Chapitre 2

Corrigé des exercices

• Combinatoire des graphes

Exercice 1

a) Soit G = (V, E) un graphe non orienté simple. Notons V_1 l'ensemble des sommets de degré pair et V_2 l'ensemble des sommets de degré impair. Nous savons que $\sum_{v\in V} \deg(v) = 2|E|$, donc :

$$\sum_{v\in V_2} \deg(v) = 2|E| - \sum_{v\in V_1} \deg(v).$$

De cette égalité il résulte que $\sum_{v\in V_2} \deg(v)$ est un entier pair, ce qui impose à $|V_2|$ d'être pair.

b) Si aucune personne n'a d'ami dans cette assemblée, ils ont tous le même nombre d'amis, à savoir 0.
On suppose maintenant qu'il existe au moins un couple d'amis dans l'assemblée, et on considère le graphe G = (V, E) où V est l'ensemble des personnes *qui ont au moins un ami* et où E est défini par :

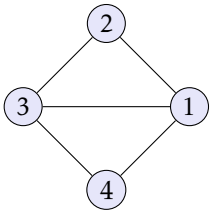
$$(a, b) \in E \iff a \text{ et } b \text{ sont amis.}$$

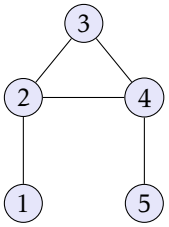
G est un graphe non orienté simple d'ordre k, et chacun des k sommets a un degré compris entre 1 et $k-1$; d'après le principe des tiroirs il existe au moins deux sommets de même degré.

c) Considérons un graphe G = (V, E) dont les sommets sont les ordinateurs et les arêtes les liaisons qui les relient. Si chaque sommet est de degré 3, la formule $\sum_{v\in V} \deg(v) = 2|E|$ s'écrit : $3 \times 15 = 2|E|$, ce qui ne se peut.

Exercice 2

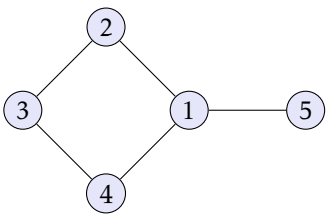
a) Les suites (3, 3, 2, 2) et (3, 3, 2, 1, 1) sont graphiques, comme le prouvent les deux graphes ci-dessous :

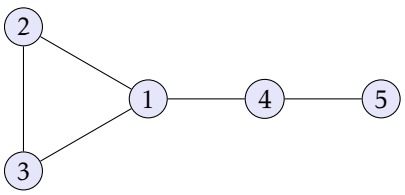




En revanche, la suite (3, 3, 1, 1) ne peut être graphique. En effet, dans un graphe d'ordre 4 ayant deux sommets de degré 3, autrement dit deux sommets reliés à chacun des trois autres, les deux derniers sommets sont au moins de degré 2.

b) Les deux graphes suivants sont distincts bien que correspondants tous deux à la suite (3, 2, 2, 2, 1) :





c) L'implication $(ii) \Longrightarrow (i)$ est claire : s'il existe un graphe dont les sommets ont pour degrés la suite $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n)$, il suffit de rajouter un sommet et de le relier aux d_1 premiers sommets du graphe pour obtenir un nouveau graphe correspondant à la suite :

$$(d_1, d_2, d_3, \ldots, d_{d_1+1}, d_{d_1+2}, d_{d_1+3}, \ldots, d_n) = (d_1, d_2, \ldots, d_n).$$

Réciproquement, si $(d_1, d_2, \dots, d_n)$ est graphique, nous allons montrer qu'il existe un graphe G = (V, E) tel que V = $(v_1, \dots, v_n)$, $\deg(v_i) = d_i$ et *tel que v_1 soit adjacent aux sommets $v_2, v_3, \dots, v_{d_1+1}$*. Une fois ceci prouvé, il suffira de supprimer v_1 ainsi que les arêtes qui le relient au reste du graphe pour en déduire que la suite $(d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \dots, d_n)$ est graphique.

Nous allons prouver l'existence de G par l'absurde en considérant un graphe G pour lequel v_1 est relié au plus grand nombre possible de sommets parmi $v_2, \dots, v_{d_1+1}$, mais pas à tous.

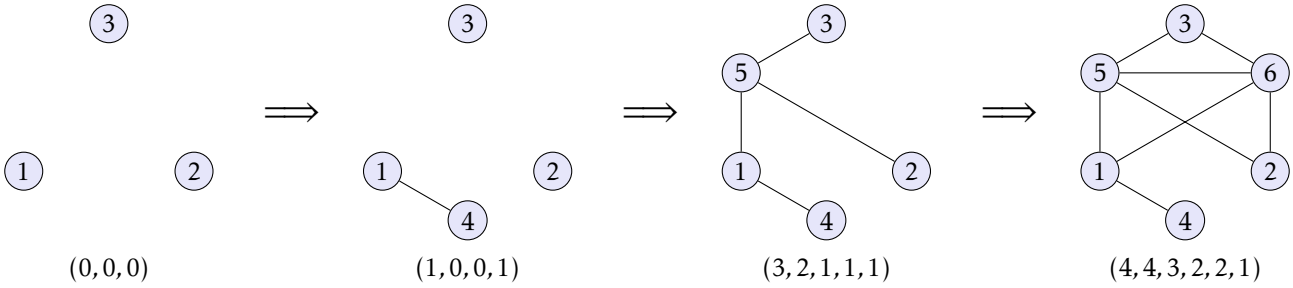
Dans ces conditions, il existe $i \in [\![2, d_1 + 1]\!]$ tel que $(v_1, v_i) \notin$ E et $j \in [\![d_1 + 2, n]\!]$ tel que $(v_1, v_j) \in$ E. Puisque $i \leqslant j$ on a $\deg(v_i) \geqslant \deg(v_j)$. Traitons alors deux cas.

