une) solution optimale, toujours en utilisant l'information contenue dans la table (éventuellement en enrichissant un petit peu la table: on peut mémoriser pour une valeur d'o elle vient...); grosso modo, on parcourt la table à l'envers par rapport à sa construction, i.e. en partant de la case correspondant au problème principal et en "remontant" vers les plus petits problèmes.

Quelques exemples classiques: Il y a de nombreux exemples d'algorithmes "classiques" qui utilisent le principe de la programmation dynamique: l'algorithme de Warshall-Floyd (clture transitive d'une relation, plus court chemin dans un graphe), les arbres binaires optimaux, la triangulation d'un polygone, le calcul de la distance de deux canes (cf *diff* d'Unix), les problèmes d'alignement de séquences, le problème du sac à dos, l'algorithme de Viterbi...