Programmation Effective – TD 12 : Union Find

matthieu.gallet@ens-lyon.fr

mardi 13 mai 2008

1 Union Find

On cherche à créer une structure de données d'ensemble disjoints, utilisant une collection $S = \{S_1, S_2, \ldots, S_k\}$ d'ensembles dynamiques disjoints. Chaque ensemble va être identifié par un représentation, qui sera l'un de ses membres. Si x est un élément quelconque, on note S_x l'ensemble le contenant. On veut disposer de 3 opérations de base :

- $\mathrm{Crer-Ensemble(x)}$, qui crée un nouvel ensemble à partir de x (x n'étant dans aucun autre ensemble).
- $\mathrm{Union(x, y)}$, qui calcule l'union $S_x \cup S_y$
- $\mathrm{Trouver(x)}$, qui renvoie un pointeur vers l'unique ensemble S_x contenant x.

Une application classique de cette structure de données est l'utilisation de composantes connexes dans un graphe, ou la création de labyrinthes.

2 Implémentation avec des listes chaînées

L'idée est tout simplement de représenter chaque S_i par une liste chaînée, et le représentant sera le premier élément de cette liste. Chaque élément de la liste contiendra un élément x, un pointeur vers l'élément suivant et vers un pointeur vers le représentant. Chaque liste doit également avoir un pointeur vers sa tête et vers sa queue.

$\mathrm{Crer-Ensemble(x)}$ et $\mathrm{Trouver(x)}$ sont faciles à implémenter en temps $\mathcal{O}(1)$. Quant à l'union, elle se fait en concaténant les deux listes, en temps moyen $\Theta(n)$ (cependant il est plus rapide de concaténer la liste la plus courte à la plus longue).

3 Implémentation avec des forêts d'ensembles

Dans cette implémentation, chaque ensemble S_i est un arbre, dont la racine est le représentant. Chaque nœud pointe uniquement sur son parent, sauf la racine qui pointe vers elle-même. Deux heuristiques sont souvent utilisées pour améliorer le temps d'exécution :

- *union par rang* : le principe est de faire pointer la racine de l'arbre contenant le moins de nœuds vers celle contenant le plus de nœuds. Plutôt que de gérer directement le nombre de nœuds, on préfère utiliser le rang, qui est un majorant de la hauteur du nœud, et ainsi on fait pointer la racine de moindre rang vers celle de rang supérieur lors d'une union.
- *compression de chemin* : l'idée est de faire pointer chaque nœud directement vers la racine.

4 Applications

Résolvez les problèmes suivants :

- An antiarithmetic permutation (`http://acm.uva.es/p/v111/11129.html`),
- Network Connections (`http://acm.uva.es/p/v7/793.html`).

- Monkeys in a Regular Forest (`http://acm.uva.es/p/v7/776.html`),
- Problem D : Blackbeard the Pirate (`http://acm.uva.es/p/v109/10937.html`),
- Problem F : AntiFloyd (`http://acm.uva.es/p/v109/10987.html`),