- Si $\deg(v_i) = \deg(v_j)$, tous les sommets d'indices dans $[\![i, j]\!]$ ont même degré, et il suffit d'inverser la numérotation des sommets v_i et v_j pour aboutir à une contradiction.
- Si $\deg(v_i) > \deg(v_j)$, il existe un sommet v_k tel que $(v_i, v_k) \in$ E mais $(v_j, v_k) \notin$ E. En particulier, notons que $v_k \neq v_1$. Supprimons alors les arêtes (v_1, v_j) et (v_i, v_k) et ajoutons les arêtes (v_1, v_i) et (v_j, v_k). La suite des degrés reste inchangée, mais maintenant v_1 est relié à un sommet de plus parmi $v_2, v_3, \dots, v_{d_1+1}$ ce qui contredit le caractère maximal de G et achève la preuve du théorème.

d) Prouvons tout d'abord que la suite (4, 4, 3, 2, 2, 1) est graphique en appliquant le théorème :

$$\begin{aligned}(4,4,3,2,2,1) \text{ est graphique} &\iff (3,2,1,1,1) \text{ est graphique} \iff (1,0,0,1) = (1,1,0,0) \text{ est graphique} \\ &\iff (0,0,0) \text{ est graphique.}\end{aligned}$$

La suite (0, 0, 0) est bien graphique donc la construction demandée est possible, et la preuve du théorème nous donne un moyen de construire une solution à partir des équivalences ci-dessus :



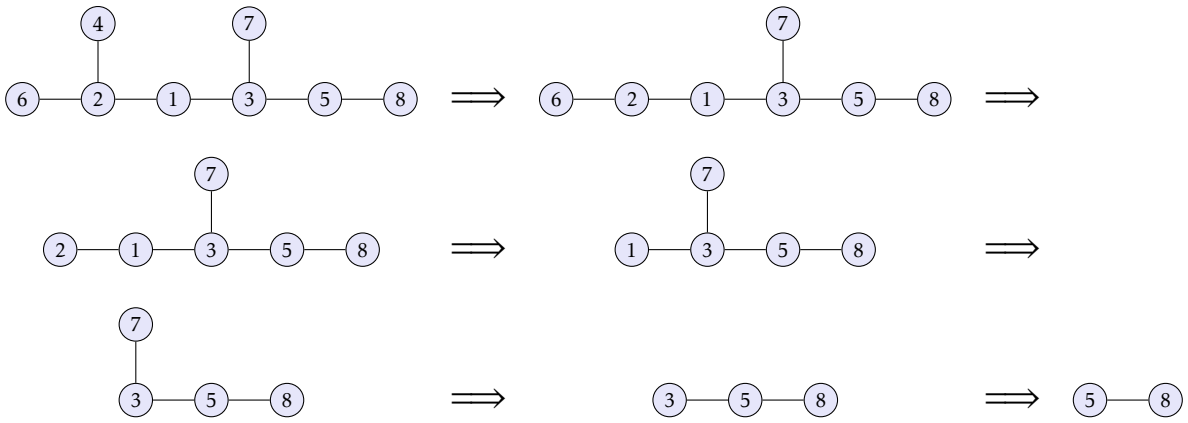
Si les deux sommets de degré 2 étaient voisins, on obtiendrait en retirant l'arête qui les relie un graphe dont la suite des degrés serait (4, 4, 3, 1, 1, 1). Or si on applique le théorème à cette suite on obtient :

$$\begin{aligned}(4,4,3,1,1,1) \text{ est graphique} &\iff (3,2,0,0,1) = (3,2,1,0,0) \text{ est graphique} \\ &\iff (1,0,-1,0) = (1,0,0,-1) \text{ est graphique}\end{aligned}$$

ce qui est absurde.

Exercice 3 Notons que la formule $\sum_{v \in V} \deg(v) = 2|E| = 2n - 2$ montre que dans un arbre il existe au moins un sommet de degré 1, ce qui prouve que la terminaison de l'algorithme de Prüfer.

a) La suppression des différents sommets s'effectue de cette façon :



et conduit au codage : (2, 2, 1, 3, 3, 5).

b) Pour décoder un codage de Prüfer on propose l'algorithme suivant :

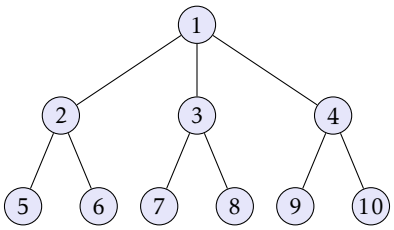
fonction DECODE(liste : L)
  $V \leftarrow [\![1, n]\!]$, $E \leftarrow \emptyset$
  $I \leftarrow V$
  tant que $|I| > 2$ **faire**
    $i \leftarrow \min\left\{j \in I \;\middle|\; j \notin L\right\}$
    $E \leftarrow E \cup \{(i, \text{tête}(L))\}$
    $I \leftarrow I \setminus \{i\}$
    $L \leftarrow \text{queue}(L)$
  $E \leftarrow E \cup \{(a, b)\}$ avec $I = \{a, b\}$
  renvoyer (V, E)

qui reconstitue la liste des arêtes qui ont été supprimées par l'algorithme de Prüfer.

c) Avec la suite $(2, 2, 1, 3, 3, 1, 4, 4)$, la suite L et les ensembles I et E évoluent de la façon suivante :

