Université de Montréal

TP2 : Monceaux p-aires

Par

Farzin Faridfar - 20031884

Catherine Laprise - C4411

Baccalauréat en informatique

Faculté des Arts et des Sciences

Travail remis le 8 mars 2016

Dans le cadre du cours IFT2015

Structures de données

# Graphique des résultats

# Analyse théorique

Complexité des opérations (dans le pire cas)

|  |  |  |
| --- | --- | --- |
|  | Insertion | Suppression du minimum |
| QP (tri par insertion) |  | ) |
| QP (liste circulaire) |  |  |
| Monceau binaire |  |  |
| Monceau 3-aire |  |  |
| Monceau 4-aire |  |  |
| Monceau 5-aire |  |  |
| Monceau p-aire |  |  |

# Analyse numérique

En premier lieu, on peut constater que lorsque n est très petit, la différence entre les différentes structures est négligeable. Même si la complexité de l’insertion varie entre en dans le cas des QP et en dans le cas des monceaux (où p est l’arité du monceau), pour n <= 1000, la différence entre les deux n’est pas très significative – il est quand même assez rapide de parcourir la liste en entier pour faire le tri insertion. Toutefois, à partir de n = 5000, on commence à voir une différence très marquée entre les queues de priorité (QP\_insertion et QP\_circulaire) et les files basées sur un monceau puisque plus n est grand, plus l’écart entre n et son log augmente. Nous avons d’ailleurs arrêté de tester les QP après n = 25000 car le temps de calcul était très long et l’écart était si grand que les résultats n’étaient plus significatifs.

Les QP insertion et circulaire retournent des résultats assez semblables, malgré le fait que l’opération de suppression du minimum soit plus efficace dans le cas de la file circulaire. En effet, même lorsqu’on doit faire appel à la suppression du minimum, il faut quand même ensuite retrier la liste, ce qui se fait en dans les deux types de structures. Ainsi on a l’opération de suppression de minimum et d’ajout de la nouvelle valeur qui se fait en pour le QP insertion et en pour le QP circulaire, ce qui est dans tous les cas en De plus, malgré les économies réalisées lors de la suppression, l’insertion dans la liste circulaire demande plus d’opérations que celle dans la liste régulière puisqu’on doit toujours prendre le modulo des index pour s’assurer d’être à la bonne position dans la file.

En ce qui concerne les monceaux, les performances sont assez similaires car la complexité de toutes les opérations pour les monceaux est en On commence à remarquer une légère différence entre les monceaux de différentes arités lorsque n est grand, aux alentours de n = 50000. Effectivement, comme on doit conserver les n premières valeurs reçues, avec un très grand n, il est garanti qu’on doive effectuer un grand nombre d’insertion. Or, lorsqu’on fait varier l’arité des monceaux, on a que :

* L’opération *upheap* est plus rapide quand p est grand car on a moins de niveaux à parcourir (donc l’insertion est plus rapide).
* L’opération *downheap* est plus rapide quand p est petit car on a moins d’enfants à comparer à chaque niveau (donc la suppression est plus rapide).

Ainsi, aux deux extrêmes (p=2 ou p=31) on a une perte d’efficacité dans un sens ou dans l’autre.

Toutefois, la différence est plutôt négligeable en comparaison aux performances offertes par les QP insertion ou circulaire.

# Discussion et conclusion

On peut conclure facilement que l’utilisation des files basées sur des monceaux est beaucoup plus efficaces.

Particulièrement, les monceaux 4-aires et 5-aires offrent d’excellentes performances puisque, comme discuté précédemment, elles ont un meilleur rapport entre le nombre d’enfants par niveau et le nombre de niveaux nécessaire pour représenter l’arbre complet.

Nous supposons que l’avantage d’utiliser l’arbre binaire réside dans la facilité à coder la structure. En effet, il n’est pas nécessaire pour faire un *downheap* d’utiliser des boucles pour comparer les enfants. Une seule opération de comparaison suffit. On peut également facilement désigner par deux variables un enfant gauche et un enfant droit, et rapidement vérifier si ces enfants existent sans avoir recours à des fonctions supplémentaires. Ceci résulte en un code plus compact et facile à lire et modifier.

Au niveau de l’optimisation du code pour les monceaux p-aire, nous avons constaté qu’à chaque *downheap,* on doit dans tous les cas comparer les enfants pour trouver celui ayant la valeur minimale (ceci nécessite dans le pire cas p-1 comparaisons). L’opération est donc de l’ordre de , peu importe si p est variable ou non. Nous croyons donc qu’il n’y a pas d’avantage à utiliser un monceau dont l’arité est fixe en comparaison à un monceau avec un p variable, à l’exception du monceau binaire.