Devoir 3

BLAIS REGOUT, Lucien - 18073291
SAVOIE, Olivier - 18114274

IFT436 - Algoritmes et structures de données

Faculté des Sciences
Université de Sherbrooke

Présenté à Pr BLONDIN, Michael

Jeudi, 10 octobre 2019

Question 1

a)

$V =$ Les bassins du manège aquatique (sommets).

$E =$ Les corridors entre les bassins, potentiellement des glissades (arêtes).

$\mathcal{G} =$ Manège aquatique est dirigé.

b)

$\mathcal{G}$ ne peut pas être cyclique dû aux contraintes stipulées. Une contrainte particulière, ci-dessous, en témoigne.

> Il n'y a *pas* de corridor passant d'un bassin vers lui-même.

Il est impossible qu'un cycle soit établi avec un seul bassin considérant qu'il y a aucun corridor qui part d'un dit bassin en arrivant dans ce même bassin.

Finalement, cette seconde contrainte, implique que, dû à l'inclinaison, l'un ne peut pas revenir au bassin précédent. Cela implique qu'il y a aucun cycle possible et que de ce fait, le manège aquatique $\mathcal{G}$ est acyclique et dirigé.

> Puisque les bassins sont situés de plus en plus bas, le long d'une colline, si on peut atteindre un bassin j à partir d'un bassin i, alors on ne peut pas atteindre le bassin i à partir du bassin j.

c)

Si le graphe $\mathcal{G}$ est considéré non dirigé, il y a plusieurs arrangements d'arêtes qui rendrait $\mathcal{G}$ cyclique et pour lesquels $|E| > |V| - 1$. Ainsi, dans la majorité des cas, $\mathcal{G}$ n'est pas un arbre, car il ne respecte pas les propriétés essentielles pour en être un. Toutefois, il y a toujours la possibilité qu'aucun cycle, simple ou non, ne s'y retrouve et que le nombre d'arête soit équivalent au nombre de sommet -1. De sorte que les propriétés suivantes soit respectées.

— $\mathcal{G}$ est $connexe$ et $|E| = |V| - 1$

d)

Dans le cas minimal, les arêtes placées représenterons un graphe $\mathcal{G}$ qui sera forcément un arbre, respectant les contraintes, ainsi que l'explication élaborée à la lettre précédente, **c)**.

$$min = n - 1, \quad \forall n \geq 1$$

Advennant que chacun des bassins présents se voient attribués un corridor vers chacun des bassins soudjacents, l'équation maximal reflètera la composition du chemin le plus long.

$$max = \sum_{i=1}^{n} n - i = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n}{2}(n+1) = \frac{n^2 - n}{2}$$

e)

Le **bassin de départ :** est le seul bassin qui ne vas jamais se retrouver à droite dans une paire de sommet, donc qui ne sera jamais j dans un couple (i, j).

Le **bassin d'arrivée :** est le seul bassin qui ne vas jamais se retrouver à gauche dans une paire de sommet, donc qui ne sera jamais i dans un couple (i, j).

f)

Algorithme 1 : Calcul du temps maximal passé dans une attraction $\mathcal{G}$.

Entrées : Liste d'adjacence, tableau asssociatif $S : [n] \times [n] \to \mathbb{N} > 0$.

Résultat : Le temps maximal.

```
1  MaxSommet ← [n]
2  SommeTemps ← 0
3  Enfant ← 0
4  parcours (x) :
5      si x non marquée alors
6          pour v ∈ S[x] faire
7              parcours(v)
8              SommeTemps ← MaxSommet[Enfant] + c[x, v]
9              si SommeTemps > MaxSommet[x] alors
10                 MaxSommet[x] ← SommeTemps
11             marquer x
12     Enfant ← x − 1
13 parcours(u)
14 retourner MaxSommet[0]
```

Dans cette algorithme, un sommet est marqué uniquement lorsque tous ces voisins ont été marqués. Le fait d'utiliser un tableau dans lequel se trouve le temps maximal pour chaque sommet, il n'est plus nécéssaire de faire tous les chemins possibles. On a juste besoin d'arreter, dès qu'on voit un sommet marqué et prendre la valeur qui lui est attribué.

Cet algorithme se rappele récursivement sur chacun des enfants soudjacents, jusqu'à ce qu'on rencontre un sommet qui a déjè été marqué. Ainsi, on peut en déduire que son temps d'exécution sera, dans le pire cas, $\mathcal{O}(|E| + |V|)$ ce qui respecte la contrainte comme quoi l'algorithme doit être e de $\mathcal{O}(n + m)$.

g)

Algorithme 2 : Calcul de la séquence au temps maximal passé dans une attraction $\mathcal{G}$, donné 5 remontées.