L	I	E
$(2,2,1,3,3,1,4,4)$	$\{1,2,3,4,5,6,7,8,9,10\}$	$\emptyset$
$(2,1,3,3,1,4,4)$	$\{1,2,3,4,6,7,8,9,10\}$	$\{(5,2)\}$
$(1,3,3,1,4,4)$	$\{1,2,3,4,7,8,9,10\}$	$\{(5,2),(6,2)\}$
$(3,3,1,4,4)$	$\{1,3,4,7,8,9,10\}$	$\{(5,2),(6,2),(2,1)\}$
$(3,1,4,4)$	$\{1,3,4,8,9,10\}$	$\{(5,2),(6,2),(2,1),(7,3)\}$
$(1,4,4)$	$\{1,3,4,9,10\}$	$\{(5,2),(6,2),(2,1),(7,3),(8,3)\}$
$(4,4)$	$\{1,4,9,10\}$	$\{(5,2),(6,2),(2,1),(7,3),(8,3),(3,1)\}$
(4)	$\{4,9,10\}$	$\{(5,2),(6,2),(2,1),(7,3),(8,3),(3,1),(1,4)\}$
$()$	$\{4,10\}$	$\{(5,2),(6,2),(2,1),(7,3),(8,3),(3,1),(1,4),(9,4)\}$
	$\emptyset$	$\{(5,2),(6,2),(2,1),(7,3),(8,3),(3,1),(1,4),(9,4),(10,4)\}$

ce qui conduit à l'arbre :



• Représentation d'un graphe

Exercice 4 Pour calculer les degrés sortants, il suffit de calculer la longueur de chacune des listes d'adjacence :

```
let deg_sortant = map_vect list_length ;;
```

Le calcul de la longueur d'une liste a un coût proportionnel à sa longueur donc le coût total de cette fonction est un $\Theta(n+p)$, où $n = |V|$ et $p = |E|$.
Pour calculer les degrés entrants, il faut parcourir chaque liste d'adjacence et pour chaque arête (a, b) rencontrée itérer la case d'indice b du tableau **deg_e** :

```
let deg_entrant g =
  let n = vect_length g in
  let deg_e = make_vect n 0 in
  let rec aux = function
    | []   -> ()
    | b::q -> deg_e.(b) <- deg_e.(b) + 1 ; aux q
  in do_vect aux g ;
  deg_e ;;
```

La fonction **aux** a un coût proportionnel à la longueur de la liste à laquelle elle s'applique donc le coût total est là encore un $\Theta(n+p)$.

Exercice 5 Calculer la transposée d'un graphe représenté par matrice d'adjacence est facile puisqu'il suffit de calculer la transposée de celle-ci ; le coût est à l'évidence un $\Theta(n^2)$.
Pour un graphe représenté par liste d'adjacence, il n'y a guère d'autre possibilité que de parcourir chacune des listes d'adjacence en ajoutant l'arc inverse au graphe transposé, qui se construit peu à peu.
Pour un graphe de type *graphe*, cela conduit à la définition suivante :

```
let transposee g =
  let n = vect_length g in
  let gt = make_vect n [] in
  let rec aux a = function
    | [] -> ()
    | b::q -> gt.(b) <- a::gt.(b) ; aux a q
  in
  for a = 0 to n-1 do aux a g.(a) done ;
  gt ;;
```

Sachant que l'insertion en tête de liste a un coût constant, le coût total de cette fonction est un $\Theta(n+p)$.

Exercice 6 Notons $(1,2,\dots,n)$ les sommets de $G=(V,E)$ et pour tout $i \in [\![1,n]\!]$ notons $\mathscr{V}(i)$ l'ensemble des voisins de i. Posons en outre $M=(m_{ij})_{1\leqslant i,j\leqslant n}$ et $M^p=(m_{ij}^{(p)})_{1\leqslant i,j\leqslant n}$.

a) Montrons par récurrence sur p que $m_{ij}^{(p)}$ est égal au nombre de chemins de longueur p qui mènent de i à j.

– C'est clair si $p=1$ puisque $m_{ij}^{(1)}=m_{ij}=\begin{cases}1 & \text{si } (i,j)\in E\\ 0 & \text{sinon}\end{cases}$.

– Si $p>1$, supposons le résultat acquis au rang $p-1$. On a : $m_{ij}^{(p)}=\sum_{k=1}^{n} m_{ik}m_{kj}^{(p-1)}=\sum_{k\in\mathscr{V}(i)} m_{kj}^{(p-1)}$.

Puisque $m_{kj}^{(p-1)}$ est égal au nombre de chemins de longueur $p-1$ reliant k à j, $m_{ij}^{(p)}$ est bien le nombre de chemins de longueur p reliant i à j.

En particulier, $m_{ii}^{(p)}$ est le nombre de cycles de longueur p passant par le sommet i. Ainsi, $\mathrm{tr}(M^{(p)})$ est égal au nombre de cycles de longueur p (chaque cycle étant compté autant de fois que le cardinal de son support).

b) En particulier, la matrice M^n n'est pas nulle si et seulement s'il existe un chemin de longueur n. Or dans un tel chemin il existe nécessairement un sommet atteint au moins deux fois, ce qui dénote l'existence d'un cycle au sein de ce chemin.
Réciproquement, à partir d'un cycle de longueur p il est facile de construire un chemin de longueur n en considérant la division euclidienne de n par p : $n=pq+r$ avec $0\leqslant r<p$. Il suffit de parcourir le cycle dans son entier q fois, puis de parcourir encore r sommets en son sein.
La méthode classique pour calculer le produit de deux matrices d'ordre n a un coût en $\Theta(n^3)$. L'algorithme d'exponentiation rapide appliqué au calcul de M^n aurait donc un coût en $\Theta(n^3\log n)$.

