



POLYTECHNIQUE
MONTRÉAL

Corrigé
examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Total	

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>		
INF2010 – Structures de données et algorithmes		Tous	20191		
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>		
Tarek Ould Bachir		A-343.14	2452		
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>		
Mercredi	18 avril 2019	2h30	9h30-12h00		
<i>Documentation</i>		<i>Calculatrice</i>			
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Aucune	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.			
<input type="checkbox"/> Toute	<input type="checkbox"/> Toutes				
<input checked="" type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable				
<i>Directives particulières</i>					
Ne posez pas de questions durant l'examen. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.					
Un cahier d'examen vous est fourni à titre de brouillon. Ne remettez pas le cahier d'examen.					
Remettez ce questionnaire avec vos réponses dans les espaces réservés à cet effet.					
Important	Cet examen contient 7 questions sur un total de 18 pages (excluant cette page)				
	La pondération de cet examen est de 40 %				
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux				
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non				

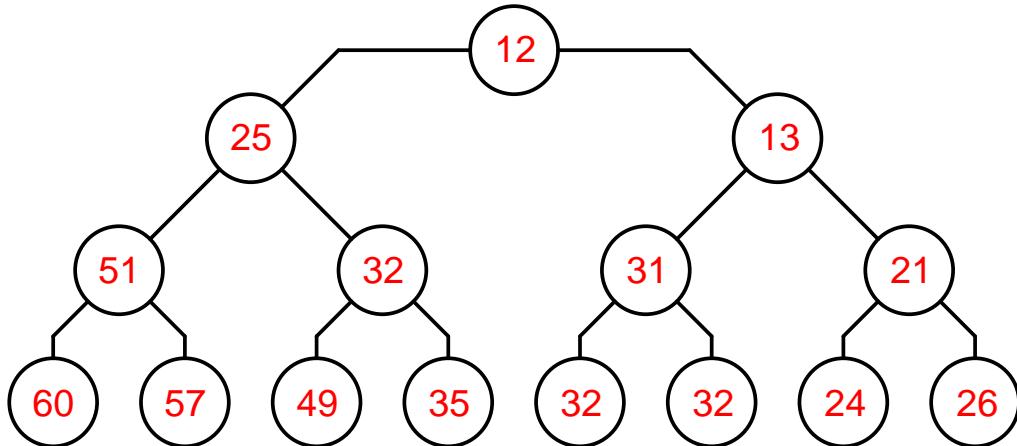
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(18 points)**

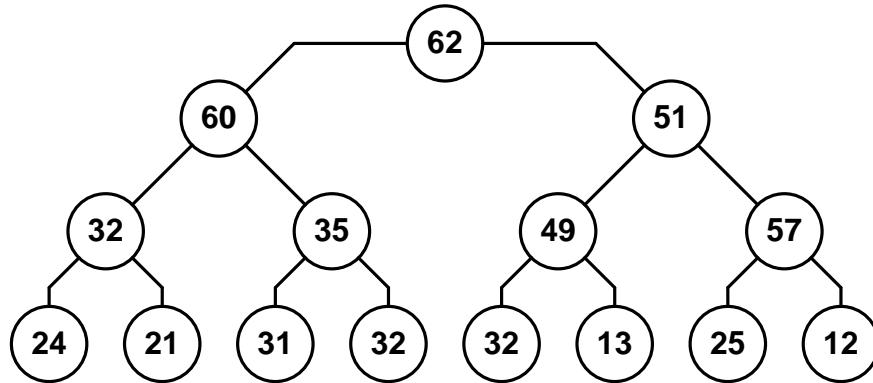
- 1.1) (2 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	12	25	13	51	32	31	21	60	57	49	35	32	32	24	26

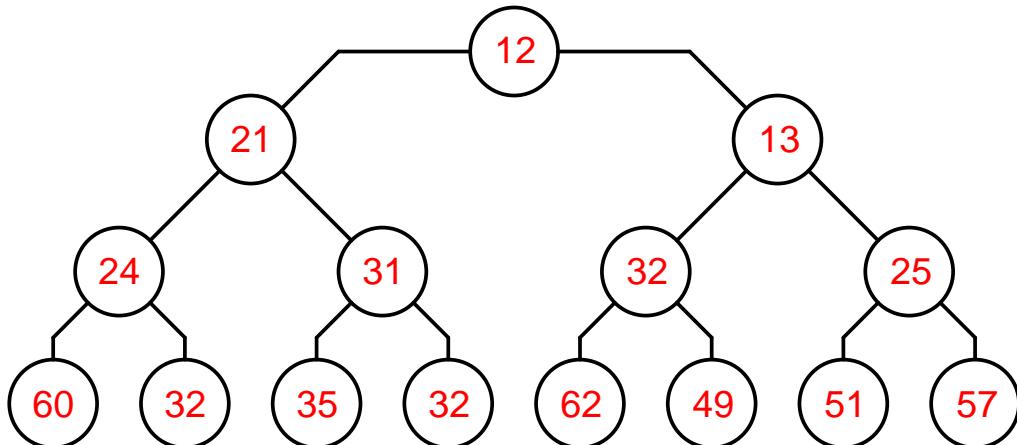
Votre réponse :



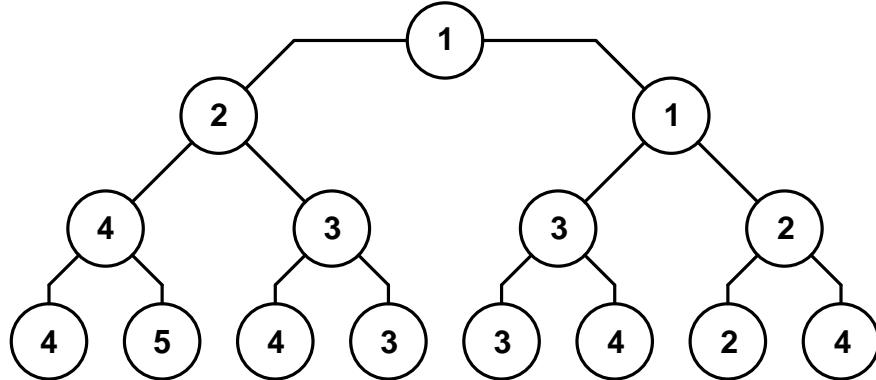
- 1.2) (3 pts) Construisez, selon la technique vue en cours, un monceau MIN à partir de l'arbre binaire suivant :



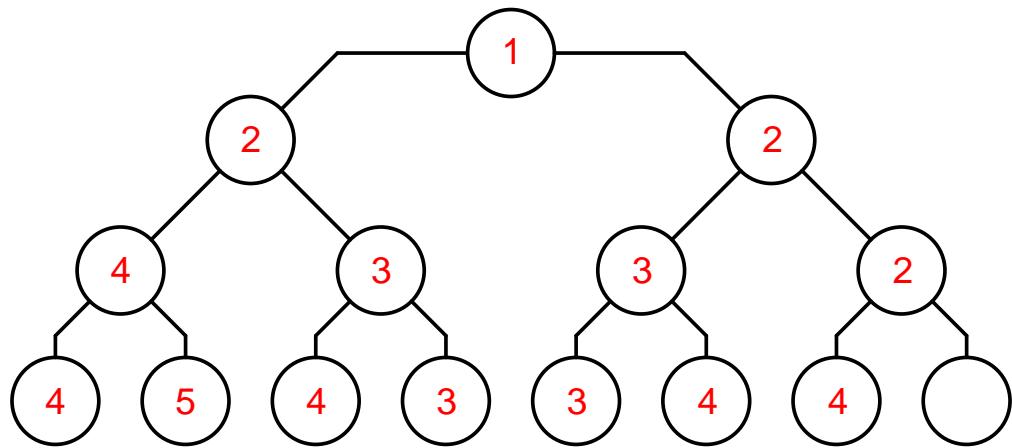
Monceau résultant :



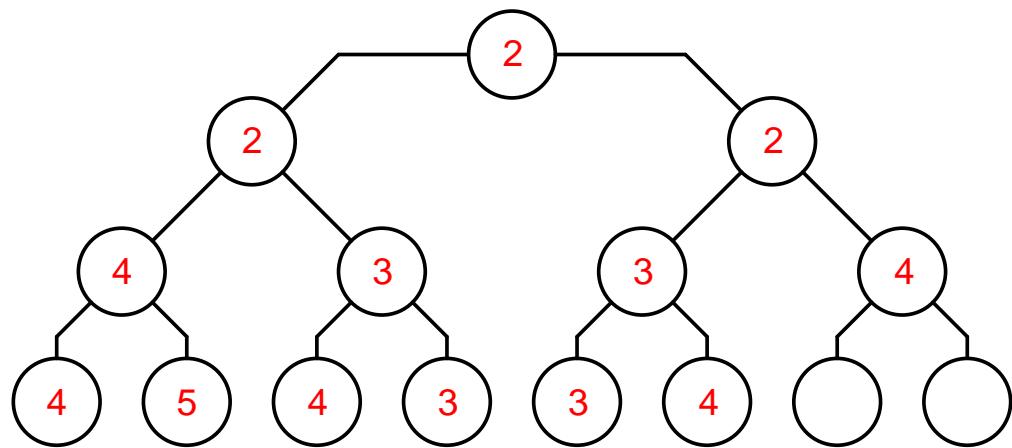
1.3) (4 pts) Dessinez l'état du monceau MIN suite à deux appels consécutifs à `deleteMin()`:



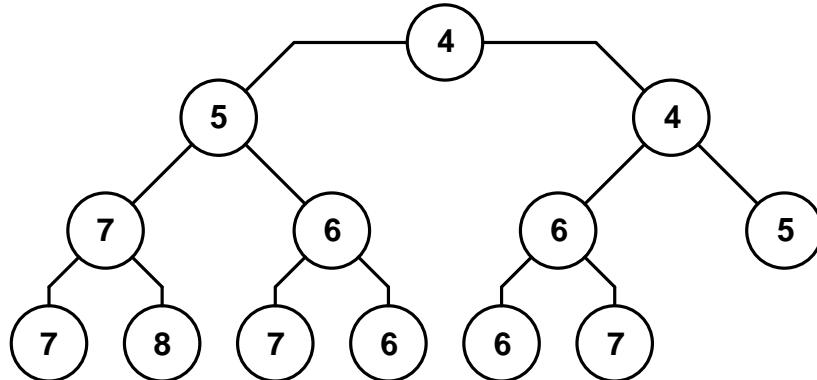
1.3.1) (2 pts) Monceau résultant du premier `deleteMin()`:



