# Algorithmique et programmation
# Travaux dirigés, 14 et 15 décembre 1998
Louis Granboulan

## 1 Ensembles

Nous voulons réaliser efficacement les opérations de manipulation d'ensembles, à l'aide de tableaux où de structures chaînées. Nous devons d'abord distinguer le cas où les éléments des ensembles manipulés sont choisis parmi un même ensemble de taille raisonnable ou s'ils sont choisis dans un ensemble presque infini. Nous appellerons univers cet ensemble de tous les éléments considérés. Nous devons aussi distinguer le cas où il existe une relation d'ordre sur les éléments de l'univers, ce qui est nécessaire pour certaines opérations.

1. **ELEMENT** : donne un élément de l'ensemble. Cette opération permet de savoir si l'ensemble est vide.
2. **APPARTENANCE** : savoir si un élément est membre de l'ensemble.
3. **INSERTION** : ajout d'un élément sachant qu'il n'est pas membre.
4. **SUPPRESSION** : enlèvement d'un élément sachant qu'il est membre.
5. **INTERSECTION** : intersection de deux ensembles.
6. **UNION** : réunion de deux ensembles disjoints.
7. **LOCALISATION** : trouver, parmi une collection d'ensembles formant une partition de l'univers, lequel contient un élément donné.
8. **MINIMUM** : extrait le plus petit élément de l'ensemble.
9. **PARTAGE** : un pivot sert à séparer l'ensemble en deux parties : les éléments plus petits et les éléments plus grands que le pivot.
10. **CONCAT** : réunion de deux ensembles tels que tous les éléments de l'un soient inférieurs à tous les éléments de l'autre.

On appelle **dictionnaire** une structure de données permettant les opérations APPARTENANCE, INSERTION et SUPPRESSION. Une **file de priorité** doit permettre INSERTION, SUPPRESSION et MINIMUM.
On s'intéresse à la complexité d'une opération, ou bien d'une séquence d'opérations, dans le cas le pire ou en moyenne.

1. Remarques générales
   (a) Que pouvez-vous faire comme remarques sur les définitions de ces opérations ?
   (b) Quelles sont celles qu'on peut réaliser avec une ou plusieurs autres ? Quelle est la complexité de ceci ?
2. Exemples
   (a) Existe-t-il une structure de données pour faire les opérations ELEMENT, APPARTENANCE, INSERTION et SUPPRESSION en temps constant ?
   (b) Peut-on faire INSERTION et MINIMUM en temps constant ?
   (c) Existe-t-il une structure de données pour faire une ou plusieurs des opérations INTERSECTION, UNION et LOCALISATION en temps constant ?

## 2 Structures de données élémentaires

1. Liste chaînée

   Chaque ensemble est représenté par une liste chaînée de ses éléments.

   (a) Quelles sont les limitations de cette représentation ?
   (b) Quelles sont les opérations qu'on peut réaliser efficacement avec cette représentation ? Leur complexité ?
   (c) Qu'y gagne-t-on à utiliser une liste doublement chaînée ?
2. Vecteur de bits

   Les éléments de l'univers sont numérotés et on représente un ensemble par sa fonction caractéristique, sous la forme d'un vecteur de bits.

   (a) Quelles sont les limitations de cette représentation ?
   (b) Quelles sont les opérations qu'on peut réaliser efficacement avec cette représentation ? Leur complexité ?

## 3 Hachage

On se donne fonction $h$ de l'univers dans l'intervalle $\{1, ..., n\}$. Cela permet de représenter dans une table indicée par $\{1, ..., n\}$ tous les ensembles pour lesquels $h$ n'a pas de collision.

(a) Quelles sont les opérations qu'on peut réaliser efficacement ainsi, et quelle est leur complexité ?
(b) Comment gérer les collisions ? Quelle est la complexité dans ce cas ?

# 4 Arbres

1. Tas

   Dans un univers ordonné, on représente un ensemble par un arbre binaire dont les nœuds internes et les feuilles contiennent les valeurs des éléments et tel que la valeur des fils soit supérieure à la valeur du père. On veut aussi que l'arbre soit quasi-complet (les feuilles sont de hauteur $h$ ou $h-1$ et celles de hauteur $h-1$ sont à droite).

   (a) Montrer qu'un tas permet de gérer les opérations d'une file de priorité en $\mathcal{O}(\log n)$ instructions.

   (b) Que dire des autres opérations ?

2. Arbre binaire de recherche

   Dans un univers ordonné, on représente un ensemble par un arbre binaire dont les nœuds sont étiquetés de telle sorte qu'un parcours infixe donne une suite croissante. On dit que l'arbre est équilibré (on parle d'arbre AVL, d'après Adel'son, Vel'skii et Landis) si pour tout nœud la différence entre son sous-arbre gauche et son sous-arbre droit n'excède pas 1.

   (a) Quelle est la complexité des opérations APPARTENANCE, INSERTION, SUPPRESSION et MINIMUM avec une représentation des ensembles par des arbres de recherche non nécessairement équilibrés.

   (b) Encadrer le nombre de nœuds d'un arbre AVL de hauteur $h$.

   (c) Montrer que pour une représentation d'un ensemble de taille $n$ par un arbre AVL, les quatre opérations ci-dessus peuvent être effectuées en $\mathcal{O}(\log n)$ instructions.

   (d) Que dire des autres opérations ?

   (e) Comment construire un arbre binaire de recherche pour un ensemble donné, tel que le coût moyen de l'opération APPARTENANCE soit minimal, connaissant une distribution de probabilité sur l'univers ?