• Parcours dans un graphe

Exercice 7 Considérons un graphe pour lequel on a pu attribuer un rang à chaque sommet. S'il existait un circuit dans G, on pourrait considérer le sommet v de ce circuit qui possède le rang minimal ainsi que son prédécesseur u dans ce circuit et aboutir à une contradiction car on aurait à la fois $r(u)<r(v)$ et $r(v)\leqslant r(u)$. G ne possède donc pas de circuit.

Réciproquement, supposons que $G=(V,E)$ ne possède pas de circuit, et raisonnons par récurrence sur $|V|$.

– Le résultat est évident lorsque $|V|=1$.

– Si $|V|>1$, supposons le résultat acquis jusqu'au rang $|V|-1$.
Il existe au moins dans G un sommet sans prédécesseur (dans le cas contraire, on pourrait remonter successivement d'un sommet à un prédécesseur jusqu'à former un cycle). Attribuons le rang 1 à tous ces sommets, puis supprimons ces sommets et les arcs qui les concernent. On obtient un nouveau graphe $G'=(V',E')$, toujours sans circuit, avec $|V'|<|V|$. On peut lui appliquer l'hypothèse de récurrence en attribuant un rang à chacun de ces sommets, en faisant en sorte que pour tout $v\in V'$, $r(v)\geqslant 2$. On obtient ainsi une numérotation des sommets qui respecte la contrainte demandée.

L'algorithme d'attribution d'un rang à chaque sommet d'un graphe sans circuit s'écrit donc :

```
procedure RANG(graphe : G = (V, E))
    k ← 1
    S ← V
    tant que S ≠ ∅ faire
        P ← {v ∈ S | ∀u ∈ S, (u, v) ∉ E}
        pour tout v ∈ P faire
            r(v) ← k
        S ← S \ P
        k ← k + 1
```

Lorsque G est représenté par sa matrice d'adjacence, déterminer si un sommet n'a pas de prédécesseur a un coût linéaire et calculer P un coût quadratique, donc le coût total de cet algorithme est un $O(n^3)$, avec $n = |V|$.

Exercice 8

a) Notons N_n le graphe d'ordre n sans arêtes ; alors $\alpha(N_n) = n$.
Notons C_n le graphe complet d'ordre $n \geqslant 1$; alors $\alpha(C_n) = \max(\alpha(C_{n-1}), 1)$ et par une récurrence immédiate $\alpha(C_n) = 1$.
Notons L_n le chemin d'ordre $n \geqslant 2$ non ramifié ; alors $\alpha(L_n) = \max(\alpha(L_{n-1}), 1 + \alpha(L_{n-2}))$. On prouve dans ce cas par récurrence que $\alpha(L_n) = \left\lfloor \frac{n}{2} \right\rfloor$.

Notons U_n le graphe cyclique d'ordre $n \geqslant 3$; alors $\alpha(U_n) = \max(\alpha(L_{n-1}), 1 + \alpha(L_{n-3}))$ donc $\alpha(U_n) = \left\lfloor \frac{n-1}{2} \right\rfloor$.

b) On raisonne par récurrence sur l'ordre n du graphe G.

- Si $n = 1$ alors $E = \emptyset$ donc $\alpha(G) = 1$, ce qui est bien le nombre de stabilité de G.
- Si $n > 1$, on suppose le résultat acquis jusqu'au rang $n - 1$.
  Si $E = \emptyset$ alors $\alpha(G) = n$, ce qui correspond bien à son nombre de stabilité.
  Sinon, considérons $v \in V$ tel que $\deg v \geqslant 1$. Posons $G_1 = G \setminus \{v\}$ et $G_2 = G \setminus (\{v\} \cup \mathscr{V}(v))$. Par hypothèse de récurrence, $p = \alpha(G_1)$ et $q = \alpha(G_2)$ sont les nombres de stabilité de G_1 et G_2. Considérons alors un stable maximal S de G.
  - Si v appartient à S alors $S \cap \mathscr{V}(v) = \emptyset$ donc $S \setminus \{v\}$ est un stable de G_2, ce qui montre que $|S| - 1 \leqslant q$.
  - Si v n'appartient pas à S alors S est un stable de G_1 et $|S| \leqslant p$.

  Dans tous les cas, nous avons prouvé que $|S| \leqslant \max(p, q + 1)$.
  Réciproquement, considérons un stable maximal S_1 de G_1. S_1 est aussi un stable de G (puisqu'il ne contient pas v) donc $|S_1| = p \leqslant |S|$.
  De même, si S_2 est un stable maximal de G_2 il ne contient ni v ni voisin de v donc $S_2 \cup \{v\}$ est un stable de G. Ceci montre que $|S_2| + 1 = q + 1 \leqslant |S|$.
  En définitive nous avons prouvé que $\max(p, q + 1) \leqslant |S|$ et donc que $\alpha(G) = |S|$, ce qui achève la récurrence.

c) Considérons le cas de L_n, chemin non ramifié d'ordre n. Nous avons vu que $\alpha(L_n) = \max(\alpha(L_{n-1}), 1 + \alpha(L_{n-2}))$ donc la complexité c_n de cet algorithme dans ce cas de figure vérifie : $c_n = c_{n-1} + c_{n-2} + \Theta(1)$, ce qui conduit à : $c_n = \Theta(\varphi^n)$ avec $\varphi = \frac{1 + \sqrt{5}}{2}$. En conséquence, le coût de l'algorithme est exponentiel dans le pire des cas.