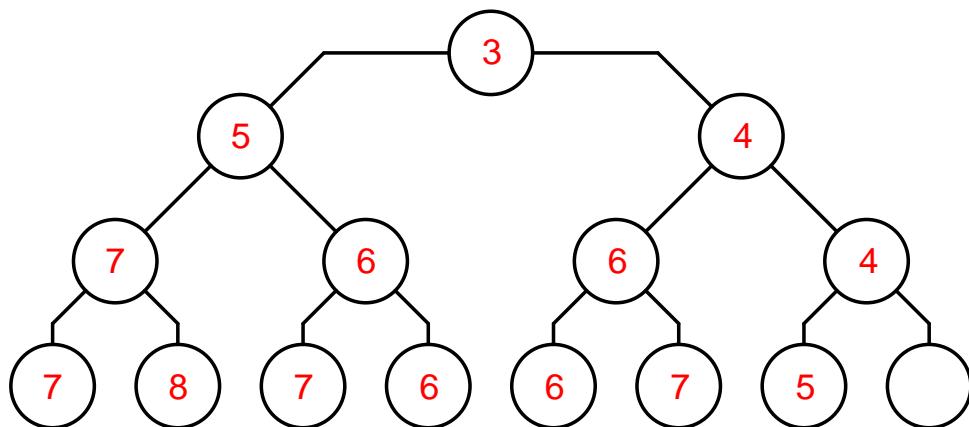
1.3.2) (2 pts) Monceau résultant du second `deleteMin()`:



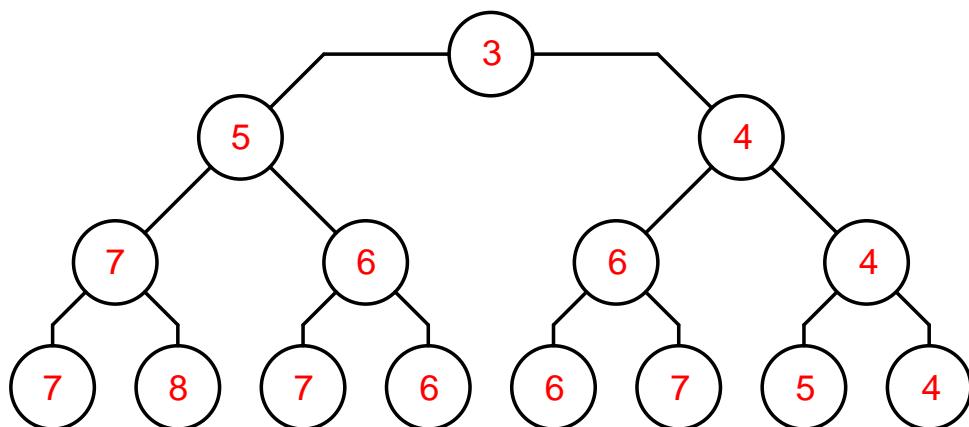
1.4) (4 pts) Dessinez l'état du monceau MIN suite à deux insertions : `insert(3)` suivi de `insert(4)`.



1.4.1) (2 pts) Monceau résultant du premier `insert(3)`:



1.4.2) (2 pts) Monceau résultant du second `insert(4)`:



- 1.5) (5 pts) Complétez la fonction `troisieme()` qui retourne la troisième plus petite valeur d'un monceau MIN. Le code Java complet de l'implémentation du monceau MIN est donné à l'Annexe 1 :

```
@SuppressWarnings("unchecked")
public AnyType troisieme() throws Exception {

    if( currentSize < 3)
        throw new Exception("Le monceau a moins de 3 éléments");

    int maxIndex = Math.min(7, this.currentSize); // COMPLÉTER ICI

    AnyType[] arr = (AnyType[]) new Comparable[3];

    arr[ 0 ] = array[ 1 ];

    arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
                array[ 2 ] : array[ 3 ];

    arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
                array[ 2 ] : array[ 3 ];

    for(int i = 4; i <= maxIndex; i++) {
        if( array[ i ].compareTo( arr[ 1 ] ) < 0 ) {
            % COMPLÉTER ICI

            arr[ 2 ] = arr[ 1 ];
            arr[ 1 ] = array[ i ];

        }
        else if( array[ i ].compareTo( arr[ 2 ] ) < 0 ) {
            % COMPLÉTER ICI

            arr[ 2 ] = array[ i ];

        }
    }

    return arr[2];
}
```

Question 2 : Recherche de patron**(18 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier via un calcul de hachage. On considère l'alphabet $\Sigma = \{A, C, G, T\}$ où $d = |\Sigma| = 4$. On admettra l'encodage suivant :

Symbol	A	C	G	T
Code	0	1	2	3

Le hachage est calculé en base $d = 4$ modulo $q = 17$. Par exemple, la séquence "GATTACA", encodée **2, 0, 3, 3, 0, 1, 0** produira une valeur de hash :

$$\begin{aligned} \text{mod}((((((2 \cdot d + 0) \cdot d + 3) \cdot d + 3) \cdot d + 0) \cdot d + 1) \cdot d + 0, q) &= \\ \text{mod}((((((2 \cdot 4 + 0) \cdot 4 + 3) \cdot 4 + 3) \cdot 4 + 0) \cdot 4 + 1) \cdot 4 + 0, 17) &= \\ &= 10. \end{aligned}$$

2.1) (3 points) Donnez le hash p produit par $P[1:4] = \text{"CATA"}$. Justifiez votre réponse par un calcul.

$$p = \text{mod}(((1 \cdot 4 + 0) \cdot 4 + 3) \cdot 4 + 0, 17) = \text{mod}(76, 17) = 8$$

Afin d'accélérer les calculs lors de la recherche du patron dans le texte, Rabin Karp exploite une formule récursive : $t_{s+1} = \text{mod}((t_s - hT[s+1])d + T[s+m+1], q)$, où $h = \text{mod}(d^{m-1}, q)$.

2.2) (3 points) Donnez la valeur de la variable h pour le patron $P[1 :4] = \text{"CATA"}$.

$$\text{Votre réponse : } h = \text{mod}(4^3, 17) = \text{mod}(64, 17) = 13$$

Supposons maintenant que nous recherchions un patron P de longueur $m= 17$ dans un texte T de longueur $n = 21$;

- En admettant que le hash associé à P est $p = 0$;
- En admettant que $h = \text{mod}(d^{m-1}, q) = \text{mod}(4^{16}, 17) = 1$;
- En admettant que "GA" est préfixe de P ($P = "GA..."$) ;
- En admettant que "AA" est suffixe de P ($P = "...AA"$) ;
- En admettant que T est la concaténation de "AA", de P et de "TT", c.-à-d. $T = "AA" P "TT"$.

2.3) (6 points) Complétez la table ci-après. On admettra $t_s \geq 0$ (ajoutez 17 si votre résultat est négatif).

s : décalage	t_s	Correspondance du hash
0	0	✓
1	0	✓
2	0	✓
3	12	
4	0	✓

2.4) (6 points) Combien de faux positifs seront détectés en recherchant P dans T ? Justifiez votre réponse.

Il y aurait au total 3 faux positifs, au décalage 0, 1 et 4.

À 0, on sait que la chaîne détectée est incorrecte car P débute par G alors que $T[1] = A$.

À 1, on sait que la chaîne détectée est incorrecte car P débute par G alors que $T[2] = A$.

À 4, on sait que la chaîne détectée est incorrecte car P termine par AA alors que $T[20 : 21] = TT$.

Question 3 : Plus Longue Sous-séquence Commune (PLSC) (18 points)

3.1) (8 pts) En vous aidant du tableau suivant, donnez la PLSC des deux mots "aabccabcba" et "abcbaccba".

		a	a	b	c	c	a	b	c	b	a
		0	0	0	0	0	0	0	0	0	0
a	0										
b											
c	0										
b	0										
a	0										
c	0										
c	0										
b	0										
a	0										
a	0										

PLSC: **abcbcba**

3.2) (4 pts) Proposez une autre PLSC au couple de mots "aabccabcba" et "abcbaccba".

abcacba
abcccba

3.3) (6 pts) En utilisant les résultats du tableau que vous avez obtenu en 3.1), donnez l'ensemble des PLSC des mots "aabccab" et "abcbac".

abca
abcb
abcc

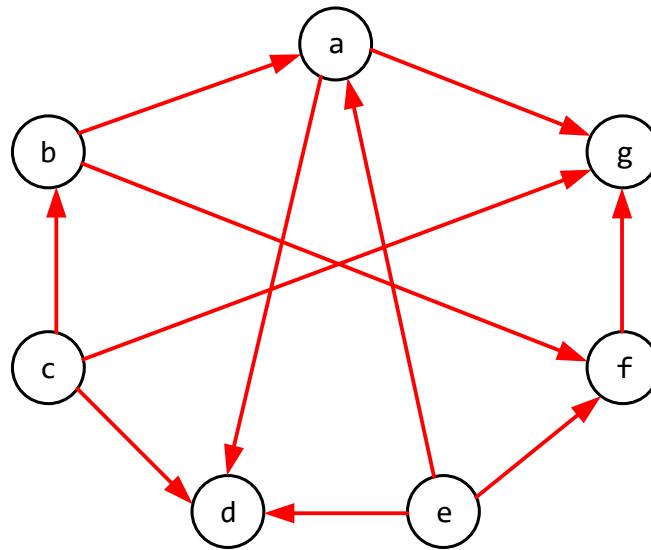
Question 4 : Ordre topologique**(17 points)**

On veut connaître l'ordre topologique du graphe dirigé acyclique suivant :

$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(a, d), (a, g), (b, a), (b, f), (c, b), (c, d), (c, g), (e, a), (e, d), (e, f), (f, g)\}$$

4.1) (2 pts) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (6 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant une liste de travail vu en classe. Utilisez une file (FIFO) comme liste de travail. Visitez les voisins par ordre alphabétique.

Nœud	1	2	3	4	5	6	7
a	2	2	1	0	-	-	-
b	1	0	-	-	-	-	-
c	0	-	-	-	-	-	-
d	3	2	1	1	0	-	-
e	0	-	-	-	-	-	-
f	2	2	1	0	-	-	-
g	3	2	2	2	1	0	-
Nouveaux nœuds	c, e	b	-	a, f	d	g	-
Nœud retiré	c	e	b	a	f	d	g

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	4	3	1	6	2	5	7

- 4.3) (2 pts)** Aurions-nous pu utiliser une pile (LIFO) à la place de la file pour exécuter l'algorithme précédent et obtenir un résultat correct ? Justifiez brièvement.

Oui, l'ordre de sortie de la liste de travail n'importe pas.

