TD de Complexité/Algorithmique

Année 2017-18

Version 1.1

Université de Montpellier
Place Eugène Bataillon
34095 Montpellier Cedex 5

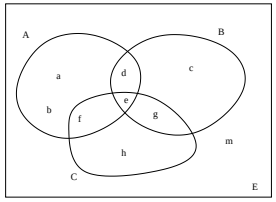
Annie Chateau et Rodolphe Giroudeau
161, rue Ada
34392 MONTPELLIER Cedex 5
TEL : 04-67-41-85-40
Mail : {chateau,rgirou}@lirmm.fr

Université de Montpellier — 2016-2017
HMIN329 – Complexité et algorithmes — Durée: 4.5h

Notions de base
TD – Séance n 1

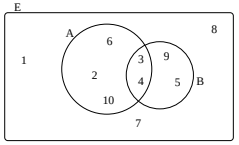
1 Ensembles

Exercice 1 – Opérations ensemblistes
Dans cet exercice, on considère les ensembles E, A, B et C décrits dans le diagramme de Venn de la figure ci-contre. Dites si les affirmations suivantes sont justes ou fausses :



$g \in A \cap \overline{B}$	$g \in \overline{A} \cap \overline{B}$
$f \in C \setminus \overline{A}$	$g \in \overline{A} \cup \overline{B} \cup \overline{C}$
$e \in \overline{A} \cap \overline{B} \cap \overline{C}$	$m \in \overline{A} \cap \overline{B} \cap \overline{C}$
$\{h, m\} \subset \overline{A} \cap \overline{B}$	$(A \setminus B) \cup C \cup \{c\} \in \mathcal{P}(E)$

Exercice 2 – Opérations ensemblistes



Soient E, A, et B les ensembles décrits par le diagramme de Venn ci-contre, trouvez
- M tel que $M \cap A = \emptyset$ et $M \cap B = \{9\}$
- $\mathcal{O}$ tel que $\mathcal{O} \cap A = \{2, 6\}$ et $\mathcal{O} \cap B = \{3, 5\}$
- Q tel que $Q \setminus A = \{1, 5\}$ et $Q \setminus B = \{1, 6\}$

Exercice 3 – Ensemble des parties Donnez en extension l'ensemble $\mathcal{P}(\{1, 2, 3, 4\})$.

Exercice 4 – Ensemble des parties A–t–on $\emptyset = \mathcal{P}(\emptyset)$? Calculez $\mathcal{P}(\mathcal{P}(\emptyset))$.

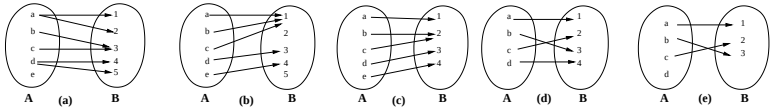
Exercice 5 – Ensemble des parties Soient deux ensembles E, F. Comparez $\mathcal{P}(E \cap F)$ avec $\mathcal{P}(E) \cap \mathcal{P}(F)$ d'une part, et $\mathcal{P}(E \cup F)$ avec $\mathcal{P}(E) \cup \mathcal{P}(F)$ d'autre part.

2 Relations binaires, applications

Exercice 6 – Type de relation
Les ensembles qui suivent définissent–ils une relation binaire fonctionnelle ? Si oui, donnez un ensemble de départ qui en fasse une application, vous direz ensuite si elle est injective, et s'il existe un choix de l'ensemble d'arrivée qui la rende surjective.

1. a, b sont deux symboles. $\{(0, a), (1, a), (2, b)\}$
2. $\{(x, y) \in \mathbb{N}^2 : x + y \leqslant 4\}$
3. $A = \{(x, y) \in \mathbb{N}^2 : x \geqslant 4 \text{ et } y \geqslant 3\}$

Exercice 7 – Diagramme sagittal On rappelle qu'une relation binaire de l'ensemble A vers l'ensemble B est définie par son graphe, une partie de $A \times B$. On peut la représenter par un diagramme sagittal. Dites parmi les diagrammes sagittaux suivants lesquels sont :
- le graphe d'une application
- injective
- surjective
- bijective



Exercice 8 – Injectivité, surjectivité Soit $d : \mathbb{N} \longrightarrow \mathbb{N}$, $n \longmapsto 2n$ l'application double et $m : \mathbb{N} \longrightarrow \mathbb{N}$, $n \longmapsto n \div 2$ (division entière) l'application moitié. Sont-elles injectives, surjectives, bijectives ? Mêmes questions pour $d \circ m$ et $m \circ d$.

Exercice 9 – Injectivité, surjectivité Soit E un ensemble, et soit $(A, B) \in \mathcal{P}(E)$ deux parties de E.

$$f : \mathcal{P}(E) \longrightarrow \mathcal{P}(A) \times \mathcal{P}(B), \quad X \longmapsto (X \cap A, X \cap B)$$

1. Donnez un exemple avec E fini de cardinal 6, A et B parties de E de cardinaux 4 et 3 ayant deux élément communs.
   Pour cet exemple, f est–elle injective. surjective. bijective ?
2. Trouvez des conditions nécessaires et suffisantes sur A et B, en général, pour que f soit *i.* injective. *ii.* surjective. *iii.* bijective.

Exercice 10 – Injectivité, surjectivité Soit une application $f : E \longrightarrow F$, A une partie de E et B une partie de F.
Montrer que l'image directe d'une partition de E n'est pas toujours une partition de F.
Montrer que, lorsque f est surjective, l'image réciproque d'une partition de F par f est une partition de E.

3 Cardinalité d'ensembles, dénombrement

Exercice 11 – Cardinalité et opérations ensemblistes A, B sont des parties de E un ensemble fini. Exprimez en fonction de $|A|$, $|A \cap B|$, $|B|$ les cardinaux des ensembles $A \cup B$, $A \setminus B$ et $A \times B$.

Exercice 12 – Cardinalité et opérations ensemblistes Soit E un ensemble fini non vide. Peut–on construire une application bijective de $E \times E$ vers E ?

Exercice 13 – Cardinalité et injectivité, surjectivité Soient E, F deux ensembles finis non vides, et A une partie de E, B une partie de F.

1. Faites un diagramme sagittal, pour une application f quelconque, des ensembles E, F, A, B de tailles respectives 5,6,3,2. Déterminez, avec votre exemple $f(A)$, $f^{-1}(B)$.
2. En général quelle est la relation entre $|A|$ et $|f(A)|$? $(<, =, \leqslant)$. Et entre $|B|$ et $|f^{-1}(B)|$?
3. Que deviennent vos relations quand f est injective ? surjective ?
4. Y a-t-il une réciproque, c'est à dire une relation entre les cardinaux qui impliquerait le caractère injectif, ou surjectif de f ?

Exercice 14 – Comptage Soit un ensemble de 6 couleurs $Coul = \{R, V, J, B, M, C\}$, et un ensemble de points $E = \{p_1, p_2, \ldots, p_n\}$, avec $n \geqslant 1$. Un *coloriage* associe à **chaque** point de E **une** couleur.

1. Montrez qu'un coloriage est une application. De quel ensemble vers quel ensemble ?
2. En fonction de n, indiquez le nombre de coloriages différents possibles.
3. Si $n = 5$ peut-on assurer que dans chaque coloriage on aura au moins 2 points de la même couleur ?
4. Et si $n = 6$? $n = 7$?
5. Toujours avec 6 couleurs, combien de points faut-il pour être certain que chaque coloriage ait au moins 3 points de la même couleur ?
6. Quand $n = 3$ on *tricolorie* les 3 points p_1, p_2, p_3 en leur associant 3 couleurs différentes. Combien existe-t-il de tricoloriages différents ?

Exercice 15 – Comptage Soit $E = \{A, B, C, D\}$.

1. Combien de mots de trois lettres peuvent être formés à partir de l'alphabet E en autorisant les répétitions ? sans autoriser les répétitions ?
2. commençant par un A, avec puis sans répétitions ?
3. ne contenant pas de A, avec puis sans répétitions ?
4. contenant un A, avec puis sans répétitions ?

