Recherche Operationnelle

Vincent Gripon

École Nationale Supérieure d'Ingénieurs de Brest

1 avril 2011

À quoi sert un parcours ?

Objectif

Un parcours peut servir à plusieurs choses :

- Visiter l'ensemble des noeuds du graphes accessibles depuis un noeud de départ,
  Utile par exemple en compilation : existe-il des lignes du programme qui ne seront jamais utilisées ?
- Parcourir tous les chemins élémentaires qui partent d'un noeud de départ,
  Utile par exemple en optimisation : quel est le chemin le moins couteux qui parte de ce noeud ?
- Parcourir l'ensemble des chemins qui partent d'un noeud donné.
  En pratique n'est pas utilisé, sa seule différence avec le parcours au dessus étant qu'il peut être infini s'il y a un cycle. Utilisé dans une certaine mesure en vérification.
- ...

Parcours en profondeur de tous les chemins

```
Parcours_en_profondeur n:
  Pour tous les successeurs s de n
    Parcours_en_profondeur s
  Fin pour
Fin

Parcours_en_profondeur_depuis n
```


Parcours en profondeur de tous les noeuds

```
Noeuds_visités = Ensemble_vide

Parcours_en_profondeur n:
  Rajouter n à Noeuds_visités
  Pour tous les successeurs s de n
    Si s pas dans Noeuds_visités
    Alors
      Parcours_en_profondeur s
    Fin si
  Fin pour
Fin

Parcours_en_profondeur_depuis n
```

Parcours en profondeur de tous les chemins élémentaires

```
Parcours_en_profondeur n Noeuds_visités:
  Rajouter n Noeuds_visités
  Pour tous les successeurs s de n
    Si s pas dans Noeuds_visités
    Alors
      Noeuds_visités_s = copie Noeuds_visités
      Parcours_en_profondeur s Noeuds_visités_s
    Fin si
  Fin pour
Fin

Parcours_en_profondeur_depuis n Noeuds_visités
```


Remarques

Sur le fonctionement du parcours

- Philosophie : "Si je vois un nouveau noeud, je le visite aussitôt."
- Si plusieurs noeuds arrivent... ? Je rajoute tous mes noeuds en tête de ma structure et je les visite dans cet ordre.
- Structure de donnée *LIFO* (Pile).

Conséquences

- Les premiers noeuds, voisins du noeud de départ, sont visités après un grand nombre d'itérations.
- Le premier chemin choisi détermine tout l'intérêt de l'algorithme.

Parcours en profondeur des chemins élémentaires revisité

```
Noeuds_a_visiter = pile ()
Empiler (n,Noeuds_visités) Noeuds_a_visiter
Parcours_en_profondeur:
  Si Noeuds_a_visiter non vide
  Alors
    n,Noeuds_visités = Dépiler Noeuds_a_visiter
    Rajouter n Noeuds_visités
    Pour tous les successeurs s de n
      Si s pas dans Noeuds_visités
      Alors
        Noeuds_visités_s = copie Noeuds_visités
        Empiler (s,Noeuds_visités_s) dans Noeuds_visités
      Fin si
    Fin pour
    Parcours_en_profondeur ()
  Fin si
```


Parcours en largeur

Une autre philosophie

- Alternative évidente : utiliser une structure *FIFO* (file),
- Les noeuds ajoutés sont visités dans le même ordre,
- C'est le parcours en largeur.

Différences

- Les noeuds les plus "proches" sont les premiers visités,
- L'influence du choix des noeuds est beaucoup moins cruciale.

Parcours en largeur des chemins élémentaires

```
Noeuds_a_visiter = file ()
Enfiler (n,Noeuds_visités) Noeuds_a_visiter
Parcours_en_largeur:
  Si Noeuds_a_visiter non vide
  Alors
    n,Noeuds_visités = Défiler Noeuds_a_visiter
    Rajouter n Noeuds_visités
    Pour tous les successeurs s de n
      Si s pas dans Noeuds_visités
      Alors
        Noeuds_visités_s = copie Noeuds_visités
        Enfiler (s,Noeuds_visités_s) dans Noeuds_visités
      Fin si
    Fin pour
    Parcours_en_largeur ()
  Fin si
```


Plus court chemin

Problèmatiques

- Trouver le plus court chemin pour sortir d'un labyrinthe (1-to-1),
- Chercher l'ensemble des points atteignables à partir d'un sommet et leur distance minimale (1-to-*),
- Dresser une carte des distances minimales entre tous points d'un graphe (*-to-*).

Deux algorithmes principaux

- Roy Warshall : *-to-*,
- Dijkstra : 1-to-*.