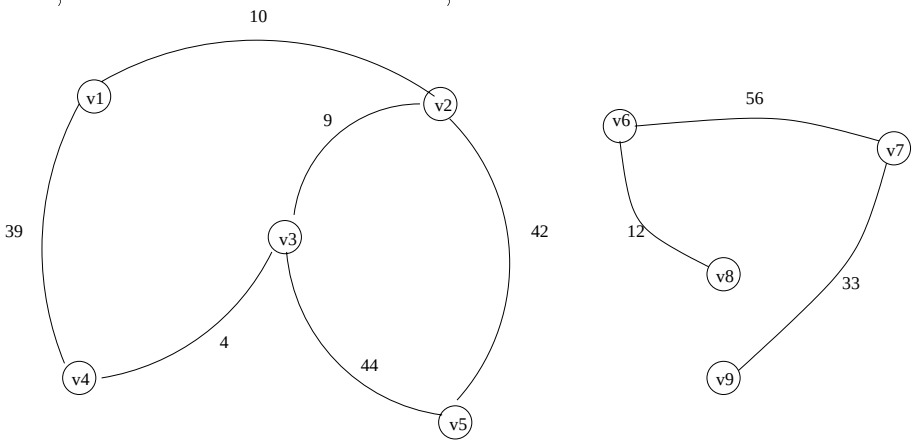
INF4100
Laboratoire no. 11

16/04/08

(Explorations de graphes, algorithmes marche arrière et *branch-and-bound*, fignolage de code)

1. Explorations de graphes

Soit le graphe suivant, formé de deux composantes connexes — une composante connexe est un groupe de sommets qui sont reliés, directement ou indirectement, les uns aux autres :



a. Indiquez dans quel ordre seront visités les différents sommets si on effectue...

   (a) Une fouille en profondeur.

   (b) Une fouille en largeur.

b. Même question, mais cette fois pour une visite des différentes arêtes, en supposant que les arêtes sont explorées dans un ordre qui correspond à l'ordre de leurs sommets — pour visiter les arêtes on visite aussi par la même occasion les sommets correspondants. (Voir aussi la question suivante.)

2. Explorations de graphes

Le programme `exploration-graphe-labo.mpd` (cf. fichier d'archive `Graphes.tar.gz` sur le site Web) définit divers types et procédures permettant de traiter des graphes. Deux procédures de parcours des sommets — en profondeur et en largeur — sont définies.

a. Définissez une procédure `explorerAretesEnProfondeur( Graphe G )` qui permet d'explorer, avec une fouille en profondeur, les diverses arêtes d'un graphe `G` — voir les procédures déjà définies pour le parcours des sommets quant à ce qui doit être fait lors d'une visite.

   Voici un exemple du résultat attendu pour le graphe `g0` :

```
--- Aretes en profondeur ---
     on visite (1, 4)
     on visite (2, 3)
       on visite (3, 5)
         on visite (5, 2)
         on visite (5, 6)
```

   Les arêtes n'étant pas dirigées, il ne faut évidemment pas les visiter deux fois.

b. Définissez une procédure `genererConnexesEnProfondeur( Graphe G )` qui permet d'explorer, en profondeur, les diverses sommet d'un graphe dans le but d'associer à chaque sommet un numéro de composante connexe, et aussi d'imprimer les divers sommets associés à une composante connexe, en indiquant le numéro de cette composante.

Voici un exemple du résultat attendu pour le graphe `g0`, en plus du fait qu'à chaque sommet est maintenant associé son numéro de composante :

```
--- Composantes connexes en profondeur ---
Composante 1:: 1 4
Composante 2:: 2 3 5 6
```

3. Algorithme *branch-and-bound* pour affectation de tâches

Dans l'exemple vu en classe, l'algorithme avec marche arrière explorait six (6) possibilités — six affectations complètes possibles. Par contre, l'algorithme *branch-and-bound* n'explorait que trois (3) affectations complètes.

Trouvez un exemple, avec trois agents et trois tâches, où l'algorithme *branch-and-bound* explore autant d'affectations complètes que l'algorithme avec marche arrière.

Note : Voir le site Web pour le fichier `agents.mpd`.

Exercices tirés des notes de cours :

- Série 6 (p. 231–232) : 0,1

Note : Voir le site Web pour le fichier `Ensembles.mpd`, qui définit un type abstrait pour des *bitsets* — ensembles de «bits», plus précisément, ensembles dont les éléments proviennent d'un intervalle $1..n$.

---

4. Fignolage de code

Le site Web contient un programme `integration.mpd` qui permet de calculer, de façon numérique, l'intégrale suivante :

$$\int_a^b f(x) \qquad \text{où} \quad f(x) = \frac{4}{x^2+1}$$

Plus précisément, lors de l'appel du programme, il faut spécifier les arguments suivants :

- a et b : Bornes entre lesquelles l'intégrale doit être calculée. Lorsque $a = 0.0$ et $b = 1.0$, le résultat de l'intégrale devrait être égal à π.
- k : Nombre d'intervalles d'intégration. Plus k est grand, plus la valeur calculée pour l'intégrale est «exacte», i.e., près du véritable résultat.

Modifiez ce programme de façon à réduire le plus possible son temps d'exécution — en vous assurant évidemment de toujours produire le même résultat.