Remarque. La détermination du stable de cardinal maximum intervient dans les problèmes de coloration des graphes (puisque tous les éléments d'un même stable peuvent être colorés de la même façon). Il s'agit d'un problème NP complet au sens de la théorie de la complexité.

Exercice 9

a) Effectuer un BFS à partir d'un sommet d'un graphe G = (V, E) permet de déterminer la distance qui le sépare des autres sommets, et donc la longueur du plus long chemin qu'il est possible de tracer à partir de ce sommet. En effectuant ceci pour tout sommet du graphe, on en détermine le diamètre.
On sait qu'un BFS a un coût en $O(n + p)$ avec $n = |V|$ et $p = |E|$, donc cet algorithme détermine le diamètre en $O(n(n + p))$.

b) On propose l'algorithme suivant :

```
fonction DIAMÈTRE(arbre : G = (V, E))
    k ← 0
```

```
tant que |V| > 2 faire
    k ← k + 1
    V ← V \ {v ∈ V | deg v = 1}
si |V| = 2 alors
    renvoyer 2k + 1
else
    renvoyer 2k
```

Autrement dit, on effeuille l'arbre jusqu'à ne plus obtenir que 1 ou 2 sommets. Si k étapes ont été nécessaires, le diamètre vaut $2k$ s'il ne reste plus qu'un sommet, et $2k+1$ s'il en reste deux.

Notons G′ l'arbre obtenu en enlevant de G tous ses sommets de degré 1 ainsi que les arêtes associées (G′ est bien un arbre car il reste connexe et acyclique). Nous allons prouver que $d(\mathrm{G}) = d(\mathrm{G}') + 2$, d désignant le diamètre, ce qui suffira à valider l'algorithme.

Considérons un chemin maximal $(v_0, v_1, \ldots, v_{i-1}, v_i)$ tracé dans G. Nécessairement, v_0 et v_i sont de degré 1 car sinon ce chemin pourrait être prolongé et ne serait pas maximal. En revanche, tous les autres sommets de ce chemin sont au moins de degré 2, donc $(v_1, \ldots, v_{i-1})$ est un chemin de G′, et $d(\mathrm{G}) - 2 \leqslant d(\mathrm{G}')$.

Par ailleurs, tout chemin de G′ se prolonge au maximum par deux arêtes dans G, donc $d(\mathrm{G}') + 2 \leqslant d(\mathrm{G})$, ce qui prouve en définitive l'égalité $d(\mathrm{G}) = d(\mathrm{G}') + 2$.

Il est possible de déterminer le degré de chaque sommet d'un arbre en temps linéaire (voir par exemple l'exercice 4) donc la recherche des feuilles d'un arbre a un coût linéaire. Une fois les feuilles supprimées, il faut mettre à jour les listes d'adjacence, ce qui peut se faire là encore en coût linéaire. La détermination du diamètre d'un arbre a donc *a priori* un coût en $\mathrm{O}(kn)$ où k désigne le diamètre, autrement dit un $\mathrm{O}(n^2)$, ce qui n'est pas mieux que l'algorithme naïf puisque dans le cas d'un arbre, $p = n - 1$.

Cependant, on peut faire mieux en utilisant un tas pour ranger les sommets par ordre croissant de degré. Plus précisément, on peut réécrire l'algorithme de cette façon :

```
fonction DIAMÈTRE(arbre : G = (V, E))
    k ← 0
    pour tout v ∈ V faire
        d_v ← deg v
    tant que |V| > 2 faire
        k ← k + 1
        pour tout u ∈ V | d_u = 1 faire
            V ← V \ {u}
            pour tout v ∈ 𝒱(u) faire
                d_v ← d_v − 1
    si |V| = 2 alors
        renvoyer 2k + 1
    else
        renvoyer 2k
```

Déterminer chacun des degrés des sommets a un coût proportionnel au nombre d'arêtes, donc un $\mathrm{O}(n)$. Insérer chacun des sommets dans un tas ordonné par degré croissant a un coût en $\mathrm{O}(n \log n)$. Supprimer un sommet de degré 1 a un coût en $\mathrm{O}(\log n)$, et puisque chaque sommet est supprimé au plus une fois, ces suppressions ont un coût total en $\mathrm{O}(n \log n)$. Enfin, mettre à jour le degré d'un sommet après une suppression a un coût en $\mathrm{O}(\log n)$, et puisque la somme des degrés est égal à $2n - 2$, le coût total de ces opérations ne peut excéder $\mathrm{O}(n \log n)$.

Au final, on obtient un algorithme de coût semi-linéaire : $\mathrm{O}(n \log n)$.

• Plus court chemin

Exercice 10 Observons les termes diagonaux des matrices $\mathrm{M}^{(k)}$ dans l'algorithme de Floyd-Warshall ; ils vérifient la récurrence : $m_{ii}^{(k+1)} = \min\left(m_{ii}^{(k)}, m_{i,k+1}^{(k)} + m_{k+1,i}^{(k)}\right)$.

Lorsqu'il n'existe pas de cycle de poids strictement négatif, et sachant que $m_{ii}^{(0)} = 0$, on a $m_{ii}^{(n)} = 0$: tous les termes diagonaux de la matrice $\mathrm{M}^{(n)}$ sont nuls.

