INFO0054 - Programmation fonctionnelle

Répétition 6

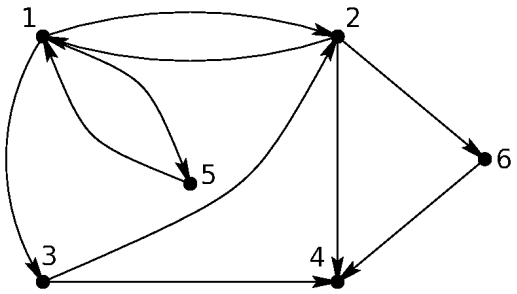
Jean-Michel Begon

31 mars 2020

Graphes

Nous convenons de représenter un *graphe orienté* comportant n nœuds par une liste de n listes ; la liste correspondant au nœud x a pour premier élément x et pour éléments suivants les successeurs (immédiats) de x.

Par exemple le graphe



sera représenté par la liste

```
'((1 2 3 5) (2 1 4 6) (3 2 4) (4) (5 1) (6 4))
```

Exercice 1.
Écrire une fonction `adj` qui prend comme argument la représentation d'un graphe orienté `g`, ainsi qu'un nœud `n` de `g`, et qui retourne la liste des nœuds adjacents de `n`.

Exercice 2.
Écrire le prédicat `(path? g n1 n2)` prenant deux nœuds `n1` et `n2` et la représentation `g` d'un graphe orienté en arguments et retournant `#t` s'il existe un chemin de $n1$ vers $n2$ dans g et `#f` sinon.

Exercice 3.
Écrire une fonction `inv` qui, à partir d'un graphe orienté, construit le graphe retourné correspondant.

Listes

Exercice 4.
Ecrire une fonction `longest-inc` retournant la plus longue sous-liste croissante de la liste d'entiers donnée en argument.

Exercice 5.
Écrire une fonction `sublists` prenant comme argument une liste `l` et retournant la liste des sous-listes de `l`. (Une sous-liste de `l` est une liste de 0, 1 ou plusieurs éléments consécutifs de `l`.)

Exercice 6.
Écrire une fonction qui renvoie la liste des permutations circulaires d'une liste donnée.

Exercice 7.
Ecrire une fonction `div-ls` qui retourne la liste des diviseurs d'un entier strictement positif.