4 Récurrence, induction

Exercice 16 – Récurrence simple Montrer par récurrence que les propriétés suivantes sont vraies sur une partie de $\mathbb{N}$ que vous déterminerez.

1. $1 + 2 + \ldots + n = n(n+1)/2$.
2. $n + 1 < n^2 < 2^n < n!$.

Exercice 17 – Récurrence simple Trouvez l'erreur dans le raisonnement par récurrence suivant :

Soit $P(n)$ la propriété : *« Dans tout groupe de n personnes, toutes les personnes ont le même âge »*.

Base : $P(1)$ vraie : évident

Récurrence : Supposons que $P(n)$ est vraie pour $n \geq 1$ et montrons que $P(n+1)$ est vraie :

Soit G un groupe de $n+1$ personnes que l'on numérote de 1 à $n+1$. Soit G_1 (resp. G_2) le groupe formé des n premières (resp. dernières) personnes. Puisque $P(n)$ vraie, toutes les personnes de G_1 (resp. de G_2) ont le même age. Or la personne numérotée n est à la fois dans G_1 et dans G_2. Donc toutes les personnes de G ont le même âge que la personne numérotée n, donc toutes les personnes de G ont le même âge. Ceci démontre $P(n+1)$.

On en déduit que $\forall n \geqslant 1 : P(n)$.

Exercice 18 – Récurrence simple On suppose que n est entier non nul. Soit un échiquier ayant 2^n cases par coté. On enlève une case de coin à cet échiquier. Un trimino est un morceau d'échiquier de la forme : ⊞ .

1. Faites un dessin pour $n = 1$ et $n = 2$, et montrez comment recouvrir cet échiquier auquel on a enlevé une case de coin par des triminos.
2. Faites un dessin pour $n = 3$, et servez vous du résultat précédent pour montrer comment recouvrir cet échiquier auquel on a enlevé une case de coin par des triminos pour tout $n >= 1$.
3. Prouvez que l'on peut recouvrir par des triminos, un échiquier ayant 2^n cases par coté et auquel on a enlevé une case de coin.
4. Prouvez que le recouvrement est possible quel que soit l'emplacement de la case que l'on enlève à l'échiquier.
5. Déduire de la question précédente que $\forall n \in \mathbb{N} : (2^{(2n)} - 1$ est divisible par 3).

Exercice 19 – Induction structurelle Soit A^* l'ensemble des mots construits à partir d'un alphabet comportant 3 lettres, $A = \{a, b, c\}$.

On définit alors E partie de A^* par l'induction suivante :

– *Base :* Les mots ϵ, a,b et c sont dans E.
– *Règle :* $\forall u \in E : a.u.a, b.u.b$ et $c.u.c \in E$

1. Construisez tous les mots de E de longueur ≤ 4.
2. Démontrez, en utilisant uniquement la définition inductive de E, que chaque mot de E est un palindrome. On rappelle qu'un mot w est un palindrome si $w = {}^t w$, où ${}^t w$ est l'écriture « inverse » de w – de droite à gauche. Exemple *ababbaba* est un palindrome, alors que *ababcaba* n'en est pas un.
3. Démontrez, par récurrence sur la longueur d'un palindrome de A^*, que tout palindrome est dans E. Conclusion ?

5 Relations binaires, équivalences, ordres

Exercice 20 – Transitivité Soient $\mathcal{R}$ et $\mathcal{R}'$ deux relations binaires définies dans le même ensemble. Montrer que si $\mathcal{R}$ et $\mathcal{R}'$ sont transitives alors $\mathcal{R} \cap \mathcal{R}'$ est transitive. Que penser de $\mathcal{R} \cup \mathcal{R}'$?

Exercice 21 – Transitivité Prouver qu'une relation binaire $\mathcal{R}$ définie dans un ensemble A est transitive si et seulement si $\mathcal{R}^2 \subseteq \mathcal{R}$.

Exercice 22 – Transitivité Soit la relation $\mathcal{R}$ définie dans $\mathbb{N}$ par $x\mathcal{R}y$ si $y = x + 1$. La relation $\mathcal{R}$ est–elle fonctionnelle ? réflexive ? symétrique ? transitive ? Définissez $\mathcal{R}^{-1}$, $\mathcal{R}^2$, $\mathcal{R}^3$, $\mathcal{R}^+$ et $\mathcal{R}^*$.

Rappel : $\mathcal{R}^+ = \bigcup_{n \geq 1} \mathcal{R}^n$ et $\mathcal{R}^* = \mathcal{R}^+ \cup I$, où I est l'identité.

Exercice 23 – Équivalence Soit $X = \{a, b, c, d, e\}$ et $Y = \{a, b, c\}$. $\mathcal{R}$ la relation binaire définie sur $\mathcal{P}(X)$ par $M \mathcal{R} N \Leftrightarrow M \cap Y = N \cap Y$.

1. Soient les parties $A = \{a, b, d\}, B = \{a, b, e\}, C = \{a, d, e\}$. Quelles sont celles qui sont en relation ?
2. Montrer que $\mathcal{R}$ est une relation d'équivalence.
3. Quelle est $\bar{A}$ la classe de A ? Quel est le nombre d'éléments de l'ensemble quotient ?

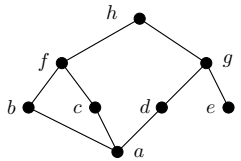
Exercice 24 – Relations d'ordre Soient les relations binaires :

1. $\subseteq$ dans $\mathcal{P}(\{a, b, c\})$.
2. $\{(1,1), (1,2), (1,3), (1,4), (1,5), (2,2), (2,3), (2,4), (3,3), (4,4), (5,5)\}$ dans $\{1, 2, 3, 4, 5\}$.
3. Dans $\mathcal{P}(E)$: $(A, B) \in \mathcal{R} \Leftrightarrow |A| \leq |B|$.
4. $<$ dans l'ensemble des nombres rationnels $\mathbb{Q}$.

Quel est le type de ces relations : ordre, préordre (pas antisymétrique), ordre strict (un ordre privé de la diagonale) ? Dans le cas où ces relations sont des ordres, donnez des exemples d'éléments comparables, incomparables (x est comparable à y si $(x, y) \in \mathcal{R}$ ou $(y, x) \in \mathcal{R}$). Dans le cas où ces relations sont des ordres finis, explicitez la relation de *couverture* et représentez le diagramme de Hasse de ces ordres.

Exercice 25 – Relations d'ordre

Identifier les minimum, maximum, éléments minimaux, maximaux, bornes inf et sup, s'ils existent de la relation d'ordre codée par le diagramme de Hasse suivant :



Exercice 26 – Relations d'ordre Soit E un ensemble ordonné par $\mathcal{R}$. E n'est pas nécessairement fini et $\mathcal{R}$ n'est pas nécessairement total. A étant une partie de E, que penser des affirmations suivantes ?

1. Si x est maximum de A, alors x est maximal de A.
2. Si x est maximal de A, alors x est maximum de A.
3. Si A est fini, et si x est l'unique élément maximal de A alors x est maximum de A.
4. Si A est une chaîne non vide de E alors A admet au plus un élément maximal.
5. Si A est une chaîne non vide de E alors A possède un minimum.

Université de Montpellier 2017/2018
HMIN329 – Complexité et algorithmes Durée: 4.5h

Complexité
TD – Séance n 2

Exercice 1 – Complexité

Soit A et B, deux algorithmes pour résoudre un même problème. La complexité de A est en $\theta(n^2)$ et son exécution avec $n = 100$ dure $1s$. La complexité de B est en $\theta(n \log n)$ et son exécution avec $n = 100$ dure $10s$.

1. L'application prévoit une valeur de n égale à 1000. Quel algorithme faut-il à priori choisir ?
2. Même question si n égale à 10000.