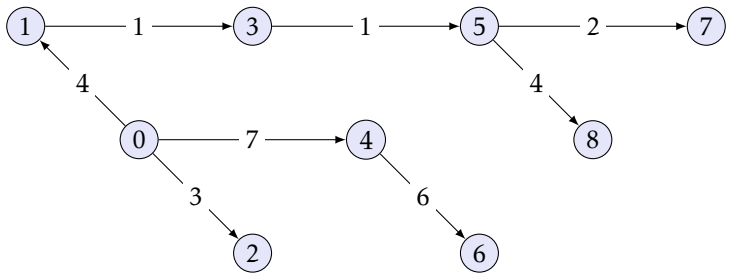
S'il existe un cycle de poids strictement négatif, considérons la plus petite valeur de k telle qu'il soit possible de tracer un tel cycle C parmi les sommets $v_1, \ldots, v_{k+1}$. On observera que ce cycle passe nécessairement par v_{k+1},

et que si v_i est un autre sommet de ce cycle, il existe deux chemins $v_i \rightsquigarrow v_{k+1}$ et $v_{k+1} \rightsquigarrow v_i$ dont la somme des poids est strictement négative.
Constatons maintenant que jusqu'au rang k, l'algorithme de Floyd-Warshall reste valide : $m_{ij}^{(k)}$ désigne le poids d'un chemin minimal reliant v_i à v_j en ne passant que par des sommets de $v_1, \dots, v_k$, puisqu'il n'existe pas de cycle de poids négatif parmi ces sommets. D'après l'observation précédente, si v_i est un sommet du cycle C, alors $m_{i,k+1}^{(k)} + m_{k+1,i}^{(k)} < 0$, ce qui implique $m_{ii}^{(k+1)} < 0$, puis $m_{ii}^{(n)} < 0$.
On peut donc affirmer qu'il existe un cycle de poids strictement négatif si et seulement s'il existe $i \in [\![1, n]\!]$ tel que $m_{ii}^{(n)} < 0$.

Exercice 11 Suivons l'évolution de l'algorithme de Dijkstra dans le tableau ci-dessous :

S	0	1	2	3	4	5	6	7	8
{0}	·	4	3	+∞	7	+∞	+∞	+∞	+∞
{0, 2}	·	4	·	+∞	7	+∞	+∞	+∞	+∞
{0, 2, 1}	·	·	·	5	7	+∞	+∞	+∞	+∞
{0, 2, 1, 3}	·	·	·	·	7	6	+∞	+∞	+∞
{0, 2, 1, 3, 5}	·	·	·	·	7	·	+∞	8	10
{0, 2, 1, 3, 5, 4}	·	·	·	·	·	·	13	8	10
{0, 2, 1, 3, 5, 4, 7}	·	·	·	·	·	·	13	·	10
{0, 2, 1, 3, 5, 4, 7, 8}	·	·	·	·	·	·	13	·	·
{0, 2, 1, 3, 5, 4, 7, 8, 6}	·	·	·	·	·	·	·	·	·

Dans le même temps, on mémorise les prédécesseurs de chaque sommet dans un chemin optimal, ce qu'on peut représenter sur le graphe :



Le chemin retourné par l'algorithme de Dijkstra est donc : $(0, 1, 3, 5, 8)$.

Exercice 12

a) Dans le tableau ci-dessous, on visualise l'évolution des valeurs du tableau d au cours de l'algorithme. Les différents tests de la deuxième partie de celui-ci reviennent à itérer une fois de plus les valeurs de d pour voir si l'une d'entre-elles est susceptible d'être encore modifiée ; voila pourquoi on applique la boucle une fois de plus dans le tableau.

k	d_1	d_2	d_3	d_4	d_5
	0	+∞	+∞	+∞	+∞
1	0	6	+∞	+∞	7
2	0	6	4	2	7
3	0	2	4	2	7
4	0	2	4	−2	7
	0	2	4	−2	7

Les valeurs du tableau d ne sont plus modifiée, l'algorithme se termine en renvoyant la valeur « Vrai ».
En revanche, si l'arête $4 \to 5$ est de poids 8, la dernière ligne de ce tableau est modifiée :

k	d_1	d_2	d_3	d_4	d_5
	0	+∞	+∞	+∞	+∞
1	0	6	+∞	+∞	7
2	0	6	4	2	7
3	0	2	4	2	7
4	0	2	4	−2	7
	0	2	4	−2	6

et l'algorithme renvoie donc la valeur « Faux ». On peut observer l'existence d'un cycle de poids négatif dans ce cas : $(2,4,5,3,2)$.

b) A l'instar de l'algorithme de Dijkstra, on démontre par récurrence qu'à l'étape k, d_u est égal au poids d'un chemin minimal allant de la source au sommet u sans passer par plus de k sommets. Sachant qu'un chemin minimal est de longueur au plus $n-1$, à la fin de la première partie de l'algorithme chaque valeur d_u est égale à $\delta(s,u)$ s'il n'y a pas de cycle de poids strictement négatif. Ainsi on aura pour tout u et v : $d_v \leqslant d_u + w(u,v)$ et la valeur retournée sera « Vrai ».

c) Supposons l'existence d'un cycle $(v_0,\dots,v_k)$ de poids strictement négatif avec $v_k = v_0$. Nous avons donc :

$$\sum_{i=1}^{k} w(v_{i-1},v_i) < 0.$$

Raisonnons par l'absurde en supposant que la réponse retournée par l'algorithme soit « Vrai ». Alors pour tout $i \in [\![1,k]\!]$, $d_{v_i} \leqslant d_{v_{i-1}} + w(v_{i-1},v_i)$. En sommant ces inégalités on obtient :

$$\sum_{i=1}^{k} d_{v_i} \leqslant \sum_{i=1}^{k} d_{vi-1} + \sum_{i=1}^{k} w(v_{i-1},v_i)$$

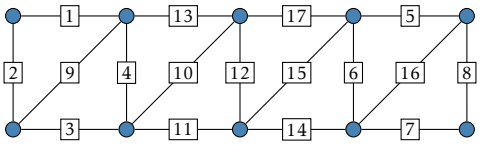
soit en simplifiant : $\displaystyle\sum_{i=1}^{k} w(v_{i-1},v_i) \geqslant 0$, ce qui est absurde.