Entrées : Liste d'adjacence, tableau asssociatif $c : [n] \times [n] -> N > 0$ et tableau asssociatif $r : [n] -> \mathbb{N} > 0$

Résultat : Le temps maximal.

```
1  MaxSommet ← [n]
2  SommeTemps ← 0
3  Enfant ← 0
4  parcours (x) :
5      si x non marquée alors
6          pour v ∈ S[x] faire
7              parcours(v)
8              SommeTemps ← MaxSommet[Enfant] + c[x, v]
9              MaxSommet[x − 1] ← MaxSommet[x − 1] + SommeTemps
10         MaxSommet[x − 1] ← MaxSommet[x − 1] + 5 · r[x]
11     Enfant ← x − 1
12 parcours(u)
13 retourner MaxSommet[0]
```

Ici, il fera exactement comme l'algorithme précédent, à l'exception que lorsqu'il se rendra au plus long parcours précédent le bassin final, et se 5 fois. La 5e fois, le sera possiblement différent mais restera cependant couvers par l'algorithme.

Question 2

a)

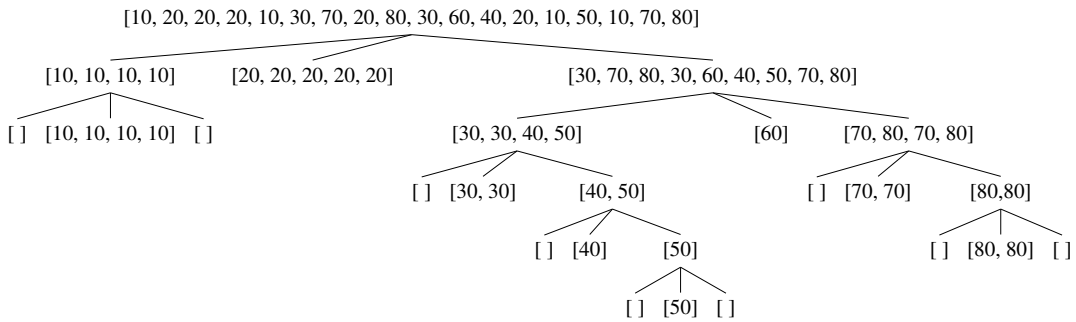
Algorithme 3 : Tri d'une séquence et calcul des valeurs modales.

Entrées : Une séquence S non vide d'éléments comparables.

Résultat : Les valeurs modales m trouvées dans la séquence S

```
1  mode ← [ ]
2  modeQty ← 0
3  trouverMode (x) :
4      si |x| = 0 alors
5          retourner mode                                  // 1re condition d'arret.
6      sinon
7          pivot ← mediane (x)
8          gauche ← [x ∈ S : x < pivot]
9          centre ← [x ∈ S : x = pivot]
10         droite ← [x ∈ S : x > pivot]
11         si |centre| = modeQty alors
12             ajouter pivot à mode
13         si |centre| > modeQty alors
14             modeQty ← |centre|
15             mode ← [pivot]
16         sinon
17             trouverMode (gauche)
18             trouverMode (droite)     // 2e cond. d'arret ≠ trouverMode(centre).
19 retourner trouverMode (S)
```

b)



Question 3

a)

> **Donnez un algorithme qui détermine si un graphe non dirigé est un arbre.**

Algorithme 4 : Détermination de la propriété d'arbre d'un graphe.

Entrées : Un graphe$\mathcal{G}$ tel que $\mathcal{G}=(V,E)$ est non dirié et un sommet $u \in V$

Résultat : Si le graphe est un arbre, ou non.

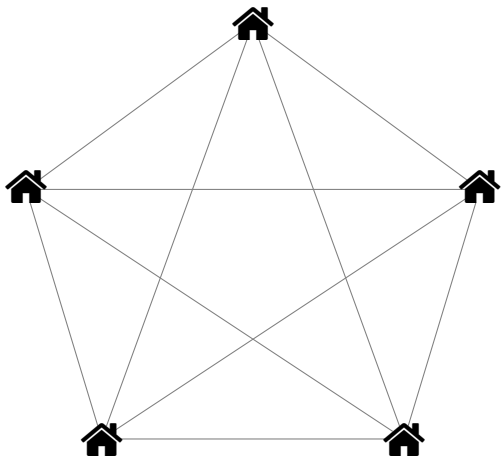
```
1  Visites ← 0
2  visiter(x) :
3      marquer x
4      Visites ← Visites + 1
5      pour tous voisins de x ∈V : x → y faire
6          si y non marqué alors
7              visiter(y)    // Récursif, mais jamais plus d'une fois par noeud.
8  visiter(u)
9  si Visites ≠ |V| alors
10     retourner Non_Arbre
11 sinon
12     retourner Est_Arbre
```

Cet algorithme parcours le graphe au complet. Si un sommet est visité plus d'une fois, c'est cyclique. Ceci dit, cet algorithme se rappele sur chacun des noeuds un maximum d'une fois. À la fin de son exécution, si la variable $Visites = |V|$, nous avons bien un arbre ! Ainsi, nous pouvons affirmer que le temps d'exécution maximal correspondra à $\mathcal{O}(|V|)$. Définissant n pour le nombre de sommet, le temps est alors de $\mathcal{O}(n)$.