Exercice 2 – Complexité

Soit un algorithme en $\theta(n^2)$. Un ordinateur X permet de traiter en 1 mn des problèmes de taille n_0. Quelle est la taille des problèmes que l'on pourra traiter en 1 mn avec un ordinateur 100 fois plus rapide? Même chose pour $\theta(2^n)$.

Exercice 3 – Complexité

Lesquelles des assertions suivantes sont vraies? Prouvez vos réponses.

1. $1000 \in O(1)$;
2. $n^2 \in O(n^3)$;
3. $n^3 \in O(n^2)$;
4. $2^{n+1} \in O(2^n)$;
5. $(n+1)^2 \in O(n^2)$;
6. $n^3 + 3n^2 + n + 1996 \in O(n^3)$;
7. $n^2 * n^3 \in O(n^3)$;
8. $2^{2n} \in O(2^n)$;
9. $\frac{1}{2}n^2 - 3n \in \theta(n^2)$.

Exercice 4 – Complexité

Comparer deux à deux les fonctions suivantes :

1. $f_1(n) = n^2 + 100$
2. Soit f_2 défini de la manière suivante :

$$\begin{aligned} f_2(n) &= n \text{ si } n \text{ est impair} \\ &= n^3 \text{ si } n \text{ est pair} \end{aligned}$$

3. Soit f_3 défini de la manière suivante :

$$\begin{aligned} f_3(n) &= n \text{ si } n < 100 \\ &= n^3 \text{ si } n \geq 100 \end{aligned}$$

Exercice 5 – Complexité

1. Soient f_1 et f_2 deux fonctions telles que $f_1 = \theta(g)$ et $f_2 = \theta(g)$ et $f_1 \geq f_2$. A-t'on $f_1 - f_2 = \theta(g)$?

Exercice 6 – Complexité

1. Démontrer que $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ et $\sum_{i=1}^{n} i^2 = \frac{(2n+1)(n+1)n}{6}$.
2. Quelle est la complexité de la boucle suivante :

```
for i:=1 to (n-1) do
    for j:=(i+1) to n do
       for k:=1 to j do
        {instruction}
```

Exercice 7 – Complexité

Montrer que $O(f + g) = O(\max(f, g))$.

Exercice 8 – Complexité

Regrouper en classes d'équivalence pour θ les fonctions suivantes. Trier les classes d'équivalence pour O.

$n,\ 2^n,\ n \log n, n - n^3 + 7n^5, n^2, \log_2 n, n^3, \sqrt(n) + \log_2 n, \log_2 n^2, n!, \log n$.

Exercice 9 – Programmes récursifs

Analyser la complexité des différentes programmes en fonction du nombre d'appels récursifs effectués :

```
Rec1(n)
si n<2 alors retourner(1)
       sinon retourner(Rec1(n-1)+2)

Rec2(n)
si n<2 alors retourner(1)
       sinon retourner(Rec2(n div 2)+2)

Rec3(n)
si n<2 alors retourner(1)
       sinon retourner(Rec3(n-1)*(Rec3(n-1)+2))

Rec4(n)
si n<2 alors retourner(1)
       sinon retourner((Rec4(n div 2)+2)*(Rec4( n div 2)+3))
```

Exercice 10 – Calculs récursifs

Calculer x^n, pour des valeurs entières et positives de n, de plusieurs possibilité en utilisant les définitions récursives suivantes (qu'il faudra compléter) et évaluer l'ordre de grandeur de la complexité de chacune des méthodes :

1. $x^n = x * x^{n-1}$
2. $x^n = x^{ndiv2} * x^{ndiv2} * x^{nmod2}$
3. Même définition que 2 mais en utilisant une variable intermédiaire pour stocker x^{ndiv2}.

Exercice 11 – Calculs récursifs

On rapelle la définition des nombres de Fibonacci :

$$f_0 = 0,\ f_1 = 1,\ \forall n \geq 2\ f_n = f_{n-1} + f_{n-2}$$

Programmer la fonction f_n :

1. En utilisant la définition récursive (évaluer sa complexité).
2. En utilisant une version itérative efficace (évaluer sa complexité).
3. Montrer que si $n \geq 1$ alors
   - $f_{2n} = f_n^2 + 2 * f_n * f_{n-1}$
   - $f_{2n+1} = f_n^2 + f_{n+1}^2$

   Evaluer approximativement l'ordre de grandeur de la complexité d'un programme pour calculer f_n qui utilise directement cette nouvelle définition récursive.

   En utilisant des variables intermédiaires améliorer le programme précédent et calculer sa nouvelle complexité.
4. Montrer que

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

   En déduire un programme en $\theta(\log n)$ pour calculer f_n.

Université de Montpellier 2017/2018
HMIN329 – Complexité et algorithmes Durée: 3h

Tris
TD – Séance n 3

Exercice 1 – Tri bulle

Soit l'algorithme suivant qui trie un tableau t :

```
for i:=1 to n-1 do
    for j:=n downto i+1 do
     if t[j] < t[j-1] then echanger (t[j],t[j-1])
```

1. Quel est le nombre de comparaisons dans le pire des cas et en moyenne ?
2. Même question pour le nombre d'échanges.
3. Modifier l'algorithme pour trouver le k^{ieme} plus petit élément du tableau t. Quelle est la complexité de cet algorithme ?

Exercice 2 – Tri par sélection

Considérons l'algorithme qui consiste pour trier les éléments dans les cases de 1 à n ($n > 1$) à cherche l'indice, max du plus grand élément dans les cases de 1 à n et échanger les cases d'indice max et n, puis trier les éléments dans les cases de 1 à $(n-1)$.

1. Ecrire l'algorithme
2. Quel est le nombre de comparaisons dans le pire des cas et en moyenne ?
3. Même question pour le nombre d'échanges.
4. Quel est le nombre d'appels récursifs ?

Exercice 3 – Tri fusion

Nous souhaitons étudier le tri fusion.

1. Rappeler le principe.
2. Appliquer le sur le tableau suivant :

20	5	7	12	21	33	1	6

3. Calculer le nombre maximum de comparaisons. Pour simplifier, on supposera que n est une puissance de 2.

Exercice 4 – Tri rapide

1. Soit T un tableau de n entiers distincts. Donner un algorithme efficace qui déplace les entiers de T de telle façon que les cases de plus petits indices contiennent les entiers inférieurs à l'entier qui était intialement en $T[1]$ (cet entier sera appelé pivot dans la suite), et les cases de plus grands indices les entiers plus grands que le pivot. Plus précisément, s'il y a dans T, $(p-1)$ entiers inférieurs au pivot alors les cases de 1 à $(p-1)$ contiennent ces entiers, la case p contient le pivot, et les cases de $(p+1)$ à n conteinnent les entiers plus grand que le pivot.
2. En déduire un algorithme de tri utilisant deux appels récursifs et analyser cet algorithme au pire et en moyenne. Pour le calcul de la moyenne nous rappelons que $moy_A(n) = \sum_{d \in D_n} p(d) \times cout_A(d)$ avec $p(d)$ la probabilité que l'on ait la donnée d en entrée de l'algorithme et D_n est l'ensemble des données de taille n.

   (a) Quel est la probabilité que le pivot se trouve à la case k ?
   (b) Quels est la conséquence sur la taille des tableaux dans les deux appels récursifs ?
   (c) En déduire une formule de récurrence pour la moyenne ?
3. quelle est la taille maximum de la pile nécessaire pour gérer les appels récursifs de cet algorithme ?
4. Réécrire l'algorithme avec un seul appel récursif et une boucle **tant que** en décursifiant l'appel récursif terminal.
5. Modifier l'algorithme proposé en 4) pour que l'appel récursif porte sur la partie du tableau contenant le moins d'entiers et montrer que la pile nécessaire est alors de hauteur logarithmique au pire.
6. Modifier l'algorithme de tri rapide de façon à avoir le k^{ieme} plus petit élément du tableau.
7. **Subsidiaire** Montrer que l'algorithme précédent est linéaire (c'est à dire en $\theta(n)$) en moyenne).