La réponse retournée dans le cas où il existe un cycle de poids strictement négatif est donc « Faux ».

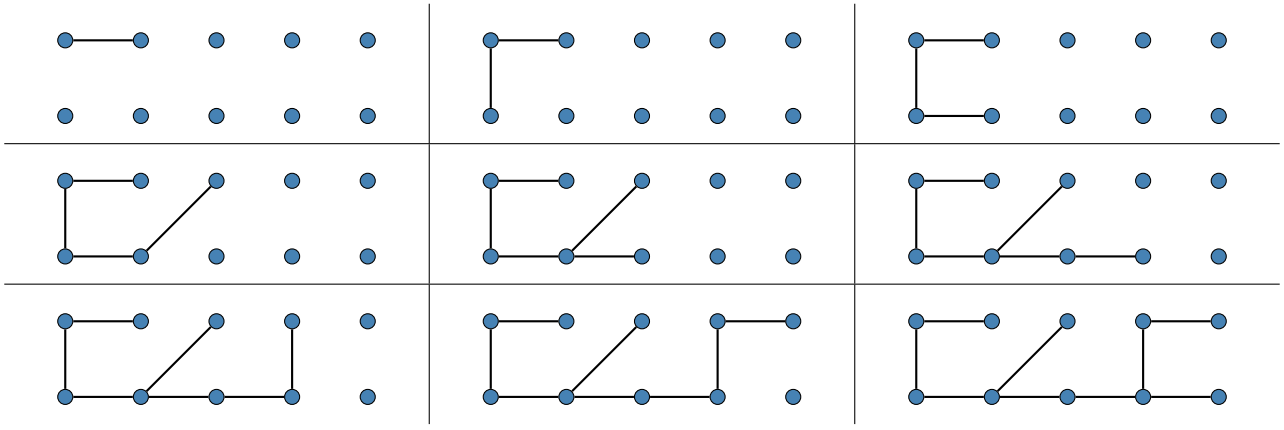
d) Posons $n = |\mathrm{V}|$ et $p = |\mathrm{E}|$. L'initialisation du tableau d a un coût temporel en $\Theta(n)$; le corps principal de la fonction est en $\Theta(np)$ et la détermination de l'existence d'un cycle de poids strictement négatif est un $\mathrm{O}(p)$ donc le coût total de cet algorithme est un $\Theta(np)$.

• Arbre couvrant minimal

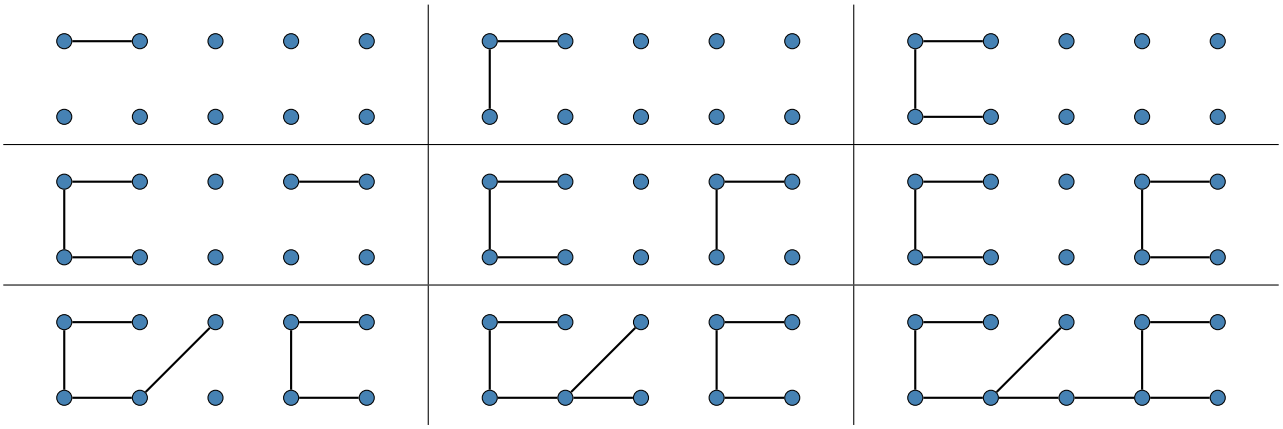
Exercice 13 Le résultat des algorithmes de Prim et Kruskal dépend de l'ordre dans lequel on traite des arêtes de poids égal. Je commence donc par choisir un ordre croissant arbitraire entre ces différentes arêtes :



Suivant cet ordre, voici le déroulement de l'algorithme de Prim à partir du sommet 0 :



Voici maintenant, toujours avec le même ordre, l'algorithme de Kruskal :



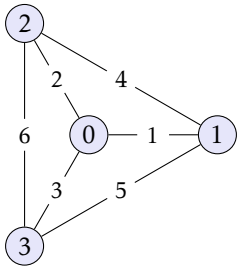
Exercice 14 Raisonnons par l'absurde en supposant qu'un tel arbre n'existe pas, et considérons un arbre couvrant (V, B) tel que $|A \cap B|$ soit maximal. On considère alors une arête $(a, b) \in A \mid (a, b) \notin B$.
Puisque (V, B) est un arbre couvrant, il existe un chemin $a \rightsquigarrow b$ n'empruntant que des arêtes de B. Ce chemin contient au moins une arête (c, d) qui n'appartient pas à A, car dans le cas contraire A contiendrait un cycle. On pose alors $B' = B \setminus \{(c, d)\} \cup \{(a, b)\}$. (V, B') est toujours un arbre couvrant mais $|A \cap B'| > |A \cap B|$, ce qui est absurde.

