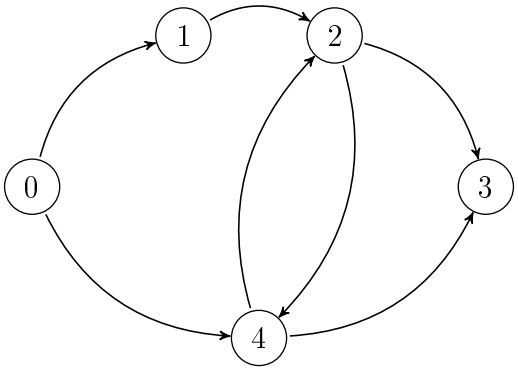
TD 4

Représentation d'un graphe

1 Représentation par matrice d'adjacence

La matrice d'adjacence est une matrice booléenne (carrée et d'ordre n) telle que $m.(i).(j)$ vaut true si et seulement si on a un arc du sommet i au sommet j.



Nous utiliserons alors le type suivant :

```
type graphe_1 = bool array array ;;
```

1. Définir l'exemple avec ce type.
2. Écrire deux fonctions `taille` et `nb_arcs` comptant respectivement le nombre de sommets et le nombre d'arcs d'un graphe.

   ```
   taille : graphe_1 -> int
   ```

   ```
   nb_arc : graphe -> int
   ```
3. Écrire les fonctions `ajout_arc` et `supprime_arc` :
   - La fonction `ajout_arc` prend en argument un graphe et les sommets i et j, et ajoute l'arc (i, j).

     ```
     ajout_arc : graphe_1 -> int -> int -> unit
     ```
   - La fonction `supprime_arc` prend en argument un graphe et les sommets i et j, et supprime l'arc (i, j).

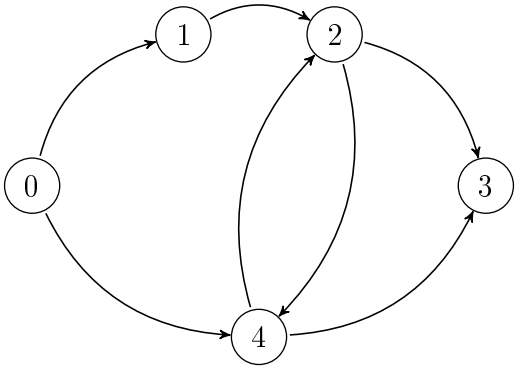
     ```
     supprime_arc : graphe_1 -> int -> int -> unit
     ```
4. Écrire la fonction `supprime_sommet` qui rend inaccessible le sommet i du graphe.

   ```
   supprime_sommet : graphe_1 -> int -> unit
   ```
5. Écrire la fonction `desoriente` qui transforme un graphe orienté en graphe non orienté.

   ```
   desoriente : graphe_1  -> unit
   ```

2 Représentation par listes d'adjacence

Les listes d'adjacence sont représentées par un tableau v de listes d'entiers tel que $v.(s)$ contient la liste des voisins du sommet s.



Nous utiliserons alors le type suivant :

```
type graphe_2 = int list array ;;
```

1. Définir l'exemple avec ce type.
2. Écrire deux fonctions `taille` et `nb_arcs` comptant respectivement le nombre de sommets et le nombre d'arcs d'un graphe.

   `taille : graphe_2 -> int`

   `nb_arc : graphe_2 -> int`
3. Écrire les fonctions `ajout_arc` et `supprime_arc` :
   - La fonction `ajout_arc` prend en argument un graphe et les sommets i et j, et ajoute l'arc (i, j).

     `ajout_arc : graphe_2 -> int -> int -> unit`
   - La fonction `supprime_arc` prend en argument un graphe et les sommets i et j, et supprime l'arc (i, j).

     `supprime_arc : graphe_2 -> int -> int -> unit`
4. Écrire la fonction `supprime_sommet` qui rend inaccessible le sommet i du graphe.

   `supprime_sommet : graphe_2 -> int -> unit`
5. Écrire la fonction `desoriente` qui transforme un graphe orienté en graphe non orienté.

   `desoriente : graphe_2 -> unit`
6. Écrire les fonctions `mat_of_liste` et `liste_of_mat` :
   - La fonction `mat_of_liste` renvoie la matrice d'adjacence du graphe dont on connait les listes d'adjacence.

     `mat_of_liste : int list array -> bool array array`
   - La fonction `liste_of_mat` renvoie les listes d'adjacence du graphe dont on connait la matrice d'adjacence.

     `liste_of_mat : bool array array -> int list array`