Informations concernant l'examen final INF4100

- **Quand** : L'examen final a lieu le mercredi 23 avril 2008, de 9h00 à 12h00
- **Où** (selon votre code permanent):

  SH-3580 : AUBC26088409–JAOM05107704
  SH-3720 : LACA30098309–TURC07038500

- **Forme** : Examen à livre ouvert

Thèmes couverts :

3 Programmation dynamique :

- Pour certains problèmes, une solution récursive diviser-pour-régner n'est pas efficace à cause de la présence de *sous-problèmes superposés*.
- Stratégie = Définir des équations récursives qui décrivent/*caractérisent la solution optimale* + Mémoriser les solutions des sous-problèmes dans une structure de données auxiliaire.
- Solution itérative ascendante : on analyse les *dépendances de données* pour déterminer dans quel ordre remplir la structure de données.

- Exemples : Nombres de Fibonacci ; Multiplication de chaînes de matrices ; Sac à dos 0–1 ; Coefficients binomiaux ; Algorithme de Floyd pour les plus courts chemins ; Distance d'édition entre deux chaînes ; Déplacement d'un robot sur une grille bi-dimensionnelle ; Affectation d'items à des boites pour envois ; Somme maximum des sous-séquences d'éléments contigus.

4 Algorithmes voraces :

- Algorithme où on effectue une série de choix, chaque choix se faisant relativement «rapidement», sur la base d'informations locales, sans jamais revenir en arrière — sans jamais «changer d'idée» quant à un choix préalablement effectué.
- Exemples : Rendre la monnaie ; Sac à dos fractionnaire ; Encodage de Huffman ; Arbre couvrant minimum (Prim vs. Kruskal) ; Traversée du désert avec ravitaillement en eau.
- Note : Pour certains problèmes d'optimisation, il peut exister un algorithme vorace qui produit une solution optimale, mais ce n'est pas le cas pour tous les problèmes.
- Files de priorité : Trois (3) mises en oeuvre : séquence non-ordonnée/ordonnée et monceau. Divers exemples d'utilisation : Tri (sélection, insertion, *heapsort*) ; Sac à dos fractionnaire ; Encodage de Huffman.

1' Algorithmes avec *backtracking* et algorithmes *branch-and-bound*

- Exploration de graphes : Algorithmes pour explorer, une fois, les divers sommets (ou arêtes/arcs) d'un graphe :
  - En profondeur : On explore un voisin, puis le voisin de ce voisin, etc. ⇒ Mise en oeuvre récursive ou avec une pile.
  - En largeur : Une «couche» de voisins adjacents à la fois ⇒ Mise en oeuvre avec une file FIFO (*First-In First-Out*).

- Algorithmes avec *backtracking*
  - Algorithme où on effectue une série de choix, mais avec possibilité de «retour en arrière», de façon à explorer l'ensemble des solutions acceptables d'un espace de solutions.
  - Exemples : Affectation de n tâches à n agents ; Sac à dos 0–1.

- Algorithmes *branch-and-bound*
  - Algorithme où on effectue une série de choix, avec possibilité de «retour en arrière» et avec *élagage* possible des solutions qui ne semblent pas «intéressantes» (au niveau de leur coût), et ce de façon à réduire l'espace de solutions qui est exploré.
  - Exemples : Affectation de n tâches à n agents ; Sac à dos 0–1.