- 4.4) (5 pts)** Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant le parcours DFS post-ordre inverse. Partez du noeud a et visitez les noeuds alphabétiquement.

- 4.4.1) (3 pts)** Donnez l'affichage DFS post-ordre obtenu :

d g a f b c e

- 4.4.2) (1 pts)** Donnez l'affichage DFS post-ordre inverse obtenu :

e c b f a g d

- 4.4.3) (1 pts)** Donnez l'ordre topologique trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	5	3	2	7	1	4	6

- 4.5) (2 pts)** Proposez un ordre topologique alternatif qui diffère des deux trouvés précédemment.

Nœud	a	b	c	d	e	f	g
Ordre :	4	2	1	6	3	5	7

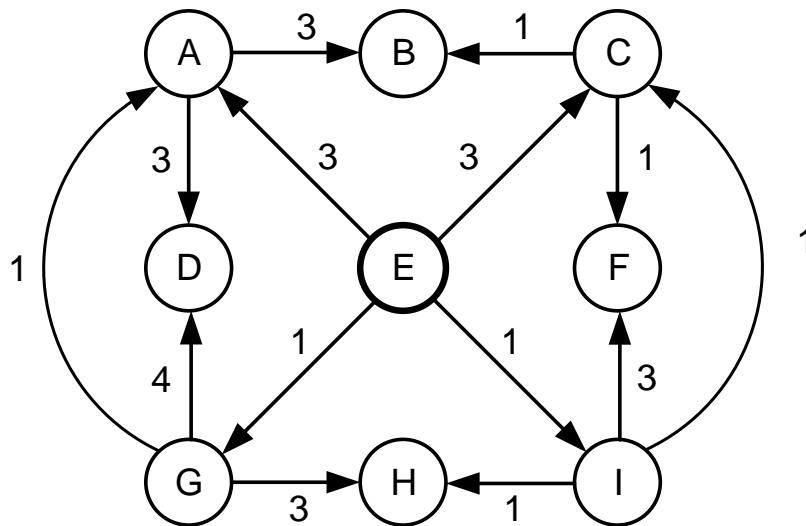
D'autres solutions existent, par exemple:

Nœud	a	b	c	d	e	f	g
Ordre :	5	3	1	6	2	4	7

Nœud	a	b	c	d	e	f	g
Ordre :	5	2	1	6	3	4	7

Question 5 : Algorithme de Dijkstra**(14 points)**

Considérez le graphe suivant.



- a) (8 pts) Exécutez l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de E. Visitez les voisins d'un nœud en ordre alphabétique. Le remplissage du tableau est noté.

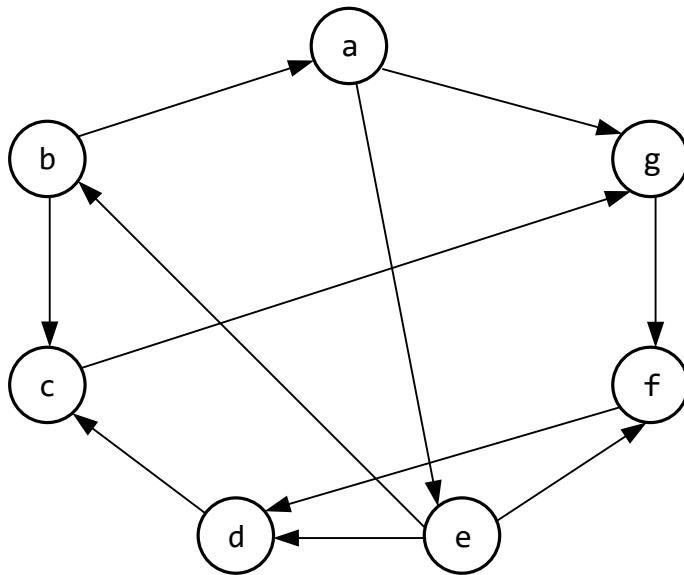
Nœud	Connu	Dist min.	Parent
A	✓	$\infty, 3, 2$	E, G
B	✓	$\infty, 5, 3$	A, C
C	✓	$\infty, 3, 2$	E, I
D	✓	$\infty, 5$	G
E	✓	0,	-
F	✓	$\infty, 4, 3$	I, C
G	✓	$\infty, 1$	E
H	✓	$\infty, 4, 2$	G, I
I	✓	$\infty, 1$	E

b) (6 pts) Détaillez chacun des chemins les plus courts trouvés parmi ceux demandés :

Destination	Le plus court chemin	Distance parcourue
B	E → I → C → B	3
D	E → G → D	5
F	E → I → C → F	3
H	E → I → H	2

Question 6 : Composantes fortement connexes**(10 points)**

On veut connaître les composantes fortement connexes du graphe dirigé suivant :

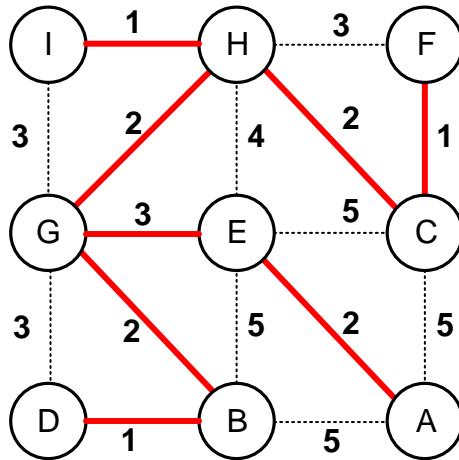


Complétez le tableau suivant en associant chacun des nœuds a à g à une composante fortement connexe que vous aurez identifié. Chaque colonne représente composante. Un nœud X est associé à une composante Y en noircissant la case correspondant à l'intersection de la ligne X et de la colonne Y. Par exemple, le nœud a sera associé à la composante 1. Laissez les colonnes inutilisées vides.

Nœud	Composante						
	1	2	3	4	5	6	7
a	●						
b	●						
c		●					
d		●					
e	●						
f		●					
g		●					

Question 7 : Arbre sous-tendant minimum**(5 points)**

Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal en noircissant les arêtes retenues dans le graphe ci-après. Donnez le coût de l'arbre ainsi obtenu.



Coût: _____

Kruskal :

Arête	Retenue?
(A, B)	
(A, C)	
(A, E)	✓
(B, D)	✓
(B, E)	
(B, G)	✓
(C, E)	
(C, F)	✓
(C, H)	✓
(D, G)	
(E, G)	✓
(E, H)	
(F, H)	
(G, H)	✓
(G, I)	
(H, I)	✓

Annexe 1

```

public class BinaryHeap<AnyType extends Comparable<? super AnyType>>
{
    public BinaryHeap( )
    {
        this( DEFAULT_CAPACITY );
    }

    @SuppressWarnings("unchecked")
    public BinaryHeap( int capacity )
    {
        currentSize = 0;
        array = (AnyType[]) new Comparable[ capacity + 1 ];
    }

    @SuppressWarnings("unchecked")
    public BinaryHeap( AnyType [ ] items )
    {
        currentSize = items.length;
        array = (AnyType[]) new Comparable[ ( currentSize + 2 ) * 11 / 10 ];

        int i = 1;
        for( AnyType item : items )
            array[ i++ ] = item;
        buildHeap();
    }

    public void insert( AnyType x )
    {
        if( currentSize == array.length - 1 )
            enlargeArray( array.length * 2 + 1 );

        // Percolate up
        int hole = ++currentSize;
        for( ; hole > 1 && x.compareTo( array[ hole / 2 ] ) < 0; hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
        array[ hole ] = x;
    }

    @SuppressWarnings("unchecked")
    private void enlargeArray( int newSize )
    {
        AnyType [ ] old = array;
        array = (AnyType []) new Comparable[ newSize ];
        for( int i = 0; i < old.length; i++ )
            array[ i ] = old[ i ];
    }

    public AnyType findMin( )
    {
        if( isEmpty( ) )
            return null;
        return array[ 1 ];
    }
}

```

```
public AnyType deleteMin( )
{
    if( isEmpty( ) )
        return null;

    AnyType minItem = findMin( );
    array[ 1 ] = array[ currentSize-- ];
    percolateDown( 1 );

    return minItem;
}

private void buildHeap( )
{
    for( int i = currentSize / 2; i > 0; i-- )
        percolateDown( i );
}

public boolean isEmpty( )
{
    return currentSize == 0;
}

public void makeEmpty( )
{
    currentSize = 0;
}

private static final int DEFAULT_CAPACITY = 10;

private int currentSize;      // Number of elements in heap
private AnyType [ ] array; // The heap array

private void percolateDown( int hole )
{
    int child;
    AnyType tmp = array[ hole ];

    for( ; hole * 2 <= currentSize; hole = child )
    {
        child = hole * 2;
        if( child != currentSize &&
            array[ child + 1 ].compareTo( array[ child ] ) < 0 )
            child++;
        if( array[ child ].compareTo( tmp ) < 0 )
            array[ hole ] = array[ child ];
        else
            break;
    }
    array[ hole ] = tmp;
}
```

```
@SuppressWarnings("unchecked")
public AnyType troisieme() throws Exception
{
    if( currentSize < 3)
        throw new Exception("Le monceau possède moins de 3 éléments");

    int maxIndex = // masqué pour la question;

    AnyType[] arr = (AnyType[]) new Comparable[3];
    arr[ 0 ] = array[ 1 ];
    arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
        array[ 2 ] : array[ 3 ];
    arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
        array[ 2 ] : array[ 3 ];

    for( int i=4; i <= maxIndex; i++)
    {
        if( array[ i ].compareTo( arr[ 1 ] ) < 0 )
        {
            // masqué pour la question
        }
        else if( array[ i ].compareTo( arr[ 2 ] ) < 0 )
        {
            // masqué pour la question
        }
    }

    return arr[2];
}
```



POLYTECHNIQUE
MONTRÉAL

Solutions de l'examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Total	

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20192
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Alejandro Quintero, responsable Tarek Ould Bachir, chargé de cours		B506	
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Mercredi	19 juin 2019	2 h 30	9 h 30 – 12 h 00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
<i>Directives particulières</i>			
<input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.			
Important	Ce corrigé contient 7 questions sur un total de 15 pages (incluant cette page) La pondération de cet examen est de 40 % Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(3 points/20)**

Pour cette question, référez-vous au code Java donné à l'Annexe 1. Il s'agit de l'implémentation d'un monceau min telle que vue en classe (Weiss) et augmentée de la méthode :

```
public String printArray() {
    StringBuilder sb = new StringBuilder();
    for (int i=1; i<=currentSize; i++ )
        sb.append(array[i].toString());
    return sb.toString();
}
```

Considérez la méthode `main` de l'Annexe 1 qui manipule un monceau `h` sur des objets de type `Character` ('A' < 'B' < 'C' < ...) et reproduisez les cinq (5) affichages qui y sont indiqués pour chacune des questions suivantes, tel que les exécute la méthode `main`.

