Algorithmes pour les graphes

24 novembre 2016

1 Représentation des graphes

1. Rappeler la définition d'un graphe orienté.
2. Structure de données : **ensemble** de sommets (vertex, vertices) V et **liste** d'arcs E (edges) qui sont des couples d'éléments de V. On peut rajouter des attributs aux sommets et aux arcs. Nombre de sommets N_v. Nombre d'arcs : N_E.
3. Autres représentations : matrice d'adjacence, liste d'adjacence.
4. Installer python. Suivre un tutoriel python pour apprendre les fonctionnalités de base.
5. En python programmer le passage d'une représentation à une autre.
6. Programmer la version efficace de la liste d'adjacence.
7. Installer cairo pour python et igraph. Faire le tutoriel igraph pour vérifier que tout marche bien.
8. Apprendre à créer dans igraph un graphe orienté (indication : directed = True), un graphe valué.

2 Parcours des graphes

1. Donner les algorithmes de parcours en largeur et en profondeur.
2. Les programmer dans python. Utiliser une pile (stack) pour le parcours en profondeur (depth-first search, DFS). Utiliser une file (queue) pour le parcours en largeur (breadth-first search BFS).
3. Graphe direct acyclique (DAG). Tri topologique, algorithme.
4. Programmer le tri topologique dans python. Vérifier les résultats avec igraph.

3 Chemins optimaux dans les graphes

1. Définir rigoureusement le problème de chemin optimal dans un graphe orienté.
2. Donner l'algorithme de Dijkstra. Quel est sa complexité ?
3. Exercices : feuilles de TD 1 et 2 de `http://www-irma.u-strasbg.fr/~helluy/rechop/rechop.html`. Traiter les exercices "à la main", puis avec igraph. Pouvez-vous faire tous les exercices ?
4. Algorithme de Bellman-Ford. Complexité ?
5. Programmer l'algorithme de Bellman-Ford en utilisant python et igraph.
6. Terminer les feuilles de TD de la question 3.

4 Flots optimaux dans les graphes

1. Poser le problème du flot optimal.
2. Conséquence de la loi des noeuds. Arc fictif de retour.
3. Notion de chaîne augmentante. Théorème du flot optimal.
4. Algorithme de de Ford-Fulkerson.
5. Exercices : `http://www-irma.u-strasbg.fr/~helluy/rechop/rechop-td4.pdf`
6. Vérifier les exercices avec igraph.

5 Algorithme de Cuthill–McKee

1. Approximation du problème de Dirichlet sur un rectangle par différences finies
2. Matrice ligne de ciel. Largeur de bande.
3. Programmer en C la résolution du problème. Afficher la matrice. Constater l'effet d'une renumérotation.
4. Algorithme de Cuthill-McKee.
5. Programmer l'algorithme de Cuthill-McKee en C. On pourra utiliser la bibliothèque igraph.
6. Comparer le temps de calcul avec la nouvelle numérotation.

Références

[1] The igraph core team. Python igraph tutorial. `http://igraph.org/python/doc/tutorial/tutorial.html`. Accessed : 2016-09-15.

[2] Jaehyun Park. Introduction to programming contests. basic graph algorithms. `https://web.stanford.edu/class/cs97si/`. Accessed : 2016-09-15.