Exercice 5 – Arbres binaires : Application au tri par tas

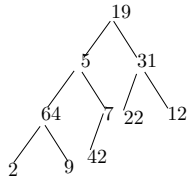


FIG. 1 – Arbre binaire

1. Un arbre parfait est un arbre binaire dont tous les niveaux sont complétement remplis sauf éventuellement le dernier niveau, et dans ce cas les sommets (feuilles) du dernier niveau sont groupés le plus à gauche possible.
   (a) Comment peut-on représenter efficacement un arbre binaire parfait par un tableau et un entier (représentant la taille en nombre d'éléments du tas) ? Donner le tableau reprsésentant l'arbre binaire proposé à la figure 1.
   (b) Donner les fonctions permettant de savoir si un sommet est une feuille, de connaître le père d'un sommet, le fils gauche, le fils droit.
   (c) Calculer la hauteur d'un arbre binaire parfait.
2. Un tas est un arbre binaire parfait tel que chaque élément d'un sommet est inférieur ou égal aux éléments de ses fils.
   (a) Donner les fonctions permettant d'obtenir le minimum d'un tas,
   (b) de supprimer le minimum d'un tas,
   (c) d'ajouter un élément à un tas.
   (d) Evaluer la complexité de chacune de ces fonctions
3. Donnez une fonction qui structure un tableau quelconque en tas en ajoutant successivement les éléments au tas déjà construit (initialement le tas est vide, la partie gauche du tableau est strucutrée en tas, les éléments du tableau sont ajoutés un à un de la gauche vers la droite). Evaluer la complexité de cette construction.

4. Même question mais en construisant le tas de la droite vers la gauche. On considère que les feuilles sont des tas et on fusionne les tas en ajoutant les sommets internes. Evaluer la complexité de cette construction.
5. En déduire un algorithme de tri par ordre décroissant d'un tableau (tri par tas). Evaluer sa complexité.
6. Adapter-le pour trouver le k^{ieme} plus petit élément d'un tableau. Evaluer la complexité de recherche de cet élément.
7. Donner un algorithme qui structure un tableau en tas en $O(n)$.

Université de Montpellier 2017/2018
HMIN329 – Complexité et algorithmes Durée: 4.5h

Graphes
TD – Séance n 4

Exercice 1 – Graphe Combien y-a-t'il de graphes simples non isomorphes avec 4 sommets ? Dessiner chacun de ces graphes.

Exercice 2 – Graphe

Calculez les paramètres n (nombre de sommets), m (nombre d'arcs ou d'arêtes), δ (degré min), Δ (degré max), D (diamètre du graphe) des graphes suivants : B_d (les arbres binomiaux de dim. d), C_n (cycle à n sommets), K_n (graphe complet à n sommets), $GR_{p\times q}$ (grille $p \times q$), $TR_{p\times q}$ (tore $p \times q$), H_d (hypercube de dim. d). Dessinez B_3, K_5, $GR_{4\times4}$, $TR_{4\times4}$, H_2, H_3, H_4.

Exercice 3 – Graphe

Soit $G(V, E)$ un graphe d'ordre n. Soient $d_1, d_2, \ldots, d_n$ les degrés du graphe. Montrer que $\sum_{i=1}^{n} d_i = 2|E|$.

Exercice 4 – Graphe Montrer que dans un graphe il y a toujours un nombre pair de sommets de degré impair.

Exercice 5 – Graphe Quel est le nombre d'arêtes d'un graphe $B(V_1, V_2, E)$ biparti complet où $|V_1| = n_1$ et $|V_2| = n_2$?

Exercice 6 – Graphe Soit $G(X_1, X_2, E)$ un graphe simple biparti d'ordre n. Montrer que $|E| \leq n^2/4$.

Exercice 7 – Graphe

Les nombres $\delta(G)$ et $\Delta(G)$ représentent respectivement les degrés minimum et maximum d'un graphe $G(X, E)$, où $n = |X|$ et $m = |E|$. Montrer que $\delta(G) \leq 2m/n \leq \Delta(G)$.

Exercice 8 – Graphe

Montrer que si un graphe biparti $G(X_1, X_2, E)$ est k-régulier ($k > 0$), alors $|X_1| = |X_2|$.

Exercice 9 – Graphe

Montrer que tout graphe simple possède au moins deux sommets de même degré.

Exercice 10 – Graphe Un n-cube (ou hypercube de dimension n) est un graphe dont les sommets représentent les éléments de $\{0, 1\}^n$ et deux sommets sont adjacents si et seulement si les n-uplets correspondants diffèrent en exactement une composante. Montrer que :

1. Le n-cube possède 2^n sommets
2. Le n-cube est n-régulier
3. Le nombre d'arêtes est $n2^{n-1}$
4. Représenter le 1-cube, le 2-cube et le 3-cube.

Exercice 11 – Graphe Montrer que tout arbre d'ordre n a au moins 2 sommets pendants (un sommet pendant est un sommet de degré 1).

Exercice 12 – Graphe On note $E(G)$ le nombre d'arêtes du graphe G. Montrer que

1. $\forall n, n' \in \mathbb{N}$, $E(K_n) + E(K_{n'}) < E(K_{n+n'})$.
2. Si G est un graphe simple avec n sommets et p composantes connexes, donner une borne supérieure sur le nombre d'arêtes dans G.

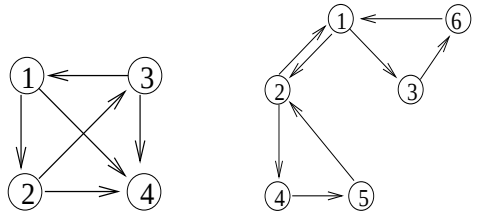
Exercice 13 – Graphe Montrer qu'un graphe simple avec n sommets et plus de $\frac{1}{2}(n-1)(n-2)$ arêtes est connexe.

Exercice 14 – Graphe Soit G un graphe alors les trois assertions suivantes sont équivalentes :

1. $G = (X, Y, E)$ est un graphe biparti.
2. G n'a pas de cycle élémentaire impair.
3. G n'a pas de cycle impair.

Exercice 15 – Graphe Prouver qu'un graphe G de degré minimum $\delta \geq 2$ contient au moins un chaîne élémentaire (chaîne qui ne passe une et une fois par les sommets) de longueur (nombres d'arêtes) supérieure ou égale à δ et un cycle élémentaire de longueur supérieur ou egale à $\delta + 1$.

Exercice 16 – Graphe Soient les graphes $G_1(X_1, A_1)$ et $G_2(X_2, A_2)$ suivants :



G_1 G_2

1. Donner $\Gamma^+(x)$, $\Gamma^-(x)$, $\Gamma(x)$, $d^+(x)$, $d^-(x)$ et $d(x)$, $\forall x \in X_2$.
2. Donner les matrice d'adjacence de G_1 et G_2.
3. Étant donné un graphe orienté $G(X, A)$ ayant n sommets et m arcs, sa matrice d'incidence est une matrice $n \times m$, notée $P = (p_{ie})$, telle que $p_{ie} = 1$ ssi le sommet i est origine de l'arc e, $p_{ie} = -1$ ssi le sommet i est extrémité de l'arc e, et $p_{ie} = 0$ sinon. Donner la matrice d'incidence des graphes G_1 et G_2.
4. Représenter les deux graphes par leurs listes d'adjacence (les sommets d'une liste d'adjacence sont rangés consécutivement dans un tableau).

Exercice 17 – Représentation d'un graphe par une matrice d'adjacence Le but de cet exercice est de voir les propriétés lorsqu'un graphe est donné par une matrice adjacence. On représente un graphe orienté par sa matrice d'adjacence (n, n).