Utilisez les cases du tableau pour écrire un caractère, comme illustré dans l'exemple suivant, résultant du 1^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 17 de l'Annexe 1, ligne 54 :

E	X	P													
---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--

Q1.1) **(0.5 point)** Donnez le résultat du 2^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 18 de l'Annexe 1, ligne 11.

A	E	D	T	R	T										
---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--

Q1.2) **(0.5 point)** Donnez le résultat du 3^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 18 de l'Annexe 1, ligne 19.

R	T	T													
---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--

Q1.3) **(0.5 point)** Donnez le résultat du 4^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 18 de l'Annexe 1, ligne 32.

E	T	O	T	U	R	S									
---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--

Q1.4) **(0.5 point)** Donnez le résultat du 5^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 18 de l'Annexe 1, ligne 40.

S	T	U	T												
---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--

Q1.5) **(1 point)** Donnez le résultat du 6^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray())` — en page 18 de l'Annexe 1, ligne 48.

E	E	E	L	L	E	M	R	N	T						
---	---	---	---	---	---	---	---	---	---	--	--	--	--	--	--

Question 2 : Recherche de patron**(3 points/20)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier via un calcul de hachage. On considère l'alphabet $\Sigma = \{A, C, G, T\}$ où $d = |\Sigma| = 4$. On admettra l'encodage suivant :

Symbol	A	C	G	T
Code	0	1	2	3

Le hachage est calculé en base $d = 4$ modulo $q = 17$. Par exemple, la séquence "GATTACA", encodée **2, 0, 3, 3, 0, 1, 0** produira une valeur de hash :

$$\begin{aligned} \text{mod}((((((2 \cdot d + 0) \cdot d + 3) \cdot d + 3) \cdot d + 0) \cdot d + 1) \cdot d + 0, q) &= \\ \text{mod}((((((2 \cdot 4 + 0) \cdot 4 + 3) \cdot 4 + 3) \cdot 4 + 0) \cdot 4 + 1) \cdot 4 + 0, 17) &= \\ &= 10. \end{aligned}$$

2.1) (0.5 point) Donnez le hash p produit par $P[1:12] = \text{"TTAAGGAACCAA"}$. Justifiez votre réponse par un calcul.

$$\text{mod}(((15 \cdot 16 + 0) \cdot 16 + 10) \cdot 16 + 0) \cdot 16 + 5) \cdot 16 + 0, 17) = 4$$

Afin d'accélérer les calculs lors de la recherche du patron dans le texte, Rabin Karp exploite une formule récursive : $t_{s+1} = \text{mod}((t_s - hT[s+1])d + T[s+m+1], q)$, où $h = \text{mod}(d^{m-1}, q)$.

2.2) (0.5 point) Donnez la valeur de la variable h pour le patron $P[1 : 12] = \text{"TTAAGGAACCAA"}$.

Votre réponse : $h = \text{mod}(4^{12-1}, 17) = \text{mod}(4^{11}, 17) = 13$

Supposons maintenant que nous recherchions un patron P de longueur $m = 16$ dans un texte T de longueur $n = 20$;

- En admettant que le hash associé à P est $p = 0$;
- En admettant que $h = \text{mod}(d^{m-1}, q) = \text{mod}(4^{15}, 17) = 13$;
- En admettant que "AAA" est préfixe de P ($P = \text{"AAA..."}^*$) ;
- En admettant que "AAA" est suffixe de P ($P = \dots\text{AAA}$) ;
- En admettant que T est la concaténation de "GAG", de P et de "CAC", c.-à-d. $T = \text{"GAG"} P \text{"CAC"}$.

2.3) (1.5 point) Complétez la table ci-après. On admettra $t_s \geq 0$ (ajoutez 17 au hash si votre résultat est négatif).

S : décalage	t_s	Correspondance du hash
0	0	✓
1	15	
2	9	
3	0	✓
4	1	
5	4	
6	0	✓

2.4) (0.5 point) Combien de faux positifs seront détectés en recherchant P dans T ? Justifiez votre réponse.

Deux faux positifs, à $s = 0$ et $s = 6$. Dans le premier cas, la chaîne débute par GAG alors que le patron recherché débute par AAA. Dans le second cas, la chaîne se termine par CAC alors que le patron termine par AAA.

Question 3 : Programmation dynamique**(4 points/20)**

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$A_1 : 2 \times 4 ; A_2 : 4 \times 2 ; A_3 : 2 \times 3 ; A_4 : 3 \times 3 ; A_5 : 3 \times 2$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	16	28	46	54
2		0	24	42	46
3			0	18	30
4				0	18
5					0

s	1	2	3	4	5
1		1	2	2/3	2
2			2	2	2
3				3	3/4
4					4
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

3.1) (0.5 point) Donnez le parenthésage optimal pour multiplier A_1 à A_3 . Donnez son coût.

Parenthésage optimal : $(A_1 \quad A_2) \quad A_3$

Coût: **28**

3.2) (1 point) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

Parenthésage optimal : $(A_1 \quad A_2)(A_3 \quad A_4)$

Coût: **46**

ou

Parenthésage optimal : $((A_1 \quad A_2) \quad A_3)(A_4)$

Coût: **46**

3.3) (1 point) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal : $A_2 \quad (\quad A_3 \quad (\quad A_4 \quad A_5 \quad))$

Coût: **46**

ou

Parenthésage optimal : $A_2 \quad ((\quad A_3 \quad A_4 \quad) \quad A_5 \quad)$

Coût: **46**

3.4) (1.5 point) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal : $(\quad A_1 \quad A_2 \quad) \quad (\quad A_3 \quad (\quad A_4 \quad A_5 \quad))$

Coût: **54**

ou

Parenthésage optimal : $(\quad A_1 \quad A_2 \quad) \quad (\quad A_3 \quad A_4 \quad) \quad A_5 \quad)$

Coût: **54**

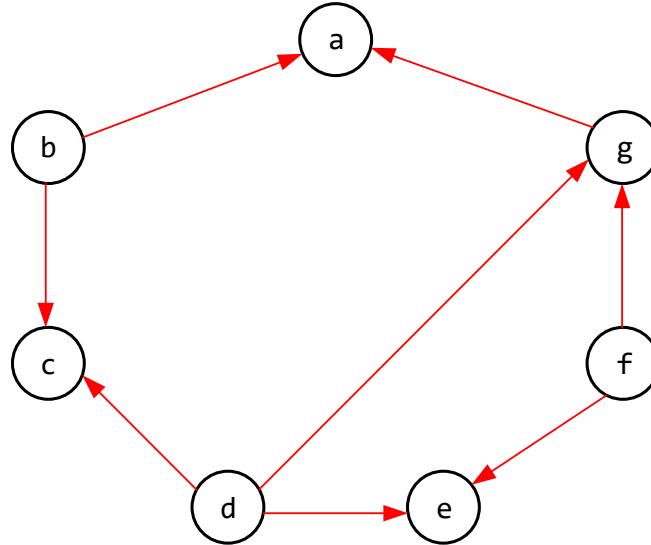
Question 4 : Ordre topologique**(2.5 points)**

On veut connaître l'ordre topologique du graphe dirigé acyclique suivant :

$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(b, a), (b, c), (d, c), (d, e), (d, g), (f, e), (f, g), (g, a)\}$$

4.1) (0.5 point) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (1 point) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant une liste de travail vu en classe. Utilisez une file (FIFO) comme liste de travail. Visitez les voisins par ordre alphabétique.

Nœud	1	2	3	4	5	6	7
a	2	1	1	1	1	1	0
b	0	-	-	-	-	-	-
c	2	1	0	-	-	-	-
d	0	-	-	-	-	-	-
e	2	2	1	0	-	-	-
f	0	-	-	-	-	-	-
g	2	2	1	0	-	-	-
Nouveaux nœuds	b, d, f	-	c	e, g	-	-	a
Nœud retiré	b	d	f	c	e	g	a

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	7	1	4	2	5	3	6

4.3) (1 point) Proposez six (6) ordres topologiques alternatifs à G :

Nœud	a	b	c	d	e	f	g
Ordre :	7	2	4	1	5	3	6

Nœud	a	b	c	d	e	f	g
Ordre :	7	2	3	1	5	4	6

Nœud	a	b	c	d	e	f	g
Ordre :	7	1	3	2	5	4	6

Nœud	a	b	c	d	e	f	g
Ordre :	6	2	4	1	7	3	5

Nœud	a	b	c	d	e	f	g
Ordre :	6	2	3	1	7	4	5

Nœud	A	b	c	d	e	f	G
Ordre :	6	1	3	2	7	4	5

Question 5 : Composantes fortement connexes **(2.5 points/20)**

On veut connaître les composantes fortement connexes des graphe dirigé $G_1 = (V_1, E_1)$ et $G_{21} = (V_2, E_2)$:

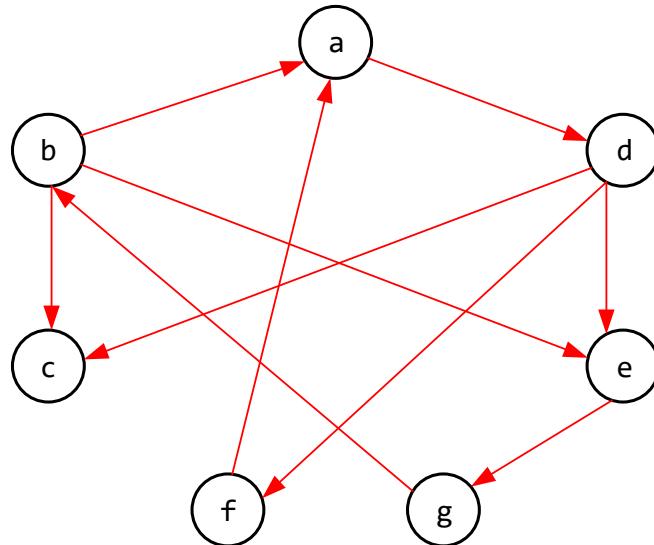
$$V_1 = \{a, b, c, d, e, f, g\}$$

$$E_1 = \{(a, d), (b, a), (b, c), (b, e), (d, c), (d, e), (d, f), (e, g), (f, a), (g, b)\}$$

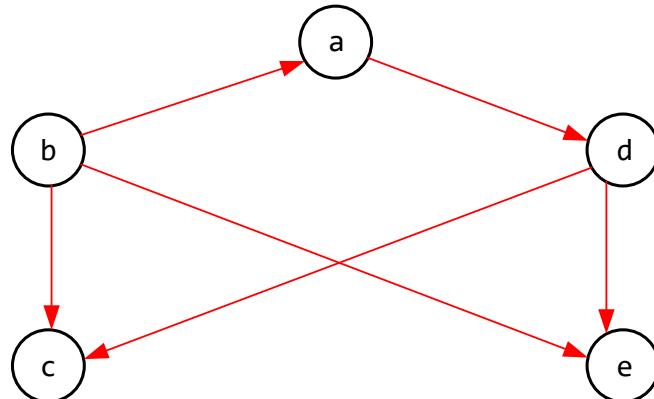
$$V_2 = \{a, b, c, d, e\}$$

$$E_2 = \{(a, d), (b, a), (b, c), (b, e), (d, c), (d, e)\}$$

5.1) (0.25 point) Reproduisez graphiquement le graphe $G_1 = (V_1, E_1)$:



5.2) (0.25 point) Reproduisez graphiquement le graphe $G_2 = (V_2, E_2)$:



5.3) (1 point) Donnez les composantes fortement connexes de G_1 en associant chacun des nœuds a à g à une composante. Les composantes sont numérotées incrémentalement et la numérotation débute à 1. Laissez les colonnes inutilisées vides.