Une autre solution consiste à affecter un poids à chacune des arêtes en posant $w(e) = 1$ si $e \in A$ et $w(e) = 2$ sinon, puis à appliquer l'algorithme de Kruskal. Celui-ci va sélectionner chacune des arêtes de A puis les compléter pour retourner un arbre couvrant (minimal pour cette répartition des poids).

Exercice 15 Considérons le graphe $G' = (V, E \setminus \{e\})$. Puisque $e \notin A'$, (V, A') est toujours un arbre couvrant de G' (ce qui prouve en particulier que G' reste connexe).
En revanche, $(V, A \setminus \{e\})$ n'est plus un arbre (il n'a plus assez d'arêtes) et possède deux composantes connexes. Posons $e = (u, v)$. Puisque (V, A') est un arbre couvrant de G', il contient un chemin menant de u à v et donc une arête $e' \in A'$ reliant la composante connexe de u à celle de v. On en déduit que $(V, A' \setminus \{e'\} \cup \{e\})$ est un arbre couvrant de G. Puisque (V, A') est de poids minimal, ceci prouve que $w(e) - w(e') \geqslant 0$, soit $w(e') \leqslant w(e)$.
Mais e a été choisi de poids minimal dans $A \triangle A'$, donc $w(e') = w(e)$.
Par contraposée il en résulte immédiatement que si w est injective $A \triangle A' = \emptyset$, autrement dit $A = A'$, ce qui prouve l'unicité de l'arbre couvrant minimal.

Exercice 16

a) Considérons le graphe ci-dessous :



L'arbre couvrant minimal est unique et a un poids total égal à 6, mais il y a deux arbres couvrants de poids 8 (et aucun de poids 7).

b) Soit e' une arête de B *de poids minimal* parmi celles qui ne se trouvent pas dans A. Posons $e' = (u', v')$. Il existe un unique chemin $u' \rightsquigarrow v'$ n'utilisant que des arêtes de A ; notons e une de celles qui n'appartiennent pas à B (il en existe au moins une car B ne contient pas de cycle). Alors $(V, A \setminus \{e\} \cup \{e'\})$ reste un arbre couvrant ; puisque A est l'unique arbre couvrant minimal on a $w(e) < w(e')$.
Mais $(V, B \setminus \{e'\} \cup \{e\})$ est aussi un arbre couvrant, de poids strictement inférieur à (V, B) puisque $w(e) < w(e')$. Par définition de B il ne peut s'agir que de l'unique arbre de poids minimal, à savoir (V, A). Ceci prouve que $B = A \setminus \{e\} \cup \{e'\}$.

c) À partir de $u \in V$, effectuons un parcours en largeur de (V, B) à la recherche de l'arête de poids maximal entre u et chacun des autres sommets. Ceci permet de calculer $\max_B(u, v)$ pour tout $v \in V$ pour un coût en $O(|B|) = O(n)$. Effectuons ceci pour chacun des sommets $u \in V$; on obtient les valeurs de $\max_B(u, v)$ pour tout $(u, v) \in V^2$ pour un coût total en $O(n^2)$.

d) Posons $n = |V|$ et $p = |E|$ et commençons par calculer l'arbre couvrant minimal (V, A). Nous savons que ceci peut être réalisé en $O(p \log n) = O(n^2)$. Calculons alors toutes les valeurs $\max_A(u, v)$ pour $(u, v) \in V^2$; nous venons de montrer que ceci peut être réalisé en $O(n^2)$. Déterminons enfin une arête $e \in E \setminus A$ qui minimise $w(e) - w(\max_A(e))$; ceci peut être réalisé en $O(p) = O(n^2)$. Compte tenu de la question 2 on peut alors affirmer que $(V, A \setminus \{\max_A(e)\} \cup \{e\})$ est un arbre second par ordre croissant de poids, et qu'il a été déterminé en un temps $O(n^2)$.

Exercice 17

a) Raisonnons par récurrence sur $|A|$.

- Si $|A| = 1$, $d(C) = 0$ et le résultat est évident.
- Si $|A| > 1$, supposons le résultat acquis au rang $|A| - 1$, et considérons le premier sommet $v \in A$ de degré 1 qui apparait dans C. Notons u son unique voisin ; puisque le cycle C est décrit par un parcours en profondeur on a $C = (x_1, \ldots, u, v, x_i, \ldots, x_k)$. Notons $A' = A \setminus \{v\}$ et $C' = (x_1, \ldots, u, x_i, \ldots, x_k)$. A′est toujours un arbre et C′ est toujours décrit par un parcours en profondeur de A′ donc par hypothèse de récurrence $d(C') \leqslant 2d(A')$.

  On a $d(C) = d(C') - d(u, x_i) + d(u, v) + d(v, x_i)$ et $d(A) = d(A') + d(u, v)$ donc :

$$d(C) \leqslant 2d(A) - d(u, x_i) - d(u, v) + d(v, x_i) \leqslant 2d(A)$$

  d'après l'inégalité triangulaire.

b) Considérons un cycle couvrant minimal C_m, et ôtons-lui une de ses arêtes. On obtient un arbre couvrant A, et $d(A) \leqslant d(C_m)$. Si A_m est un arbre couvrant minimal on a $d(A_m) \leqslant d(A)$ donc d'après la question précédente, si C est un cycle décrit par un parcours en profondeur de A_m on a :

$$d(C) \leqslant 2d(A_m) \leqslant 2d(A) \leqslant 2d(C_m).$$