1. Ecrire un algorithme pour déterminer si le graphe est sans boucle.
2. Ecrire un algorithme pour faire afficher tous les successeurs d'un sommet donné.
3. Ecrire un algorithme pour faire afficher tous les prédédecesseurs d'un sommet donné.
4. Soit G un graphe non orienté. Quel est la particularité de la matrice d'adjacence M associé au graphe G ? Quel implication concernant le stockage des données.
5. Soit G un graphe orienté. Soit M sa matrice d'adjacence. Que représente M^t où M^t désigne la matrice transposée. Appliquer les graphes G_1 et G_2 définiis précédemment.

Exercice 18 – Notion de base Soit la matrice binaire ou matrice d'adjacence, M associé au graphe $G = (S, U)$ orienté.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

1. Tracer le graphe représentatif de cette matrice.
2. Donner la matrice d'incidence sommets-arcs de ce graphe
3. Calculer M^2, M^3, M^4. Que pouvez-vous en dire ?
4. Calculer : $A = I + M + M^2 + M^3 + M^4$. Interpréter A.
5. Appliquer l'algorithme de Roy Warshall. Que constatez-vous ?

Exercice 19 – Notion de base bis Dans le graphe G, représenté par sa matrice d'incidence sommets sommets.

	A	B	C	D	E	F
A	0	1	0	0	0	1
B	0	1	1	0	1	0
C	0	0	0	1	0	0
D	0	1	1	0	1	0
E	0	1	0	0	1	1
F	1	0	1	0	0	0

1. Enumérer la liste des successeurs et des prédécesseurs de tous les sommets.
2. Donner les demi-degré intérieurs et extérieurs de chaque sommet.
3. Donner un exemple de chemin simple mais non élémentaire.
4. Existe-il un circuit hamiltonien dans G ?
5. Tracer le graphe non orienté déduit de G.
6. G est-il connexe ? Fortement connexe ?

Exercice 20 – Fermeture Transitive d'un graphe

Soit C la matrice d'adjacence de G. On veut calculer la matrice d'adjacence C^* de G^* la fermeture transitive du graphe G. Supposons que $C_k[i,j]$ représente l'existence d'une chaîne de i à j passant par des sommets inférieurs ou égaux à k.

1. Quelles sont les conditions d'existence d'une chaîne de i à j passant par des sommets inférieurs ou égaux à k ?
2. Ecrire un algorithme qui calcule la fermeture transitive d'une graphe G représenté par une matrice d'adjacence.
3. Quelle est la complexité de cet algorithme dans un graphe orienté ?
4. Donner un exemple de matrice M pour laquelle n éléments seulement sont égaux à 1 et telle que tout élément de M^* est égal à 1.
5. Donner un exemple de graphe orienté à matrice M dans laquelle tous les éléments, sauf $n-1$ sont égaux à 1, mais tel qu'il existe au moins un élément de M^* différent de 1.
6. Quel est la fermeture transitive d'un graphe non orienté ?

Exercice 21 – Composantes fortement connexes

	A	B	C	D	E	F	G	H	J	K	L
A	0	0	0	1	0	0	0	0	0	0	0
B	1	0	0	0	1	0	0	0	0	0	0
C	0	1	0	0	0	1	0	0	0	0	0
D	0	0	1	0	0	0	1	0	0	0	0
E	0	0	0	0	1	0	0	0	1	0	0
F	0	0	0	0	0	0	1	0	1	1	0
G	0	0	0	0	0	1	0	1	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	1	0
K	0	0	0	0	0	0	0	0	0	0	1
L	0	0	0	0	0	0	0	0	1	0	0

1. Décomposer, le graphe défini par sa matrice d'adjacence, en composantes fortement connexes et les ordonner en utilisant la fermeture transitive et en utilisant un parcours en profondeur.

On considère un graphe simple $G = (S, A)$ orienté et sans circuit. On appelle numérotation topologique du graphe G une bijection $r : S \to \{1, \ldots, n\}$ qui possède la propriété suivante : si $(x, y) \in A$ alors $r(x) < r(y)$.

1. Soit x un sommet sans successeur de G. On note $G\backslash\{x\}$ le"sous-graphe de G construit en retirant le sommet x et tous les arcs incidents à x.

   Montrer que si r' est une numérotation topologique de $G\backslash\{x\}$ alors la numérotation définie par :

   $$r(x) = n \text{ et } \forall y \in S,\ y \neq x\ r(x) = r'(y)$$

   est une numérotation topologique de G. En déduire qu'un graphe sans circuit admet une numérotation topologique.

   Nous supposons maintenant que les sommets de G sont numérotés de manière topologique et l'on identifie chaque sommet avec son numéro : un sommet x tel que $r(x) = i$ sera appelé sommet i. On notera $\mathcal{C}_k(i,j)$ l'ensemble des chemins du sommet i au sommet j dont tous les sommets intérieurs (s'ils existent) sont des numéros inférieurs ou égal à k. $\mathcal{C}_0(i,j)$ est l'ensemble des chemins d'intérieur vide de i à j. M^k désignera la matrice booléenne $n \times n$ définie par : $M^k[i,j] = 1$ si $\mathcal{C}_k(i,j) \neq \varnothing$; $M^k[i,j] = 0$ sinon. Par convention, on pose $M^[i,j] = 1$ pour tout i.
2. Montrer que pour $k \in \{1, \ldots, n\}$
   (a) Si $j < i$, $\mathcal{C}_k(i,j) = \varnothing$
   (b) Si $k < i$, $\mathcal{C}_k(i,j) = \mathcal{C}_0(i,j)$
   (c) Si $k > j$, $\mathcal{C}_k(i,j) = \mathcal{C}_{j-1}(i,j)$.
3. En déduire que pour tout $k > 0$, $M^k[i,j] = M^{k-1}[i,j]$ si $I \geq k$ ou $j \leq k$.
4. Indiquer un algorithme de calcul de fermeture transitive de G inspiré de l'algorithme de Roy-Warshall qui tienne compte de la propriété de la question précédente et prouver sa validité.
5. Calculer la complexité de l'algorithme précédent.

Université de Montpellier 2017/2018
HMIN329 – Complexité et algorithmes Durée: 3h

Arbres
TD – Séance n 5

Exercice 1 – Propriété des arbres

Soit T_1 et T_2 deux arbres de recouvrement d'un graphe connexe G. Montrer que T_1 peut être transformé en T_2 par une séquence d'arbres intermédiaires.

Exercice 2 – Propriété des arbres

Soit $\{v_1, v_2, \ldots, v_n\}$ des points fixés et $\{d_1, d_2, \ldots, d_n\}$ des nombres entiers donnés tels que $\sum_{i=1}^{n} d_i = 2n-2,\ d_i \geq 1$. Montrer que le nombre d'arbres sur l'ensemble $\{v_1, v_2, \ldots, v_n\}$ où le sommet v_i admet pour degré $d_i,\ i = 1, \ldots, n$ est $\frac{(n-2)!}{(d_1-1)!\ldots(d_n-1)!}$

Exercice 3 – Propriété des arbres

Montrer que le nombre de tous les arbres sur n points est n^{n-2} (formule de Cayley).

Exercice 4 – Propriété des arbres

Soient $1 \leq d_1 \leq d_2 \leq \ldots \leq d_n$ des entiers. Nous allons prouver qu'il existe un arbre avec des degrés $d_1, d_2, \ldots, d_n$ si et seulement si $d_1 + d_2 + \ldots + d_n = 2n-2$.

1. Montrer que si il existe un arbre avec des degrés $d_1, d_2, \ldots, d_n$ alors $d_1+d_2+\ldots+d_n = 2n-2$.
2. On suppose maintenant que $d_1 + d_2 + \ldots + d_n = 2n-2$. La preuve est par récurrence sur n. Que pensez-vous du cas $n = 2$?
3. On suppose $n \geq 3$. Quelle est forcément la valeur de d_1 ? Montrer que $d_n > 1$.
4. Montrer qu'il existe un arbre avec les degrés $d_2, d_3, \ldots, d_{n-1}, d_n - 1$.
5. Conclure.

Exercice 5 – Arbre couvrant

Soit G le graphe de la figure 1.