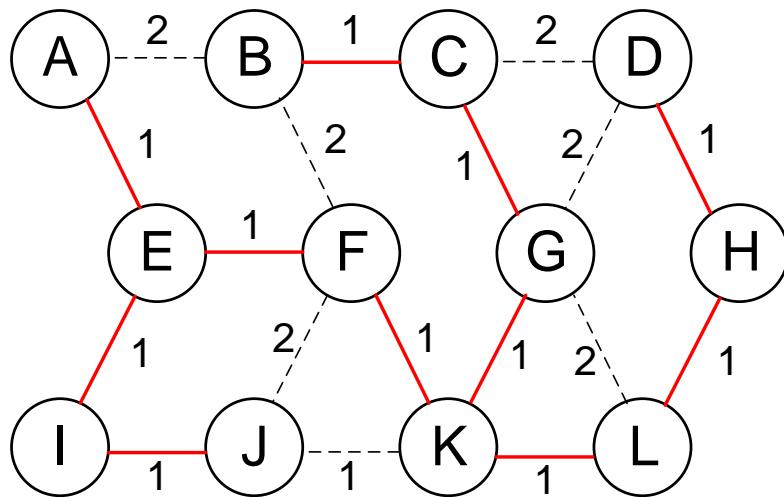
Nœud	Composantes						
	1	2	3	4	5	6	7
a	•						
b	•						
c		•	x				
d	•						
e	•						
f	•						
g	•						

5.4) (1 point) Donnez les composantes fortement connexes de G_2 en associant chacun des nœuds a à e à une composante. Les composantes sont numérotées incrémentalement et la numérotation débute à 1. Laissez les colonnes inutilisées vides.

Nœud	Composantes				
	1	2	3	4	5
a	•				
b		•			
c			•		
d				•	
e					•

Question 6 : Arbre sous-tendant minimum**(2 points/20)**

6.1) (1 point) Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal en noircissant les arêtes retenues dans le graphe ci-après. Donnez le coût de l'arbre ainsi obtenu. Aidez-vous du tableau donné en page suivante

**Coût: 11**

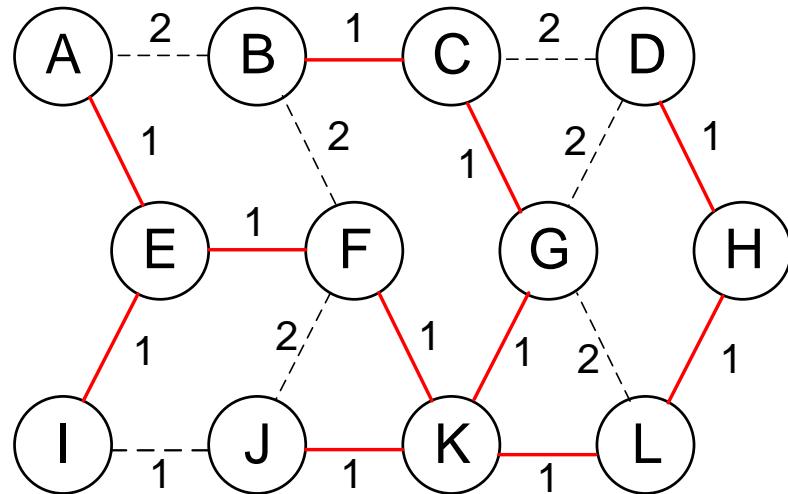
Kruskal :

Arête	Retenue?
(A, B)	
(A, E)	
(B, C)	
(B, F)	
(C, D)	
(C, G)	
(D, G)	
(D, H)	
(E, F)	
(E, I)	
(F, J)	
(F, K)	
(G, K)	
(G, L)	
(H, L)	
(I, J)	
(J, K)	
(K, L)	

6.2) (1 point) Le graphe précédent admet-il un autre arbre sous-tendant minimum? Si oui, dites lequel. Si non, expliquer pourquoi

OUI :

Arbre alternatif :



NON :

Justification :

Question 7 : Questions théoriques**(3 points/20)**

Répondez aux questions suivantes en justifiant brièvement vos réponses à chaque fois.

7.1) (0.5 point) Si l'algorithme BFS vu en classe utilise une Pile au lieu d'une File, l'algorithme ainsi obtenu réalise-t-il un parcours DFS ?

C'est exact. Le résultat sera cependant différent de celui du DFS vu en classe.

7.2) (0.5 point) Est-il vrai que le chemin le plus court d'un graph non valué peut être obtenu au moyen d'un parcours DFS ?

Faux. Un contre- exemple est un graphe non dirigé formé en anneau (ring).

7.3) (0.5 point) Est-il vrai que la solution vue en classe au problème de parenthésage des matrices est un algorithme qui s'exécute en temps polynomial ?

C'est correct, il s'exécute en $O(n^2)$, n étant le nombre de matrices.

7.4) (0.5 point) Si un graphe non dirigé G est connexe et acyclique, peut-il ne pas être un arbre ?

Il est nécessairement un arbre.

7.5) (0.5 point) Si un graphe dirigé G est connexe, G^T peut-il ne pas l'être aussi ?

Il l'est forcément, la transposition maintient les composantes fortement connexes.

7.6) (0.5 point) L'arbre sous-tendant minimum d'un graphe valué, non dirigé G est-il unique ?

Pas nécessairement. Il arrive qu'un arbre admette plus d'un arbre sous-tendant minimum.



POLYTECHNIQUE
MONTRÉAL

Questionnaire examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Total	

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20193
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo, responsable Tarek Ould Bachir, chargé de cours			5193
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Mercredi	18 décembre 2019	2 h 30	9 h 30 – 12 h 00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.
<i>Directives particulières</i>			
<p> Ne posez pas de question durant l'examen. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites. Un cahier d'examen vous est fourni à titre de brouillon. Ne remettez pas le cahier d'examen. Remettez ce questionnaire avec vos réponses dans les espaces réservés à cet effet.</p>			
Important	Cet examen contient 5 questions sur un total de 14 pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux		
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(4/20 points)**

Pour cette question, référez-vous au code Java donné à l'Annexe 1. Il s'agit de l'implémentation d'un monceau min telle que vue en classe (Weiss) et augmentée de la méthode :

```
public String printArray() {  
    StringBuilder sb = new StringBuilder();  
    for (int i=1; i<=currentSize; i++ )  
        sb.append(array[i].toString() + " ");  
    return sb.toString();  
}
```

Considérez la méthode `main` de l'Annexe 1 qui manipule un monceau `h` sur des objets de type `Integer` et reproduisez les cinq (5) affichages qui y sont indiqués pour chacune des questions suivantes, tel que les exécute la méthode `main`. À titre d'exemple, les lignes 107 et 108 de l'Annexe 1 affichera :

Exemple

1 3 2

Q1.1) **(0.75 point)** Donnez le résultat de l'affichage obtenu en exécutant les lignes 119 et 120 de l'Annexe 1.

Q 1.1

1 2 3 5 5 4

Q1.2) **(0.75 point)** Donnez le résultat de l'affichage obtenu en exécutant les lignes 130 et 131 de l'Annexe 1.

Q 1.2

4 5 5

Q1.3) (**0.75 point**) Donnez le résultat de l'affichage obtenu en exécutant les lignes 144 et 145 de l'Annexe 1.

Q 1.3

1 3 1 6 4 5 2

Q1.4) (**0.75 point**) Donnez le résultat de l'affichage obtenu en exécutant les lignes 155 et 156 de l'Annexe 1.

Q 1.4

3 4 5 6

Q1.5) (**1 point**) Donnez le résultat de l'affichage obtenu en exécutant les lignes 164 et 165 de l'Annexe 1.

Q 1.5

1 2 3 6 5 4 7 8 9

Question 2 : Automate de reconnaissance de motifs**(4/20 points)**

- 2.1) **(1.5 point)** En utilisant l'algorithme de construction des automates de reconnaissance de motifs, construisez l'automate capable de reconnaître la séquence suivante:

« ababc »

en REMPLISSANT le **Tableau 1** avec la fonction de transition d'état correspondante.

Tableau 1

Symboles États	a	b	c
0	1	0	0
1	1	2	0
2	3	0	0
3	1	4	0
4	3	0	5
5	1	0	0
6			
7			

NOTE : Ajoutez ou ignorez des lignes ou des colonnes au besoin.

État initial : 0

État final : 5

- 2.2) **(1 point)** Exécutez l'automate en reconnaissance sur le texte suivant:

« bcababcabc »

en remplissant le **Tableau 2** avec l'état de l'automate APRÈS l'analyse des sous-chaînes (préfixes) indiquées.

Tableau 2

Sous-chaîne	État
b	0
bc	0
bca	1
bcab	2
bcaba	3
bcabab	4
bcababc	5
bcababca	1
bcababcab	2
bcababcbc	0
bcababcabca	1

2.3) (0.5 point) Combien de fois le motif a été reconnu dans l'exécution de l'automate ?

Motif reconnu 1 fois au décalage: 2.

2.4) (0.5 point) Quelle est la complexité asymptotique de l'algorithme de CONSTRUCTION des automates de reconnaissance de motifs ?

