IN 102 - Cours 11

ENSTA ParisTech

1 Graphes

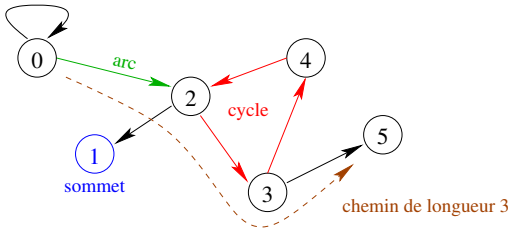
1.1 Structure de graphe

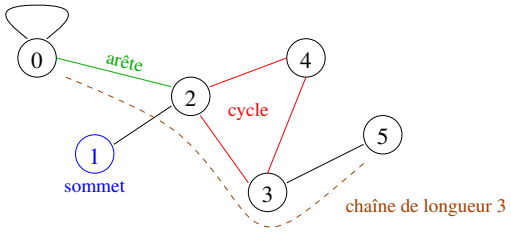
Un graphe est la donnée d'un ensemble de sommets connectés par des arcs.

Contrairement à un arbre, dans un graphe un sommet (nœud) peut avoir un nombre quelconque de parents. Il peut donc y avoir des cycles, des ≪raccourcis ≫.

⇒ Les arbres sont des cas particuliers (plus ≪simples ≫) de graphes.

Les arcs d'un graphe peuvent ou non être orientés. Dans le cas non orienté on parle souvent d'arête.





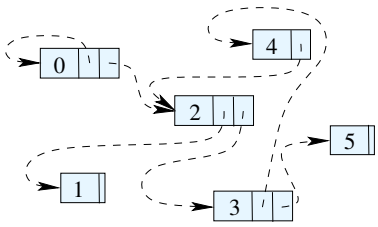
Un chemin est suite d'arcs consécutifs. Dans le cas non orienté, une chaîne est une suite d'arêtes consécutives.

La longueur d'un chemin (ou chaîne) est le nombre d'arcs (ou arêtes) sur ce chemin.

Un cycle est chemin qui ≪boucle ≫.

1.2 Implémentation de graphes

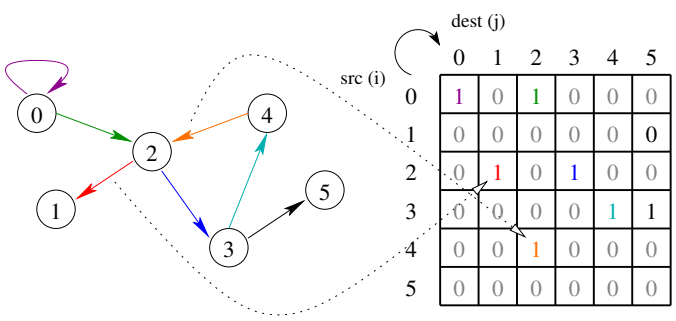
En C on peut représenter un graphe par ses sommets, chacun mémorisant la liste chaînée des pointeurs sur ses voisins.




```
struct vertex_list_t {
  struct vertex_t *vertex ;
  struct vertex_list_t *next ;
};

struct vertex_t {
  int data ;
  struct vertex_list_t *neighbours ;
};
```

On peut représenter un graphe par sa matrice d'adjacence : une matrice booléenne carrée disant s'il y a un arc entre les sommets 2 à 2. $M_{ij} = 1$ s'il existe un arc $i \to j$, sinon $M_{ij} = 0$.



La matrice d'adjacence a une complexité spatiale en n^2 pour un graphe à n sommets, ce qui est très élevé.

Certains algorithmes sont beaucoup plus efficaces sur une représentation par matrice d'adjacence, d'où son intérêt malgré sa complexité spatiale.

1.3 Parcours de graphes

Généralement un graphe est construit à partir de données et on le parcourt pour en extraire des propriétés sur ces données.

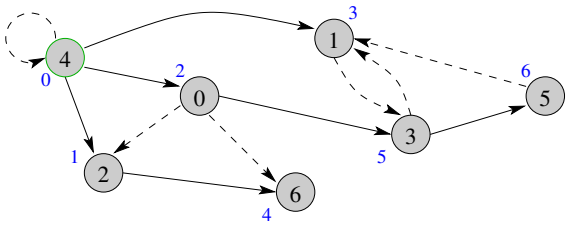
Parmi les parcours possibles, la classe des recouvrements regroupe ceux qui extraient un arbre contenant tous les sommets du graphe.

Quelque soit le parcours, il faut ne pas visiter plusieurs fois le même sommet, ne pas boucler.
⇒ On marque les sommets déjà visités et on ne re-visite pas ceux marqués.

Le parcours en largeur d'abord est similaire à celui pour les arbres : on parcours tous les voisins de distance d avant ceux de distance $d+1$.

Le parcours en largeur d'abord s'appuie sur une structure de file.

```
ParcoursLargeur (r la racine du graphe)
  Marquer r ;
  Enfiler r dans la file F ;
  Tant que F n'est pas vide {
    s = element en tete de F ;
    Traiter (s) ;
    Pour chaque voisin v du sommet s {
      Si v n'est pas marque alors {
        Marquer v ;
        Enfiler v dans la file F ;
      }
    }
  }
```



Le parcours en profondeur d'abord est similaire à celui pour les arbres : on parcours récursivement tous les voisins d'un sommet.

```
DFS (sommet s)
{
  Marquer s ;
  (Pre-traiter s ;)
  Pour chaque voisin non marque v du
      sommet s
    DFS (v) ;
  (Post-traiter s ;)
  }
}
```