1. Indiquer dans quel ordre les procédures d'exploration en largeur et en profondeur visitent les sommets de G en partant du sommet a.
2. Donner toutes les possibilités et les arbres couvrants associés.

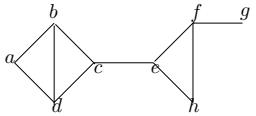


FIG. 1 – Exemple

Exercice 6 – Propriété des arbres

Prouver que si tous les poids sur les arêtes sont différents, il y a unicité de l'arbre couvrant de poids minimum.

Exercice 7 – Propriétés générales

Soit $K_n(V)$ un graphe complet sur n sommets $V = \{x_1, x_2, \ldots, x_n\}$, où les arêtes sont munies d'un poids w vérifiant $w(xy) + w(yz) \geq w(xz)$ pour tous $x, y, z \in V$. Le but de cet exercice est de montrer qu'un cycle hamiltonien de poids minimum dans K_n a un poids au plus égal à $2w(T)$, où T est un arbre couvrant de K_n de poids minimum. On procédera par récurrence sur n le nombre de sommets.

1. Montrer que le résultat est vrai lorsque $n = 3$.
2. Soit T un arbre couvrant optimal de K_n et x_n un sommet pendant de T. Soit $T' = T - x_n$ l'arbre obtenu à partir de T en enlevant le sommet x_n. Montrer que T' est un arbre couvrant optimal de $K_{n-1}(V \setminus \{x_n\})$.
3. Soit c' un cycle optimal hamiltonien de K_{n-1}, et soit $c = c' - [a,b] \cup \{[a,x_n],[b,x_n]\}$, a et b étant deux sommets consécutifs du cycle c' (on peut supposer que l'arête $[a,x_n]$ est de poids la plus faible). Montrer que le cycle c vérifie $w(c) \leq 2w(T)$. Conclure.

Exercice 8 – Propriété des arbres

1. Soit e une arête de poids minimal dans un graphe G. Montrer que e appartient à un arbre couvrant minimal de G.
2. Est-ce encore vrai pour deux arêtes, c'est à dire s'il existe deux arêtes $e,\ f$ telles que $w(e) = w(f)$ et pour tout $u \in E,\ u \neq e$ et $u \neq f,\ w(e) < w(u)$?
3. Est-ce que cela est vraie pour trois arêtes ?

Exercice 9 – Applications d'algorithmes

Exécuter les algorithmes de Prim et de Kruskal sur les graphes donnés par la figure 2 et 3.

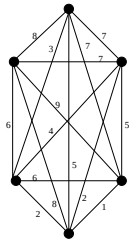


FIG. 2 –

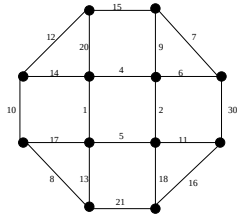


FIG. 3 –

Exercice 10 – Algorithme de Sollin

Soit un graphe $G = (N, A)$ les sommets du graphe.

Algorithm 1 Algorithme de Sollin

for $i \in N$ **do**
  $N_i := \{i\}$
end for
while $|T^*| < (n-1)$ **do**
  for chaque arbre N_k **do**
    $Plusprochevoisin(N_k, i_k, j_k)$
  end for
  for chaque arbre N_k **do**
    if les sommets i_k et j_k appartiennent à deux arbres différents **then**
      $Fusion(i_k, j_k)$;
      $T^* = t^* \cup \{(i_k, j_k)\}$
    end if
  end for
end while

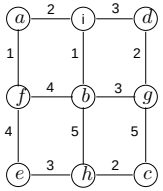


FIG. 4 – Graphe

- La fonction $plusprochevoisin(N_k, i_k, j_k)$ prend en entrée un arbre couvrant sur les sommets de N_k et détermine l'arête de poids minimale parmi toutes les arêtes de N_k ($c_{i_k j_k} = \min\{c_{ij} : (i,j) \in A, i \in N_k \text{ et } j \notin N_k\}$)
- La fonction $Fusion(i_k, j_k)$ prend en entrée deux sommets i_k et j_k, et si ces deux sommets appartiennent à deux arbres différents, on fusionne les deux arbres.

1. Expliquer pourquoi il termine et pourquoi il donne un arbre couvrant. Quel théorème l'algorithme vérifie.
2. Expliquer pourquoi il donne bien un arbre couvrant de poids minimum. Pour cela considérer un autre arbre B et montrer que forcément le poids de B est supérieur ou égal à A.
3. Donner sa complexité. Pour cela donner la complexité des deux fonctions *plusproche* et $fusion$.
4. Appliquer cet algorithme sur la figure 4.

Exercice 11 – Algorithme

Soit G un graphe non orienté connexe à n sommets et p arêtes, dont les arêtes sont valuées par des nombres. Vous employez un jeune programmeur qui, pour construire un arbre recouvrant de poids minimal de G, vous propose l'algorithme suivant :


```
Soit A le graphe dont les sommets sont ceux de G,
et qui ne contient aucune arête ;
soit s un sommet de A ;
TANT QUE l'on modifie quelque chose
  parmi les arêtes de G d'extrémités
 qui ne sont pas dans A et ne créent pas de cycle dans A,
choisir une arêtes s-s'de poids minimal ;
ajouter l'arête s-s'à A;
s:=s'
FIN TANT QUE
```

1. Montrez que l'algorithme se termine et donner sa complexité.
2. Trouver un graphe pour lequel l'algorithme propose un graphe non connexe.
3. Votre employé propose la correction suivante : si le graphe A obtenu à la fin de l'algorithme n'est pas connexe. on recommence l'exécution de la boucle TANT QUE à partir d'une feuille de A (c'est-à-dire un sommet extrémité d'au plus une arête de A). Cette correction vous satisfait-elle ? Pourquoi ?

Exercice 12 – Algorithme

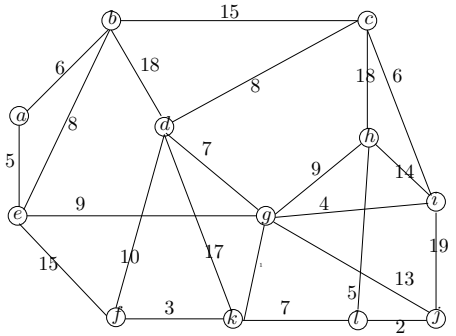


FIG. 5 –

Déterminer un arbre couvrant de poids minimum du graphe donné par la figure 5 :

1. d'abord à l'aide de l'algorithme de Kruskal,
2. puis à l'aide de l'algorithme de Prim.

Il ne sera pas nécessaire de dessiner à chaque étape l'arbre couvrant ou la forêt en cours de construction, mais on indiquera à chaque étape quelle arête est sélectionnée, et on dessinera l'arbre couvrant de poids minimum obtenu à la fin des algorithmes.

On considère maintenant l'algorithme suivant :
On part de la liste des arcs dans un ordre quelconque $u_1, u_2, \ldots, u_n$.
`Au depart` $T := u_1$ `et` $k = 1$.
`À l'étape` k `faire :`

$T := T + u_k$
Si T ne contient pas de cycle, continuer (passer à l'étape $k+1$).
Si T contient un cycle C, et si v est un arc de poids maximum dans C, faire $T := T - v$ et continuer (passer à l'étape $k+1$).

1. Expliquer brièvement pourquoi cet algorithme termine et donne un arbre couvrant de poids minimum.
2. Appliquer cet algorithme sur la figure 5.
3. Expliquer brièvement comment on peut modifier l'algorithme précédent pour obtenir un arbre couvrant de poids maximum (il n'est pas demandé d'appliquer l'algorithme ainsi modifié sur le graphe).

Exercice 13 – Arbres de recouvrement

Mettre en œuvre l'algorithme de `Prim` lorsque le graphe est représenté par une liste d'adjacence et ensuite lorsque le graphe est représenté par une matrice d'adjacence.