Complexité de construction: $O(m^3 d)$.

2.5) (0.5 point) Quelle est la complexité asymptotique de l'algorithme de RECONNAISSANCE d'un motif en utilisant un automate Q ?

Complexité de reconnaissance: $O(n)$.

Question 3 : Programmation dynamique**(4/20 points)**

3.1) (1 point) REMPLISSEZ le Tableau 3 suivant avec les informations de longueur et de provenance pour retrouver la plus longue sous-séquence commune aux chaînes en entrée :

$$X = a \ b \ a \ a \ b \ b \ a \ b \ a \ b \ b \ b \ b \ a \text{ et } Y = b \ a \ a \ b \ a \ b \ a \ b \ a$$

Tableau 3

		b	a	a	b	a	b	a	b	a
		0	0	0	0	0	0	0	0	0
a	0	=\	\=	\=	--	\=	--	\=	--	\=
	0	1	1	1	1	1	1	1	1	1
b	0	\=	=\	=\	--	\=	--	\=	--	--
	0	1	1	1	2	2	2	2	2	2
a	0		\	\	=\	\	--	\	--	\
	0	1	2	2	2	3	3	3	3	3
a	0		\	\	--	\	=\	\	--	\
	0	1	2	3	3	3	3	4	4	4
b	0	\			\	--	\	=\	\	--
	0	1	2	3	4	4	4	4	5	5
b	0	\			\	=\	\	--	\	=\
	0	1	2	3	4	4	5	5	5	5
a	0		\	\		\	=\	\	--	\
	0	1	2	3	4	5	5	6	6	6
b	0	\			\		\	=\	\	--
	0	1	2	3	4	5	6	6	7	7
a	0		\	\		\		\	=\	\
	0	1	2	3	4	5	6	7	7	8
b	0	\			\		\		\	=\
	0	1	2	3	4	5	6	7	8	8
b	0	\			\		\		\	=\
	0	1	2	3	4	5	6	7	8	8
b	0	\			\		\		\	=\
	0	1	2	3	4	5	6	7	8	8
b	0	\			\		\		\	=\
	0	1	2	3	4	5	6	7	8	9

3.2) (**0.5 point**) ÉCRIVEZ la longueur de la plus longue sous-séquence commune :

9

3.3) (**0.5 point**) ÉCRIVEZ la plus longue sous-séquence commune :

abababaab

3.4) CONSIDÉREZ les différentes sous séquences en commun de la même longueur maximale.

3.4.1) (**1 point**) Est-ce possible de savoir de façon booléenne simplement s'il y avait plus qu'une sous séquence en commun de longueur maximale sans en savoir le nombre, à partir seulement des résultats du **Tableau 3** ? Comment feriez-vous, si possible ? JUSTIFIEZ brièvement votre réponse.

Examiner toutes les cases (i, j) du la meilleure LCS et si $((p1[i] != p2[j]) \&\& (x[i-1] == x[j-1]))$ alors des chaînes optimales équivalentes existent.

3.4.2) (**1 point**) Est-ce possible de calculer le nombre de sous séquences en commun de longueur maximale à partir seulement des résultats du **Tableau 3** ? COMMENT feriez-vous, si possible ? JUSTIFIEZ brièvement votre réponse.

Examiner toutes les cases (i, j) du la meilleure LCS et si $((p1[i] != p2[j]) \&\& (x[i-1] == x[j-1]))$ alors récursivement explorer les LCS qui démarrent de $(i-1, j)$ et de $(i, j-1)$.

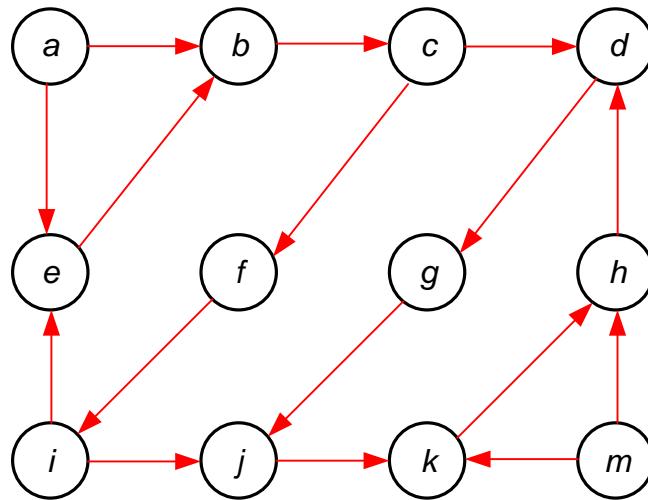
Question 4 : Composantes fortement connexes**(4/20 points)**

On veut connaître les composantes fortement connexes du graphe dirigé suivant :

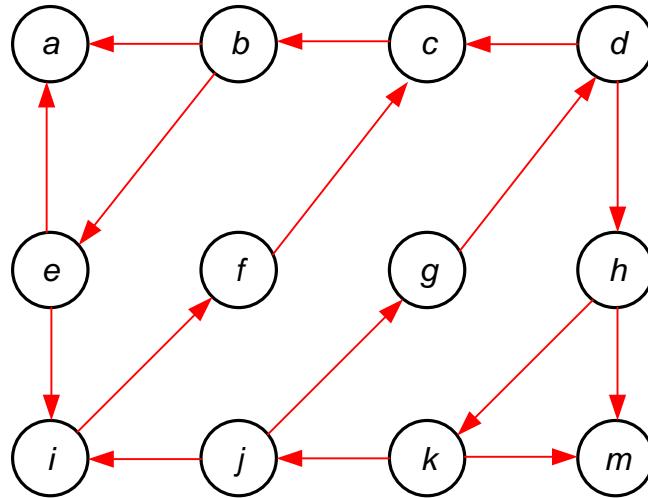
$$V = \{a, b, c, d, e, f, g, h, i, j, k, m\}$$

$$E = \{(a, b), (a, e), (b, c), (c, d), (c, f), (d, g), (e, b), (f, i), (g, j), (h, d), (i, e), (i, j), (j, k), (k, h), (m, h), (m, k)\}$$

4.1) (0.5 point) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (0.5 point) Donnez G^T , le graphe transposé de G :



4.3) (2 points) Donnez les composantes fortement connexes (CFC) de G en associant chacun des nœuds a à m ci-après à une composante. Les composantes sont numérotées de façon incrémentale et la numérotation débute à 1. Laissez les colonnes inutilisées vides.

Nœud	Composante						
	1	2	3	4	5	6	7
a	•						
b		•					
c		•					
d			•				
e		•					
f		•					
g			•				
h			•				
i		•					
j			•				
k			•				
m				•			

4.4) (0.5 point) Quel est le plus grand nombre d'arcs qu'il est possible de retirer du graphe G sans affecter ses CFC ? Justifiez brièvement votre réponse.

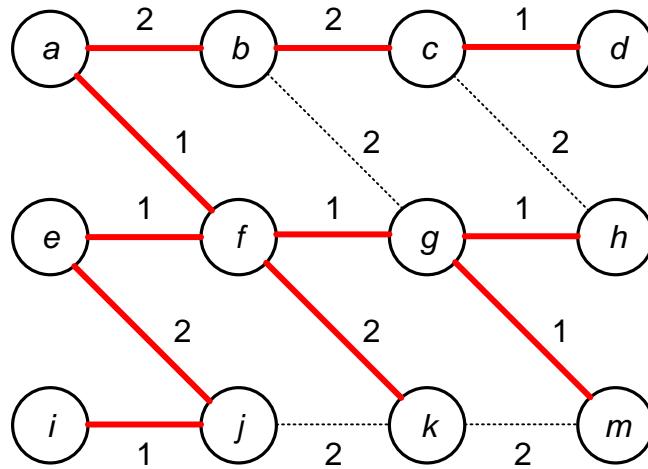
Il est possible de retirer les arcs (a, b), (a, e), (c, d), (i, j), (m, h), (m, k) sans affecter les CFC de G. La réponse est donc 6.

4.5) (0.5 point) Si on note G' le graphe obtenu en retirant de G tous les arcs n'affectant pas ses CFC tel que décrit en 4.4, combien $(G')^T$ admet-il de CFC ? Justifiez brièvement votre réponse.

Si tous les arcs identifiés en 4.4 sont retirés de G pour former G' , alors G' possède 4 CFC, et il en est de même pour $(G')^T$ puisque la transposition préserve les CFC.

Question 5 : Arbre sous-tendant minimum **(4/20 points + 1 point bonus)**

5.1) **(1.5 point)** Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal en noircissant les arêtes retenues dans le graphe ci-après. DONNEZ le coût de l'arbre ainsi obtenu.



Coût: 15

Kruskal :

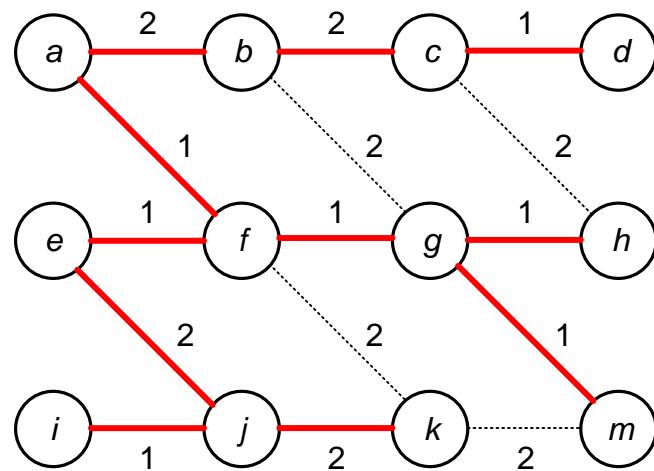
Arête	Coût	Retenue?
(a, b)	2	✓
(a, f)	1	✓
(b, c)	2	✓
(b, g)	2	
(c, d)	1	✓
(c, h)	2	
(e, f)	1	✓
(e, j)	2	✓
(f, g)	1	✓
(f, k)	2	✓
(g, h)	1	✓
(g, m)	1	✓
(i, j)	1	✓
(j, k)	2	
(k, m)	2	

5.2) (1 point) Quelle structure de données serait appropriée pour choisir l'arête de plus faible poids dans l'exécution de l'algorithme de Kruskal par ensembles disjoints ?

