Algorithme de Kruskal et application au problème du voyageur de commerce

Mihai Dusmanu, Clément Pascutto

Mardi 15 Décembre 2015

Quelques précisions sur l'implémentation...

Tri

L'agorithme de tri implémenté est un tri hybride qui utilise un quick sort pour les tabeaux de grande taille, et un tri par insertion pour les tableaux de petite taille.

Après plusieurs tests qui seront détaillés dans la présentation, le seuil optimal entre ces deux tri est atteint pour des tableaux de taille 20.

Union-find

Nous avons implémenté les deux heuristiques pour la structure de données union-find, qui permet d'atteindre une complexité amortie en $\alpha(p, q)$, où p est le nombre d'opérations effectuées sur la structure, et q est la taille de la structure.

Générateur de graphes

L'algorithme de génération de graphes connexe (nécessaire pour trouver un ACM) est le suivant :

- On ajoute chaque sommet au graphe dans un ordre aléatoire, en le reliant à l'un des sommets (choisi au hasard) déjà présent.
  Ceci assure la connexité du graphe.
- Pour arriver au nombre d'arêtes demandé, on ajoute des arêtes dont les deux extrêmités et le poids sont choisis au hasard.

Programmes vérificateurs

Algorithme de Kruskal (ok_generator.cpp, grader_eval_kruskal.cpp)

La source ok_generator.cpp, écrite en C++, qui calcule le poids d'un ACM, a été vérifiée par un *online judge*.

L'évaluateur grader_eval_kruskal.cpp vérifie la correction de l'ACM fourni (poids minimum et arbre couvrant).

Problème du voyageur de commerce (ok_generator.cpp, grader_eval_tsp.cpp)

En considérant la preuve de 2-approximation de l'algorithme, on observe qu'il suffit de comparer le poids fourni au double du poids d'un ACM.