

2

META-LANGAGE DE MANIPULATION DE GRAPHS

Sommaire

2	META-LANGAGE DE MANIPULATION DE GRAPHS	1
2.1	Introduction	2
2.2	Définition de la donnée	3
2.2.1	Ensembles constituants du graphe.....	3
2.2.2	Graphe	3
2.2.3	Eléments du graphe.....	4
2.3	Modification de graphe	5
2.3.1	Affectations	5
2.3.2	Ajout / suppression	5
2.3.3	Test d'existence	6
2.4	Parcours d'un graphe	7
2.4.1	Traitement sur la base des sommets.....	7
2.4.2	Traitement sur la base des arcs / arêtes	7
2.5	Traitement de tous les successeurs ou prédécesseurs d'un sommet.....	8

2.1 Introduction

Tous les langages de programmation ne définissent pas des instructions permettant directement de définir puis manipuler des informations représentées sous forme de graphe.

Le but de cette section est de définir un « méta-langage » spécifique que l'on utilisera en extension d'un langage algorithmique classique.

Nous décrivons donc :

- le moyen de définir une donnée informatique de type graphe ;
- les instructions permettant de les manipuler.

Le méta-langage contient a priori les éléments nécessaires et utilisés dans ce cours. Il pourra être enrichi en cas de besoin.

La description du meta-langage se fait pour partie au moyen d'exemples.

2.2 Définition de la donnée

2.2.1 Ensembles constituants du graphe

Sommets :

```
sommets S1  num|txt|...
sommets S2
```

S1 et S2 sont des ensembles de sommets.

Le libellé des sommets est de type numérique ('num') ou alphanumérique ('txt'). On peut aussi utiliser n'importe quel autre type de donnée pour définir le type des libellés. Si cette information n'est pas nécessaire, elle pourra être omise.

Arcs (graphe orienté) :

```
arcs A1  num
arcs A2
```

A1 et A2 sont des ensembles d'arcs.

En cas de graphe valué (A1), les valeurs associées sont de type numériques ('num') ou alphanumériques (on mettrait alors 'txt'). On peut aussi utiliser n'importe quel autre type de donnée pour définir le type des libellés ('int', 'char', ...).

A2 est non valué.

Arêtes (graphe non orienté) :

```
arêtes A1  num
arêtes A2
```

Semblable à la définition d'arcs.

2.2.2 Graphe

Pour définir un graphe dans son ensemble :

```
graphe <id_graphe> = {
    sommets <id_S>
    arcs    <id_A>
}

graphe <id_graphe> = {
    sommets <id_S>
    arêtes  <id_A>
}
```

On pourra faire référence :

au graphe lui-même

<id_graphe>

à l'ensemble des sommets

<id_graphe>.<id_S>

à l'ensemble des arcs ou arêtes

<id_graphe>.<id_A>

Il est bien sûr possible de spécifier les types des libellés des sommets, et/ou les types des valeurs associées aux arcs ou arêtes (graphes valués).

2.2.3 Eléments du graphe

On aura aussi besoin de faire référence à un sommet, un arc ou une arête dans un graphe. On utilisera pour cela les syntaxes suivantes :

```
sommet <id_sommet> de <id_G>.<id_S>
arête  <id_arête>  de <id_G>.<id_A>
arc    <id_arc>    de <id_G>.<id_A>
```

ou

```
sommet <id_sommet> de <id_G>
arête  <id_arête>  de <id_G>
arc    <id_arc>    de <id_G>
```

Si nécessaire, on pourra également faire référence aux sommets d'un arc ou d'une arête :

```
arête  <id_arête> = (s1,s2) de <id_G>.<id_A>
arc    <id_arc>   = (s1,s2) de <id_G>.<id_A>
```

ou

```
arête  <id_arête> = (s1,s2) de <id_G>
arc    <id_arc>   = (s1,s2) de <id_G>
```

2.3 Modification de graphe

Hypothèses (définitions) :

```

graphe G = { sommets S ...
              arcs A      ... }
graphe G1 = { sommets S ...
              arcs A      ... }
graphe G2 = { sommets S ...
              arcs A      ... }
sommet s    // un sommet
arc a       // un arc
sommets So  // un ensemble de sommets
arcs Ar     // un ensemble d'arcs

```

2.3.1 Affectations

On pourra initialiser l'ensemble d'un graphe à partir d'un autre de structure similaire :

```
G1 <- G2
```

On pourra initialiser l'ensemble des sommets uniquement :

```
G1.S <- So
```

Cette affectation est supposée initialiser l'ensemble des arcs / arêtes à vide.

Initialisation des arcs (ou arêtes) uniquement :

```
G1.A <- Ar
```

Une affectation particulière est l'initialisation d'un graphe vide :

```
G1 <- vide
```

Affectation d'une variable arc (ou arête) :

```

a <- ( s1 , s2 )      avec s1 et s2 deux sommets
a <- ( s1 , s2 , v )  en cas de graphe valué

```

2.3.2 Ajout / suppression

D'un sommet :

```

G1 <- G1 + s
G1 <- G1 - s

```

La suppression d'un sommet entraîne la suppression de tous les arcs / arêtes dont il est une des extrémités.

D'un arc (ou d'une arête) :

```

G1 <- G1 + a
G1 <- G1 + (s1,s2)      avec s1 et s2 deux sommets
G1 <- G1 + (s1,s2,v)    en cas de graphe valué

```

Idem pour la suppression avec '-'.

2.3.3 Test d'existence

Les expressions rendent pour résultat 'vrai' ou 'faux'.

Sommet :

`x est dans G.S`

Cette expression peut également être utilisées avec n'importe quelle variable représentant un ensemble de sommet, par exemple les ensembles 'ouverts' et 'fermés' de certains algorithmes.

Arc :

`a est dans G.A`

2.4 Parcours d'un graphe

2.4.1 Traitement sur la base des sommets

2.4.1.1 *Traitement itératif sur la base d'un seul sommet*

Traitement de tous les sommets :

```
pour tout sommet s de G.S
faire
    ...traitement...
fait
```

Traitement des sommets satisfaisant une certaine condition :

```
pour tout sommet s de G.S tel que degré intérieur(s) = 0
faire
    ...traitement...
fait
```

2.4.2 Traitement sur la base des arcs / arêtes

2.4.2.1 *Traitement relatif à un arc / une arête*

```
pour tout arc a de G.A
faire
    ...traitement...
fait

pour toute arête a de G.A
faire
    ...traitement...
fait
```

ou, si on a besoin de référencer explicitement les extrémités de a :

```
pour tout arc a = ( s1 , s2 ) de G.A
faire
    ...traitement...
fait

pour toute arête a = ( s1 , s2 ) de G.A
faire
    ...traitement...
fait
```

Exemples avec des conditions :

```
pour tout arc a = ( s1 , s2 ) de G.A tel que s1 <> s2
faire
    ...traitement...
fait

pour toute arête a = ( s1 , s2 ) de G.A tel que degré extérieur(s2) = 0
faire
    ...traitement...
fait
```

2.5 Traitement de tous les successeurs ou prédécesseurs d'un sommet

```
pour tout successeur s de x dans G  
faire  
    ...traitement...  
fait
```

```
pour tout prédécesseur s de x dans G  
faire  
    ...traitement...
```