3. Arbre 2-3

   On représente un ensemble par un arbre dont les nœuds internes ont deux ou trois fils et dont toutes les feuilles sont à la même hauteur. Les feuilles contiennent les valeurs des éléments.

   (a) Encadrer le nombre de nœuds d'un arbre 2-3 de hauteur $h$.

   (b) Quelles sont les opérations qu'on peut effectuer en $\mathcal{O}(\log n)$ instructions ?

4. Manipulation d'une partition de l'univers

   On s'intéresse à un univers de taille $n$ et aux opérations d'UNION et LOCALISATION. On représente chaque ensemble par un arbre dont les nœuds sont étiquetés par les éléments de l'ensemble.

   (a) L'opération d'UNION se réalise naturellement en temps constant, en ajoutant l'un des deux arbres comme fils supplémentaire de la racine de l'autre. Montrer qu'en l'absence de rééquilibrage de l'arbre, une opération de LOCALISATION peut prendre un temps $\mathcal{O}(n)$.

   (b) Montrer que si pour une UNION, on choisit de rajouter le plus petit des deux ensembles comme fils supplémentaire de la racine du plus grand (en nombre de nœuds), une opération de LOCALISATION prend un temps $\mathcal{O}(\log n)$.

   (c) On suppose qu'à chaque LOCALISATION, on modifie l'ensemble trouvé de telle sorte que tous les ancêtres de l'élément cherché deviennent des fils directs de la racine. On définit $F(0) = 1$ et $F(i+1) = 2^{F(i)}$, puis $G(n) = \min\{k \text{ tq } F(k) \geqslant n\}$.

      On s'intéresse à l'exécution d'une séquence de $\alpha n$ opérations d'UNION et LOCALISATION à partir d'une partition de l'univers en singletons.

      On définit le *rang* d'un élément $v$ comme sa hauteur dans la forêt résultant de la séquence des opérations d'UNION seules. On partitionne les éléments en groupes, selon l'image par $G$ de leur rang.

      i. Montrer qu'il y a au plus $n/F(g)$ éléments de groupe $g$.

      ii. On sait que pour chaque opération de LOCALISATION, le coût est le nombre d'ancêtres de l'élément cherché. On va répartir le coût comme suit : si $v$ est la racine, ou bien si $v$ et son père sont dans des groupes distincts, le coût attribué directement aux opérations de LOCALISATION est incrémenté de 1. Sinon, c'est le coût attribué à l'élément $v$ qui est incrémenté de 1.

          Montrer que le coût attribué directement à chaque opération de LOCALISATION est au maximum $G(n)$ et que le coût attribué à l'ensemble des éléments d'un même groupe est au plus $n$.

      iii. En déduire que l'ensemble des opérations de LOCALISATION se fait en temps $\mathcal{O}((1+\alpha)nG(n))$.

# 5 Application

Soit un graphe non orienté connexe $(V, E)$ dont les arêtes ont un poids réel. On cherche une arborescence recouvrante de poids minimal. Voici un algorithme qui va construire l'ensemble $T$ des arêtes de l'arborescence recouvrante. Le principe de l'algorithme est de partir d'une forêt recouvrante de poids minimal et de la rendre connexe.

On manipule l'ensemble $VS$ dont les éléments sont l'ensemble des sommets pour chaque arborescence de la forêt. C'est donc une partition de $V$. On initialise $T$ à l'ensemble vide et $VS$ à la partition en singletons.

À chaque étape, on choisit l'arête de poids minimal. On la suprime de $E$. si ses deux extrémités sont dans des éléments de $VS$ distincts, on rajoute cette arête à $T$ et on réunit les deux éléments de $VS$ correspondants. On s'arrête quand $VS$ est un singleton.

1. Montrer que cet algorithme est correct
2. Évaluer sa complexité.

La taille de l'univers est $u$, la taille de l'ensemble est $n$.

| | ELEMENT | APPARTENANCE | INSERTION | SUPPRESSION | INTERS. | UNION | LOCALIS. | MINIMUM | PARTAGE | CONCAT |
|---|---|---|---|---|---|---|---|---|---|---|
| Liste chaînée | 1 | $n$ | 1 | $n$ | $n$ | $n$ | | - | - | - |
| Liste triée | 1 | $n$ | $n$ | $n$ | $n$ | $n$ | | 1 | $n$ | 1 |
| Tas | 1 | $n$ | $\log n$ | $\log n$ | | | | 1 | | |
| Arbre AVL | 1 | $\log n$ | $\log n$ | $\log n$ | | | | $\log n$ | | |
| Arbre 2-3 [1] | 1 | $n$ | $\log n$ | $\log n$ | | $\log n$ | $\log n$ | 1 | | |
| Arbre 2-3 trié [2] | 1 | $\log n$ | $\log n$ | $\log n$ | | | $\log n$ | $\log n$ | $\log n$ | $\log n$ |
| Vecteur de bits | $u$ | 1 | 1 | 1 | $u$ | $u$ | | $u$ | $u$ | $u$ |
| Forêt comprimée | 1 | $G(u)$ | $G(u)$ | $G(u)$ | | $G(u)$ | $G(u)$ | | | |

[1] Les feuilles sont en ordre quelconque et chaque nœud est étiqueté par la valeur minimum de son sous-arbre.

[2] Les feuilles sont en ordre croissant et chaque nœud est étiqueté par la valeur minimum de chacun de ses sous-arbres.