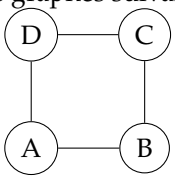
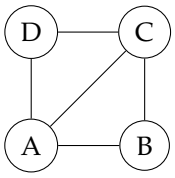
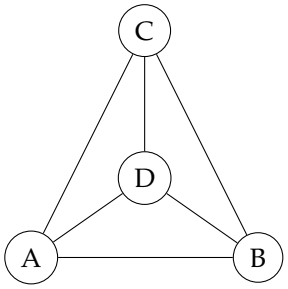
TD : Arbres couvrants

Exercice 1

Déterminer l'ensemble des arbres couvrants de chacun des graphes suivants.

1.


2.


3.


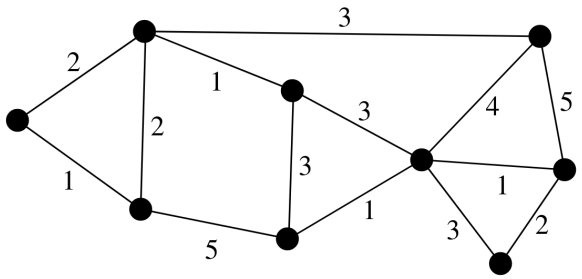
Exercice 2

Soit $G = (S, A)$ un graphe non orienté sans sommet isolé. On appelle forêt couvrante tout sous-graphe sans cycle de G (mais pas nécessairement connexe) qui est couvrant, c'est-à-dire tel que tout sommet de G est touché par une arête au moins. On dit que la forêt couvrante est maximale si on ne peut lui ajouter d'arête sans créer de cycle.

1. Donner un exemple de forêt couvrante et de forêt couvrante maximale.
2. Que dire d'une forêt couvrante maximale quand G est connexe ? Justifier.
3. Montrer que le nombre d'arêtes d'une forêt couvrante maximale est $\text{Card}(S) - k$ où k est le nombre de composantes connexes de G.

Exercice 3

Appliquer l'algorithme de Kruskal sur le graphe suivant :



Exercice 4

Soit un graphe pondéré $G = (S, A)$, connexe et non-orienté. Un algorithme alternatif à l'algorithme de Kruskal est l'algorithme de Prim (hors programme). Il fonctionne de manière glotounne :

- Choisir un sommet initial s_0 et poser $S' = \{s_0\}$ et $A' = \varnothing$.
- Tant que $S' \neq S$ Faire :
  - Choisir une arête (u, v) de poids minimal parmi les arêtes *sortantes de* S', c'est-à-dire avec $u \in S'$ et $v \notin S'$.
  - $S' \leftarrow S' \cup \{v\}$
  - $A' \leftarrow A' \cup \{(u, v)\}$
- Sortie : $G' = (S', A')$ est un arbre couvrant optimal pour G.

1. Appliquer l'algorithme de Prim sur l'exemple de l'exercice précédent.
2. Quelle structure de donnée est utile pour l'implémentation de l'algorithme de Prim ?
3. Écrire en pseudo-code l'algorithme de Prim en util-

isant les opérations de cette structure.
4. Estimer la complexité obtenue.

Exercice 5

On considère un damier de taille 10×10 où les cases ont été aléatoirement choisies blanc ou noir. On pourra le représenter en OCaml par une matrice :

```
type couleur = Blanc | Noir;;
type damier = couleur array array;;
```

Une pièce sur le damier peut se déplacer d'une case vers le bas, le haut, la gauche ou la droite à condition que la case d'arrivée soit de la même couleur que la case de départ.

1. En utilisant la structure de donnée Union-Find, écrire un programme en OCaml qui permet de déterminer si une pièce peut se déplacer d'une case à une autre à l'aide d'un nombre quelconque de déplacements.
2. Quel autre algorithme aurait-on pu utliser ?
3. Comment adapter la structure Union-Find pour obtenir le nombre de composantes connexes du damier (connexe au sens des déplacements possibles de pièce) ?

⋆Exercice 6

Soit un graphe pondéré $G = (S, A)$, pondéré, connexe et non-orienté. Peut-il exister un arbre couvrant de poids minimal qui ne puisse être obtenu par l'algorithme de Kruskal ? Autrement dit, est-ce que l'algorithme de Kruskal permet d'obtenir tous les arbres couvrants optimaux ?