IFT2810 - Automne 2009

Structures de données

Files avec priorité

Arbres

Graphes

Sylvie Hamel

André-Aisenstadt: **3161**
hamelsyl@iro.umontreal.ca

http://www.iro.umontreal.ca/~hamelsyl/IFT2810-A09

Horaire et locaux:

Cours théorique : MARDI 18H30 à 20h30

Local: AA 1355

Travaux pratiques : MARDI 20H30 à 21h30

Local: AA 1355

Objectifs:

Le cours IFT2810 vise à familiariser l'étudiant avec les techniques de base pour l'organisation, la manipulation et la recherche de données. Le but du cours est d'être capable de choisir les structures de données qui rendent un programme le plus efficace possible, que ce soit en terme du temps d'exécution ou de l'espace mémoire utilisé.

Manuel obligatoire:

Robert Sedgewick, **Algorithmes en Java, 3ieme edition**, Pearson Education, 2004.

- Livre orienté pour les programmeurs JAVA
- Livre est complet et facile à lire
- Disponible à la librairie de l'université (une quinzaine de copies)
- Disponible aussi en anglais (version originale)

Robert Sedgewick, **Algorithms en Java, Parts 1-4, 3rd edition**, Pearson Education, 2003.

Contenu:

1. Introduction: Mesure de complexité, types abstraits de données, récursivité. (Chapitres 1, 2)
2. Piles et Files et Listes Chaînées (Chapitres 3, 4)
3. Arbres binaires, Files avec priorité et Tas (Chapitre 5, 9)
4. Dictionnaires: Tables et fonctions de hachage (Chapitre 14)
5. Arbres binaires de recherche, arbres AVL, brève description des arbres rouge-noir et 2-4 (Chapitre 13)
6. Graphes: Introduction et parcours de graphes. (Notes de cours)

Manuels de références:

1. Goodrich M.T., Tamassia R., **Data structures and algorithms in Java (4th edition)**, John Wiley et Sons, Inc., 2006.

2. Mark Allen Weiss, **Data Structures and Algorithm Analysis in Java (2nd edition)**, Pearson Education, 2007.

3. T.A. Standish, **Data Structures in JAVA**, Addison Wesley, 1998.

4. A. Aho, J.E. Hopcroft and J.D. Ullman, **Data Structures and Algorithms**, Addison Wesley, 1983.

5. D.E. Knuth, **The Art of Computer Programming (Second Edition)** , Addison Wesley, 1973.

Travaux Pratiques: les mardis juste après le cours

Démonstrateur: à confirmer

Disponibilité: à confirmer

Mes disponibilités:

- Avant le cours

ou

- Sur rendez-vous

Évaluation:

2 devoirs:

- Pondération: 15 % chacun
- TP1: - reception de l'énoncé: **22 septembre**
  - remise du devoir: **6 octobre**
- TP2: - reception de l'énoncé: **10 novembre**
  - remise du devoir: **24 novembre**

Tout retard dans la remise des travaux entraînera une pénalité de 10% par jour (24 heures).

Intra (Mardi 27 octobre):

- Pondération: 30 %
- Durée: 2 heures

Final (Mardi 15 décembre):

- Pondération: 40 %
- Durée: 3 heures

Un seuil cumulatif de 50% aux examens sera appliqué (moyenne de 50% pour les deux examens combinés)

Plagiat:

- Pour vos travaux, vous pouvez utiliser tout ce qui est disponible et accessible publiquement en autant que les références utilisées soient proprement citées.
- Je considère que tout étudiant à lu, signé et respectera le code d'honneur du DIRO
- Les cas de plagiat seront traités conformément aux règles en vigueur à l'Université de Montréal.

Qu'est-ce qu'une structure de données:

Le terme structure de données, représente à la fois:

- un moyen pour organiser les données en mémoire

- des méthodes (algorithmes) pour manipuler ces données

  ajout et suppression efficace de données
  accès aux données, etc...

Pourquoi étudier les structures de données?

- tous les langages de programmation utilise le même genre de structures de données

- le bon choix de structures peut faire augmenter l'efficacité des algorithmes implémentés que ce soit au niveau de l'efficacité en temps ou de l'efficacité en espace mémoire

- d'un point de vue théorique, beaucoup de structures utilisent des algorithmes très intéressants pour l'accès, l'ajout et la suppression de données

Comment choisir la bonne structure?

Nous verrons pendant ce cours beaucoup de structures de données différentes

- tableau
- liste chaînée
- pile
- file
- tas ou monceau
- arbres de recherche
- table de hachage
- graphes, etc.

Le choix de la "meilleure" structure de données dans un cas particulier dépendra en grande partie des opérations sur les données les plus utilisées dans ce cas.

> Par exemple, si l'accès aux données est très importante mais l'ajout et le retrait beaucoup moins, nous choisirons une structure permettant un accès très efficaces aux données, même si cette structure est moins efficace pour l'ajout et le retrait de données.