L'objectif de ce sujet de TP est de dessiner des graphes avec la commande `dot`. Nous en profiterons pour faire des petites révisions sur les entrées/sorties.

1 Le langage `DOT` de la suite `graphviz`

La suite de logiciels `graphviz` a pour but la représentation graphique, dans divers formats (`pdf`, `png`, ...), de graphes orientés ou non en utilisant différents algorithmes.

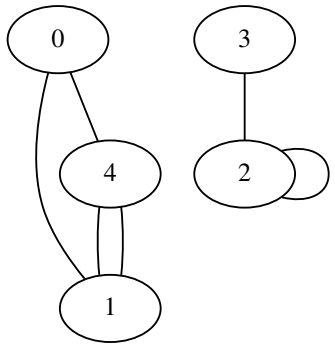
Pour notre part nous utiliserons la commande `dot`. Cette commande a la syntaxe suivante :

```
dot -Text fichier_source.dot -o fichier_dest.ext
```

où `ext` est l'extension associée au format voulu (par exemple `pdf`) et `fichier_source.dot` est un fichier texte écrit dans le langage `DOT` de descriptions de graphes. Voyons tout de suite un exemple de tel fichier et la sortie correspondante.

```
graph {
    0;
    1;
    2;
    3;
    4;

    0 -- 1;
    2 -- 2;
    3 -- 2;
    4 -- 1;
    0 -- 4;
    4 -- 1;
}
```



Dans cet exemple, le fichier `DOT` s'appelait `exemple1.dot` et le fichier `pdf` a été généré avec la commande

```
dot -Tpdf exemple1.dot -o mon_graphe.pdf
```

puis visualisé avec la commande

```
evince mon_graphe.pdf &
```

(le final permet de mettre la commande en arrière-plan et donc de garder la main sur le terminal).

Pour se contenter d'afficher le graphe dans une fenêtre, sans écrire de fichier, on peut entrer la commande suivante :

```
dot -Tx11 exemple1.dot
```

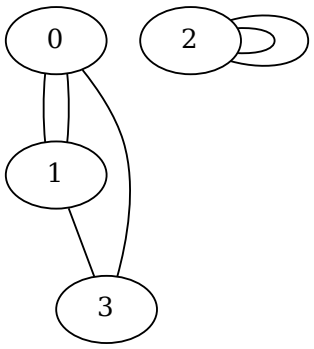


FIGURE 1 – Graphe à écrire dans la question 1.

Dans le code `DOT`, le mot-clé `graph` sert à signaler qu'on a un graphe *non-orienté*. Entre les accolades, le graphe est décrit [1]. Les premières lignes, ne comportant qu'une seule étiquette, servent à définir les sommets du graphe, et les lignes suivantes (de la forme `i -- j;`) servent à décrire les arêtes du graphe.
Question 1: Décrire au format `DOT` le graphe représenté à la figure 1, puis visualiser avec une commande `dot` le graphe correspondant.

--- * ---

2 Génération automatique de fichiers `dot`, 1ère partie

Rappels sur les entrées/sorties : le bout de code suivant ouvre un fichier en écriture, vérifie que cette ouverture s'est bien passée, écrit des caractères dans ce fichier et finalement le ferme (si le fichier existait déjà, il commence par être vidé).

```c
#include <stdio.h> /* fopen, fclose, fprintf, perror, FILE */
#include <stdlib.h> /* EXIT_SUCCESS, EXIT_FAILURE */

int main() {
    FILE *f;
    int i;
    char nom_fichier[] = "bouillie.txt";
    f = fopen(nom_fichier, "w");
    if ( f == NULL ) {
        perror("fopen"); /* écrit la dernière erreur rencontrée par fopen */
        return EXIT_FAILURE;
    }
    fprintf(f, "Je fais de la bouillie pour mes petits cochons :\n");
    for (i = 1; i < 10; i++)
        fprintf(f, "pour %d,\n", i);
    fprintf(f, "boeuf !!\n");
    fclose(f);
    return EXIT_SUCCESS;
}
```

Question 2: Écrire un simple programme en C (pour l'instant *sans* utiliser la bibliothèque `graphe-1`) qui décrit dans un fichier au format `DOT` le graphe complet d'ordre 10.

1. On peut être beaucoup plus concis en langage `DOT` mais le format très rigide que nous avons choisi est facilement automatisable.

Dessiner ce graphe en utilisant une commande `dot`.

--- * ---

3 Génération automatique de fichiers DOT, 2ème partie

Question 3: *Dans la bibliothèque* ***graphe-1***, déclarer et définir la fonction

```
int graphe_ecrire_dot(graphe *g, char *nom_fichier)
```

qui écrit dans le fichier `nom_fichier` le graphe pointé par `g` au format `DOT`.

La valeur de retour de cette fonction est :

0 si l'exécution n'a rencontré aucun problème ;

-1 si le fichier `nom_fichier` n'a pu être ouvert en écriture.

--- * ---

Question 4: Écrire un fichier `main-dessin.c` utilisant les fonctions `graphe_aleatoire` et `graphe_ecrire_dot` pour écrire un fichier `DOT` représentant un graphe aléatoire à 9 sommets. Compiler et exécuter ce programme, puis représenter graphiquement ce graphe avec une commande `dot`. Vous pouvez aussi vous amuser à modifier la constante `GRAPHE_ORDRE_MAX` et dessiner un gros graphe aléatoire (à partir de combien de sommets le dessin du graphe prend-il du temps ?)

--- * ---

4 Lecture d'un graphe au format DOT

L'écriture de fichiers au format `DOT` permet également de *sauvegarder* des graphes, encore faut-il pouvoir également les *charger* en mémoire.

Question 5: Écrire une fonction

```
int graphe_charger_dot(graphe *g, char *nom_fichier)
```

qui, étant donné un fichier nommé `nom_fichier` écrit au format `DOT` rigide décrit dans la première partie, permet d'initialiser le graphe pointé par `g` au graphe décrit dans le fichier `nom_fichier`.

La valeur de retour de cette fonction est :

0 si l'exécution n'a rencontrée aucun problème ;

-1 si le fichier `nom_fichier` n'est pas lisible en lecture ;

-2 si le nombre de sommets du graphe décrit dans `nom_fichier` est trop élevé (plus grand que `GRAPHE_ORDRE_MAX`).

Question 6: BONUS : faire en sorte que la fonction `graphe_charger_dot` retourne la valeur -3 si le fichier `nom_fichier` n'est pas conforme à ce qui est attendu (problèmes de syntaxe, sommets mal étiquetés, ...).

--- * ---