Exercice 14 – Implémentation de l'algorithme de Kruskal

Soit $G = (X, E)$ un graphe non orienté valué, on notera $\omega(e)$ le poids de l'arête e. L'algorithme consiste à partir d'un graphe T vide qui possède les mêmes sommets que G. A chaque étape on prend l'arête de poids minimum qui n'est pas déjà dans T et qui ne forme pas de cycle lorsque elle est ajoutée à T.

1. Montrer que cet algorithme donne bien un arbre de recouvrement minimum (on supposera pour simplifier que toutes les arêtes ont un poids différent).

   On se propose maintenant de trouver un algorithme efficace pour déterminer si une arête va créer un cycle si on l'ajoute à un graphe. Pour cela on utilise un tableau qui pour chaque sommet contient sa composante connexe, représentée par un des nœuds de la composante connexe. Une arête $\{x, y\}$ crée un cycle si x et y appartiennent à la même composante connexe.
2. Déterminer la complexité pour savoir si une arête crée un cycle et quelle est la durée de mise à jour de la structure lorsque l'on ajoute une arête (il faut faire la fusion des composantes connexes contenant les extrémités)..
3. On structure maintenant les composantes connexes sous forme d'arborescences, la racine étant le représentant de la composante connexe. Que deviennent les complexités de la recherche de la composante connexe d'un sommet et la fusion de deux composantes connexes ?
4. Modifier la structure pour que la composante la plus grosse absorbe la plus petite en cas de fusion. Que deviennent la complexité des fonctions de recherche de composante connexe et de fusion ?
5. Proposer une implémentation de l'algorithme de Kruskal et évaluer sa complexité.

Exercice 15 – Chaine hamiltonienne

Prouver que l'algorithme suivant ne donne pas toujours une chaîne hamiltonienne de poids minimum dans le graphe complet :

1. choisir une arête e_1 telle que $w(e_1)$ soit minimum
2. à toute étape k, choisir une arête e_k telle que le graphe engendré par $e_1, e_2, \ldots, e_k$ soit union disjointe de chaînes, et telle que $w(e_k)$ soit minimum.

Exercice 16 – Le voyageur de commerce

Le tableau suivant donne les distances en km entre cinq villes :

	A	B	C	D	E
A	0	9	7	5	7
B	9	0	9	9	8
C	7	9	0	7	6
D	5	9	7	0	6
E	7	8	6	6	0

Trouver une borne inférieure de la solution du problème du voyageur de commerce en enlevant

1. la ville B,
2. la ville E,

Université de Montpellier 2017/2018
HMIN329 – Complexité et algorithmes Durée: 4.5h

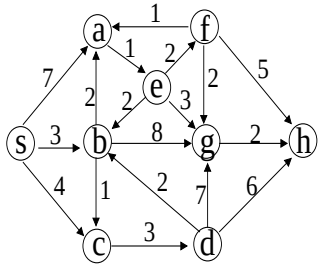
Plus courts chemins
TD – Séance n 6

Exercice 1 – Applications d'algorithmes

Soit P un plus court chemin de s à t dans un graphe G, et x, y deux sommets rencontrés dans cet ordre sur P. Montrer que la portion P_{xy} de P comprise entre x et y est un plus court chemin de x à y.

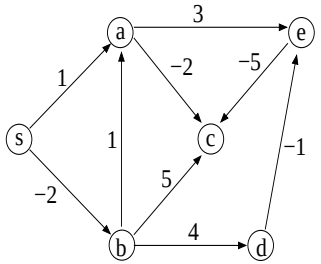
Exercice 2 – Arborescence

On considère le graphe orienté valué défini ci-dessous. Déterminer une arboresence des plus courts chemins depuis le sommet s en utilisant l'algorithme de Dijkstra.



Exercice 3 – Arborescence

On considère le graphe orienté valué défini ci-dessous.



1. Déterminer une arborescence des plus courts chemins depuis le sommet s en utilisant la programmation dynamique.
2. Qu'obtient-on avec l'algorithme de Dijkstra ? Conclure.
3. Une solution consisterait d'augmenter de $|a|$ où a est la valeur minimale sur les arcs et après d'enveler de $-a$. Que pensez-vous de cette solution ?

Exercice 4 – Propriétés des chemins

Est-il vrai que

1. Si tous les arcs ont des poids différents, l'arborescence des plus courts chemins à partir d'un sommet s fixé est unique ?
2. Parmi les n arborescences des plus courts chemins au moins l'un d'eux correspond à un arbre couvrant optimal ?

Exercice 5 – Algorithme

Soit un graphe orienté G, s un sommet de G. On cherche à déterminer le poids minimal des chemins reliant s à tous les autres sommets de G en utilisant l'algorithme glouton suivant : on pose $dist(s) = \infty$ pour tout sommet s, et $dist(s) = 0$. On visite ensuite le sommet s au moyen de la procédure récursive

```
visiter un sommet s
 SI s n'a pas encore été visité ALORS
   soit s' le successeur de s le plus proche de s
   dist(s'):=dist(s)+poids(s,s')
   visiter s'
 FIN SI
```

1. Quelle est la complexité de cet algorithme ?
2. Expliquer pourquoi il est incorrect (il y a plusieurs raisonsà cela ; on illustrera chacun des problèmes par un exemple de graphe pour lequel l'algorithme ne donne pas le résultat cherché).

Exercice 6 – Algorithme de Floyd

L'algorithme de Floyd calcule les plus courts chemins pour tout couple de sommets dans un graphe orienté valué $G = (V, E)$. Pour que ce problème ait une solution le graphe ne doit pas contenir de circuit absorbant. La donnée du problème est un graphe orienté valué représenté par sa matrice D d'adjacence : les entrées $d[i, j]$ de D sont les valuations v_{ij} des arcs ; par convention nous aurons $d[i, j] = \infty$ pour $i \neq j$ si $(i, j) \notin E$ et $d[i, j] = 0$ pour $i = j$. Le résultat de l'algorithme est une matrice Δ pour laquelle l'entrée $\Delta[i, j]$ est la longueur d'un plus court chemin d'origine i et d'extrémité j ; s'il n'existe pas de chemin $(1, \ldots, j)$ dans le graphe alors $\Delta[i, j] = \infty$.

Voici l'algorithme :

Algorithm 1 Algorithme de floyd

```
Soit Δ une matrice
Δ ← D
for k ← 1à n do
  for j ← 1à n do
    for i ← 1à n do
      Δ[i,j] ← min{Δ[i,j], Δ[i,k] + Δ[k,j]}
    end for
  end for
end for
```

1. Donner la complexité de cet algorithme.
2. Appliquer l'algorithme de Floyd pour obtenir les plus courts chemins entre tout couple de sommets du graphe donné par la figure 1

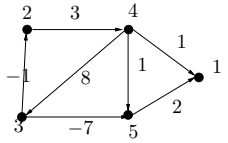


FIG. 1 – Un graphe orienté valué

Exercice 7 – Matrices et plus court chemins

Une structure de données simple pour représenter un graphe est la matrice d'adjacence M. Pour obtenir M, on numérote les sommets du graphe de façon quelconque. $X = \{x_1, \ldots, x_n\}$. M est une matrice carrée $n \times n$ dont les coefficients sont 0 et 1 telle que $M_{i,j} = 1$ si $(x_i, x_j) \in E$ et $M_{i,j} = 0$ sinon. Démontrer le la proposition suivante : **Proposition : Soit M^p la puissance $p-ieme$ de la matrice M, le coéfficient $M^p_{i,j}$ est égal au nombre de chemins de longueur p de G dont l'origine est le sommet x_i et dont l'extrémité est le sommet x_j.**

Exercice 8 – Détection d'un circuit absorbant

Nous allons montrer comment il est possible de détecter la présence de circuits absorbants en utilisant l'algorithme de Floyd.

Afin de rendre compte de circuits abosrbants consistant en une boucle de valuation strictement négative, les éléments $d[i,j]$ de la matrice d'adjacence du graphe sont définis de la façon suivante : $d[i,i] = 0$ s'il n'y a pas de boucle au sommet i ou si $v_{ii} < 0$, $d[i,i] = v_{ii}$. Soit Δ la matrice résultat obtenu en exécutant l'algorithme de Floyd.