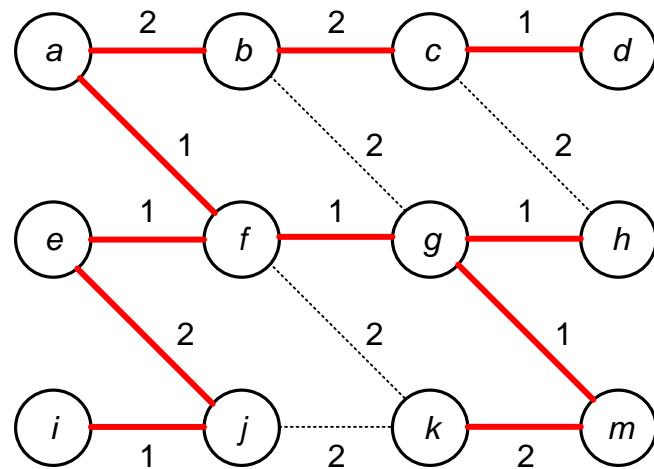
Un monceau.

5.3) (1.5 point) Le graphe précédent admet d'autres arbres sous-tendant minimum. Proposez-en trois.

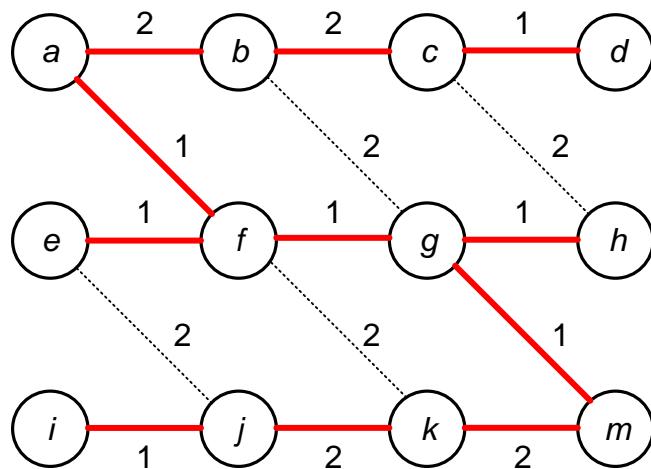
5.3.1)



5.3.2)



5.3.3)



5.4) (1 point bonus) Combien d'arbres sous-tendant minimum différents le graphe précédent admet-il au total ? Justifiez votre réponse par un calcul.

Tout arbre sous-tendant minimum du graphe contient nécessairement les arêtes ayant un coût de 1. Il reste alors à choisir 4 arêtes ayant un coût de 2 et permettant de former un arbre sous-tendant minimum. Pour ce faire, on choisira deux arêtes parmi (a, b), (b, c), (b, g), (c, h) (il existe 6 combinaisons possibles), et deux parmi (e, j), (f, k), (j, k), (k, m) (il existe 6 combinaisons aussi). Au total, il existe donc $6 \times 6 = 36$ arbres sous-tendant minimums pour ce graphe.

Annexe 1

```

001  public class BinaryHeap<AnyType extends Comparable<? super AnyType>> {
002
003      private static final int DEFAULT_CAPACITY = 11;
004
005      private int currentSize;           // Number of elements in heap
006      private AnyType [] array;         // The heap array
007
008      public BinaryHeap( ) { this( DEFAULT_CAPACITY ); }
009
010      @SuppressWarnings("unchecked")
011      public BinaryHeap( int capacity ) {
012          currentSize = 0;
013          array = (AnyType[]) new Comparable[ capacity + 1 ];
014      }
015
016      @SuppressWarnings("unchecked")
017      public BinaryHeap( AnyType [ ] items ) {
018          currentSize = items.length;
019          array = (AnyType[]) new Comparable[ ( currentSize + 2 ) * 11 / 10 ];
020
021          int i = 1;
022          for( AnyType item : items )
023              array[ i++ ] = item;
024          buildHeap( );
025      }
026
027      public void insert( AnyType x ) {
028          if( currentSize == array.length - 1 )
029              enlargeArray( array.length * 2 + 1 );
030
031          int hole = ++currentSize;
032          for( ; hole > 1 && x.compareTo( array[ hole / 2 ] ) < 0; hole /= 2 )
033              array[ hole ] = array[ hole / 2 ];
034          array[ hole ] = x;
035      }
036
037      @SuppressWarnings("unchecked")
038      private void enlargeArray( int newSize ) {
039          AnyType [] old = array;
040          array = (AnyType []) new Comparable[ newSize ];
041          for( int i = 0; i < old.length; i++ )
042              array[ i ] = old[ i ];
043      }
044
045      public AnyType findMin( ) throws Exception {
046          if( isEmpty( ) ) throw new Exception( );
047          return array[ 1 ];
048      }
049
050      public AnyType deleteMin( ) throws Exception {
051          if( isEmpty( ) ) throw new Exception( );
052
053          AnyType minItem = findMin( );
054          array[ 1 ] = array[ currentSize-- ];
055          percolateDown( 1 );
056
057          return minItem;
058      }

```

```
059  private void buildHeap( ) {
060      for( int i = currentSize / 2; i > 0; i-- )
061          percolateDown( i );
062  }
063
064  public boolean isEmpty( ) {
065      return currentSize == 0;
066  }
067
068  public void makeEmpty( ) {
069      currentSize = 0;
070  }
071
072  private void percolateDown( int hole ) {
073      int child;
074      AnyType tmp = array[ hole ];
075
076      for( ; hole * 2 <= currentSize; hole = child ) {
077          child = hole * 2;
078          if( child != currentSize &&
079              array[ child + 1 ].compareTo( array[ child ] ) < 0 )
080              child++;
081          if( array[ child ].compareTo( tmp ) < 0 )
082              array[ hole ] = array[ child ];
083          else
084              break;
085      }
086      array[ hole ] = tmp;
087  }
088
089  public String printArray() {
090      StringBuilder sb = new StringBuilder();
091      for (int i=1; i<=currentSize; i++ )
092          sb.append(array[i].toString() + " ");
093      return sb.toString();
094  }
095
096  public static void main( String [ ] args ) {
097
098      BinaryHeap<Integer> h;
099
100      // EXEMPLE
101      h = new BinaryHeap<Integer>();
102      h.insert( 3 );
103      h.insert( 2 );
104      h.insert( 1 );
105
106      // Affichage donné pour exemple
107      System.out.println( "Exemple" );
108      System.out.println( h.printArray() );
```

```
109     // QUESTION 1.1
110     h = new BinaryHeap<Integer>( );
111     h.insert( 5 );
112     h.insert( 3 );
113     h.insert( 2 );
114     h.insert( 1 );
115     h.insert( 5 );
116     h.insert( 4 );
117
118     // Affichage demandé pour Q 1.1
119     System.out.println( "Q 1.1" );
120     System.out.println( h.printArray() );
121
122     // QUESTION 1.2
123     try {
124         h.deleteMin();
125         h.deleteMin();
126         h.deleteMin();
127     } catch (Exception e) {e.printStackTrace();}
128
129     // Affichage demandé pour Q 1.2
130     System.out.println( "Q 1.2" );
131     System.out.println( h.printArray() );
132
133     // QUESTION 1.3
134     h = new BinaryHeap<Integer>( );
135     h.insert( 5 );
136     h.insert( 6 );
137     h.insert( 4 );
138     h.insert( 3 );
139     h.insert( 1 );
140     h.insert( 1 );
141     h.insert( 2 );
142
143     // Affichage demandé pour Q 1.3
144     System.out.println( "Q 1.3" );
145     System.out.println( h.printArray() );
146
147     // QUESTION 1.4
148     try {
149         h.deleteMin();
150         h.deleteMin();
151         h.deleteMin();
152     } catch (Exception e) {e.printStackTrace();}
153
154     // Affichage demandé pour Q 1.4
155     System.out.println( "Q 1.4" );
156     System.out.println( h.printArray() );
157
158     // QUESTION 1.5
159     Integer[] cs = {9, 8, 7, 6, 5, 4, 3, 2, 1};
160
161     h = new BinaryHeap<Integer>( cs );
162
163     // Affichage demandé pour Q 1.5
164     System.out.println( "Q 1.5" );
165     System.out.println( h.printArray() );
166 }
167 }
```



POLYTECHNIQUE
MONTRÉAL

Corrigé examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :		Prénom :
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre		
INF2010 – Structures de données et algorithmes		Tous	20181		
Professeur		Local	Téléphone		
Ettore Merlo, responsable / Tarek Ould Bachir, chargé					
Jour	Date	Durée	Heures		
lundi	23 avril 2014	2h30	9h30-12h00		
Documentation	Calculatrice				
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.			
Directives particulières					
<input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.					
Important	Cet corrigé contient 6 questions sur un total de 14 pages (excluant cette page) La pondération de cet examen est de 40 % Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non				

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

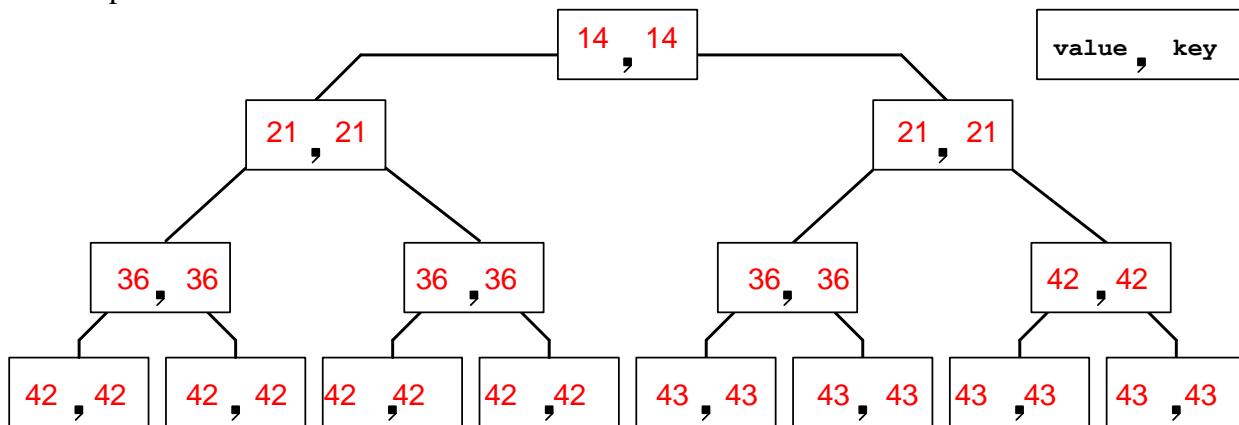
Question 1 : Monceaux **(20 points)**

Pour cette question, vous pouvez vous référer au code Java de l'Annexe 1.

- 1.1) (2 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	14	21	21	36	36	36	42	42	42	42	42	43	43	43	43

Votre réponse :



- 1.2) Considérez la fonction `changeKey(int location, int newKey)` donnée à l'Annexe 1 permettant de modifier la clé d'une entrée du monceau.