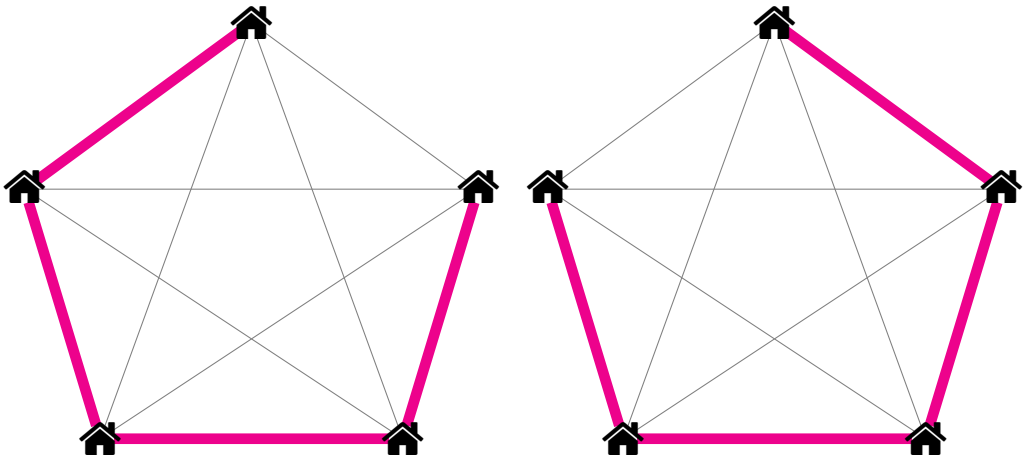
b)

Démontrez que le graphe complet $\mathcal{G}_n$ possède au moins 2^n arbres couvrants, $\forall n \in \mathbb{N} > 3$.

L'énoncé stipule que pour $n = 4$ où n est le nombre de gites, il y a $2^4 = 16$ réseaux possibles. Si nous ajoutons une autre gites, nous obtenons ce dernier, à 5 gites.



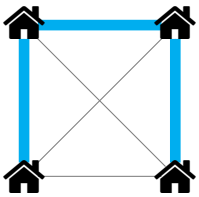
Afin de prouver qu'il y a au minimum 2^n chemins possibles, utilisons un sous-graphes $\mathcal{H}$ de $\mathcal{G}$. En prenant le sous-graphe $\mathcal{H}$ possedant 4 sommets (gites), il est possible de savoir qu'il y a 2^n chemins possibles. De ce fait, on peut voir, que ce nombre est doublé lorsque le graphe est complet (donc avec 5 sommets). En effet, pour chacune des chemins en $n = 4$, il y a 2 chemins possibles lorsque $n = 5$.



Donc pour $n = 5$, le nombre de chemin minimum possible égale $2^4 \cdot 2^1 = 2^5$ chemins possible. Donc il est possible de dire que pour un graphe complet, il existe au minimum 2^n arbre couvrant $n \in \mathbb{N} \geq 4$.

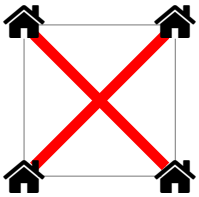
c)

> **Donnez un exemple de points où un arbre couvrant minimal entre ces points est plus long qu'un arbre couvrant minimal avec un nouveau point ajouté (une halte).**



$$3 * 1 \ = \ 3u$$

Dans cette image, on retrouve un arbre couvrant minimal qui couvre 4 sommets d'un graphe$\mathcal{G}$ à 4 sommets. Assumant qu'il y a 1 unité (u) de distance entre les coins, du carré formé, adjacants, le grapheaurait une longueur suivant l'équation ci-dessus.



$$4 \cdot (\tfrac{1}{2}) \cdot \sqrt{(1^2 + 1^2)} = \\ 2 \cdot \sqrt{2} = 2.8284 \ldots u$$

Maintenant, ces gites sont équidistant de cette halte, sont à la même position que le graphe précédent et la distance totale de chacun des gites à la halte va comme suit.