1. Montrer que G a un circuit absorbant si et seulement si il existe un sommet i de G tel que $\Delta[i,i] < 0$.

Exercice 9 – Impression de plus courts chemins

Écrire une procédure qui imprime un plus court chemin entre deux sommets i et j, s'il existe, en utilisant la matrice P calculée dans la procédure *Floyd*. Dans un premier temps vous modifierez l'algorithme de Floyd pour y inclure l'instruction nécéssaire à l'affichage du chemin de i à j.

Exercice 10 – Diamètre et excentricité

Soit un graphe $G = (V, E)$. On appelle *excentricité* d'un sommet X, et on la note $e(X)$, la plus grande distance entre le sommet X et les autres sommets du graphe :

$$e(X) = \max_{v \in V} d(X, v)$$

Le diamètre D d'un graphe est alors défini de la façon suivante :

$$D = \max_{v \in V} e(v)$$

A l'aide de l'algorithme de Dijkstra, vous calculerez pour un sommet donné v son excentricité. Vous généraliserez votre algorithme de façonà calculer le diamètre du graphe G. Vous avez toute latitude pour choisir la représentation du graphe.

Exercice 11 – Exploration en profondeur

Utiliser la procédure d'exploration en largeur d'un graphe $G(X, A)$, $|X| = n$, $|A| = m$, pour obtenir en $O(m)$ les plus court chemins en nombre d'arcs d'un sommet s vers tout autre sommet.

Exercice 12 – Plus courts chemins avec des longueurs unitaires

Dans cet exercice nous considérons le problème de déterminer les plus courts chemins d'un sommet sà tous les autres dans le cas où la valuation sur les arcs est unitaire.

1. Donner un algorithme qui donne l'arborescence des plus courts chemins d'un sommet sà tous les autres en $O(m)$.

Exercice 13 – Ordonnancement

Soit $G = (S, A, V)$ un graphe orienté valué, une source $s \in S$ et un sommet puits $p \in S$ (s n'a pas de prédécesseur, p n'a pas de successeur et pour tout sommet x, $x \neq s$, $x \neq p$ il existe un chemin de s à p passant pas x, $\mu = (s \ldots x \ldots p)$).

Une fonction potentiel $\Pi : S \to R$ doit satisfaire aux conditions suivantes :

- $\pi(s) = 0$,
- $\forall (x, y) \in A,\ \pi(x) - \pi(y) \geq v_{xy}$.

La quantité $\Pi(G) = \pi(p)$ est appelée potentiel du graphe G.

1. Montrer qu'une condition nécessaire et suffisante d'existence d'une fonction potentiel est que G ne possède pas de circuit de longueur strictement positive.

   Nous supposons maintenant que cette condition est satisfaite. Soit $\Delta(x)$ la longueur d'un plus long chemin d'origine s et d'extrémité x.
2. Montrer que $\Delta(x)$ définit une fonction potentiel.

   Une fonction potentiel π est dite minimale si $\Pi(G)$ est minimum.
3. Montrer que la fonction potentiel définie par les valeurs $\Delta(x)$ est minimale. Soit $\delta(x)$ la longueur d'un long chemin d'origine x et d'extrémité p.
4. Montrer que $\pi(x) = \Delta(p) - \delta(x)$ est une fonction potentiel minimale.

Exercice 14 – Ordonnancement bis

Cet exercice consiste à montrer comment le calcul d'une fonction potentiel ; définie dans l'exercice précédent, permet de déterminer un calendrier d'exécution des tâches d'un projet.

On considère un ensemble de tâches $\{1, \ldots, n\}$ de durée connnue $p_i \in N$. L'exemple suivant indique l'ensemble des tâches à réaliser pour la construction d'une maison ainsi que leur durée d'exécution en semaine.

tâche	description	durée
1	maçonnerie	7
2	pose de la charpente	3
3	électrticité et eau	8
4	pose de la toiture	1
5	façade	2
6	pose des fenêtres	1
7	plafonds	3
8	aménagement du jardin	1
9	peinture	2
10	emménagement	1

Les tâches reliées par des relations de précédence. Par exemple, il faut avoir fini de poser la charpente avant de commencer la toiture, et donc la tâche 2 précède la tâche 4.

On considère $G = (S, A, V)$ le graphe modélissant les relations de précédence. Il est obtenu de la façon suivante : les tâches sont les sommets et on a un arc $(i, j) \in A$ si la tâche i précède j. La valuation v_{ij} de l'arc (i, j) est la durée p_i de la tâche i.

1. Construisez le graphe de précédence.
2. Montrer que pour les précédences soient cohérentes G doit être sans circuit.

   On suppose que G est sans circuit et qu'il existe un sommet source s et un sommet puits p (s rt p sont des tâches fictives de durée nulle, s précédant toutes les tâches du projet et s succédant à toutes ces tâches).
3. Montrer qu'une fonction potentiel existe pour G.
4. Soient Π une fonction potentiel et $\mu = (i_1, \ldots, i_k)$ un chemin de G. Montrer que $\pi(i_k) \geq \pi(i_1)$. On cherche à déterminer un ordonnancement des tâches, c'està dire à calculer pour chasue tâche i du projet sa date de début d'exécution t_i de sorte que les contraintes de précédence soient satisfaites c'està dire $\forall (i, j) \in A,\ t_j \geq t_i + p_i$. Par convention, nous pososns $t_s = 0$ comme date de début du projet.
5. Montrer que les valeurs t_i définissent une fonction potentiel de G. On définit la durée d'un ordonnancement, notée D, comme la plus grande date de fin d'exécution d'une tâche : $D = \max_{i \in \{1,\ldots,n\}}(t_i + p_i)$
6. Montrer que $D = t_p$.
7. Montrer qu'un ordonnancement de durée D minimale correspondà une fonction potentiel minimale de G.

   Dans la suite, nous considérons des ordonnancements de durée minimale. La date d'exécution au plus tôt d'une tâche $\theta(i)$ est la date minimale de début de i dans tout ordonnancement : $\theta(i) = \min_{\Pi \in \Pi^*}\{\pi(i)\}$ où Π^* est l'ensemble des fonctions potentiel (des ordonnancements) de G. L'ordonnancement au plus tôt noté Θ est l'ordonnancement dans lequel la date de début de chaque tâche est sa date de début au plus tôt $\theta(i)$.
8. Montrer que l'ordonnancement au plus tôt Θ est obtenu en faisant $\theta(i) = \Delta(i)$ avec $\Delta(i)$ la longueur d'un plus long chemin $(s, \ldots, i)$. Utiliser l'algorithme de Bellman pour calculer Θ dans l'exemple présenté plus haut.

   La date d'exécution au plus tard d'une tâche $\tau(i)$ est la date maximale de début de i dans tout ordonnancement minimal : $\tau(i) = \max_{\Pi \in \Pi^*}\{\pi(i)\}$ où Π^* est l'ensemble des fonctions potentiel minimales de G. L'ordonnancement au plus tard noté T est l'ordonnancement dans lequel la date de début de chaque tâche est sa date de début au plus tard $\tau(i)$.
9. Montrer que l'ordonnancement au plus tard T est obtenu en faisant $\tau(i) = \Delta(p) - \delta(i)$ avec $\delta(i)$ la longueur d'un plus long chemin $(i, \ldots, p)$. Calculer T dans l'exemple présenté plus haut.

   On définit la marge d'une tâche comme étant la différence entre sa date de début au plus tôt et sa date de début au plus tard dans un ordonnancement de durée minimale : $M_i = \tau(i) - \theta(i)$. Une tâche i est dite critique lorsque $M_i = 0$.
10. Déterminer les marges et les tâches critiques pour l'exemple présenté plus haut.
11. Montrer qu'il existe un chemin $\mu = (s, \ldots, p)$. tel que toute tâche i de μ est critique.