- 1.2.1) (1 pts) Quelle est la complexité asymptotique de cette fonction en cas moyen ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

$O(\lg(n))$ car elle consiste à :

1. Faire une copie de l'entrée à modifier et modifier sa clé : O(1)
 2. Écraser ladite entrée avec la dernière du monceau et réduire sa taille de 1 : O(1)
 3. Percoler vers le bas cette position : O($\lg(n)$) en moyenne
 4. Insérer l'entrée modifiée : O(1) en moyenne

1.2.2) (1 pts) Quelle est la complexité asymptotique de cette fonction en pire cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

O(lg(n)) car elle consiste à :

1. Faire une copie de l'entrée à modifier et modifier sa clé : $O(1)$
 2. Écarter ladite entrée avec la dernière du monceau et réduire sa taille de 1 : $O(1)$
 3. Percoler vers le bas cette position : $O(\lg(n))$ en pire cas
 4. Insérer l'entrée modifiée : $O(\lg(n))$ en pire cas

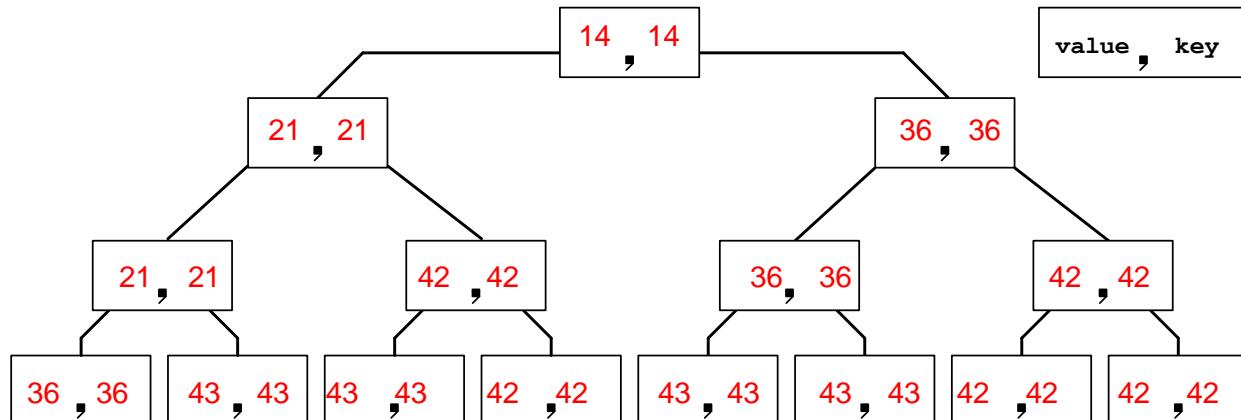
- 1.3) (2 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel :
BinaryHeap(values, true)

Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

Le tableau **values** est le suivant:

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	43	42	36	21	14	43	42	36	21	43	42	36	43	42	42

Monceau résultant :



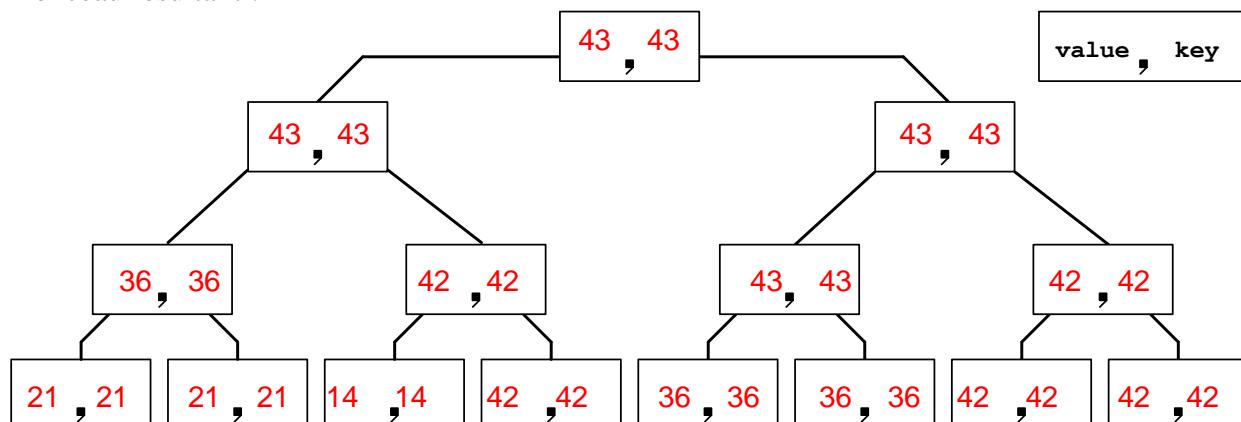
- 1.4) (2 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel :
BinaryHeap(values, false)

Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

Le tableau **values** est le suivant:

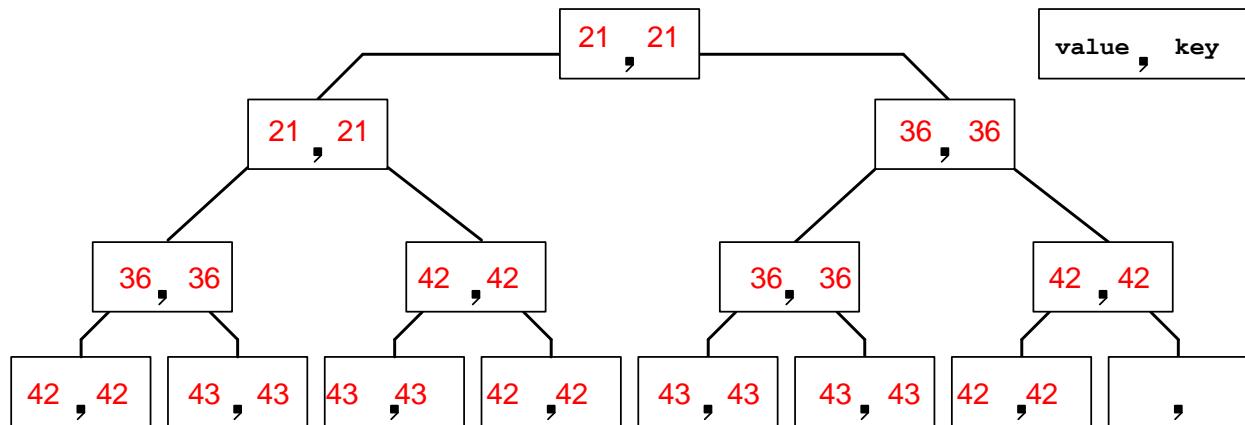
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	43	42	36	21	14	43	42	36	21	43	42	36	43	42	42

Monceau résultant :

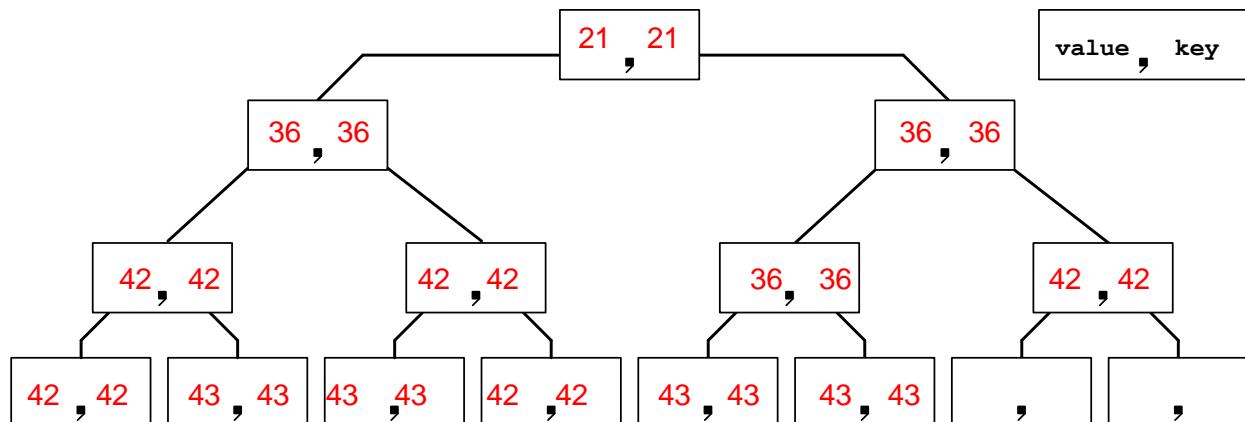


1.5) Dessinez l'état du monceau de la question 1.3) suite à deux appels consécutifs à `deleteRoot()` :

1.5.1) (1 pt) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de `value` et `key` de l'objet `Entry` contenues dans le monceau.

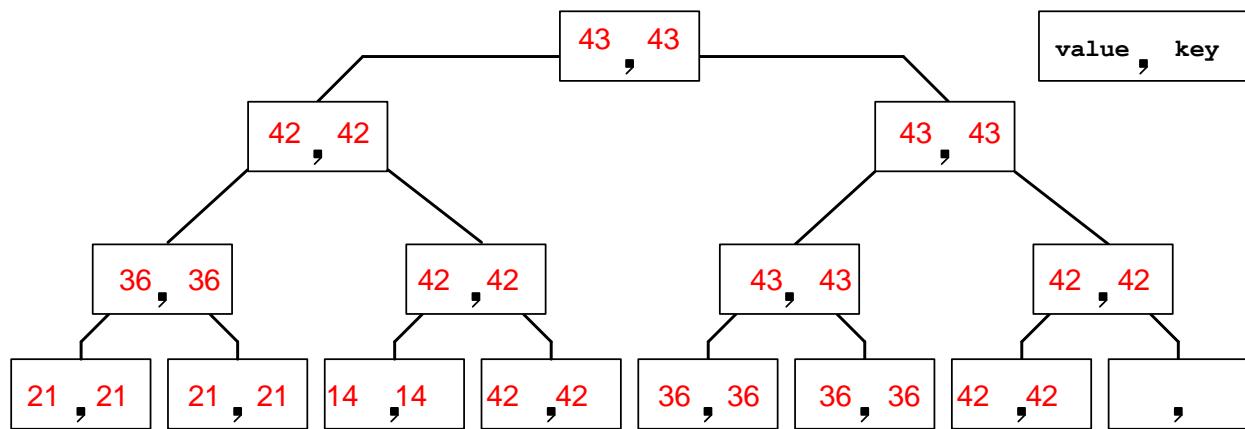


1.5.2) (1 pt) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de `value` et `key` de l'objet `Entry` contenues dans le monceau.

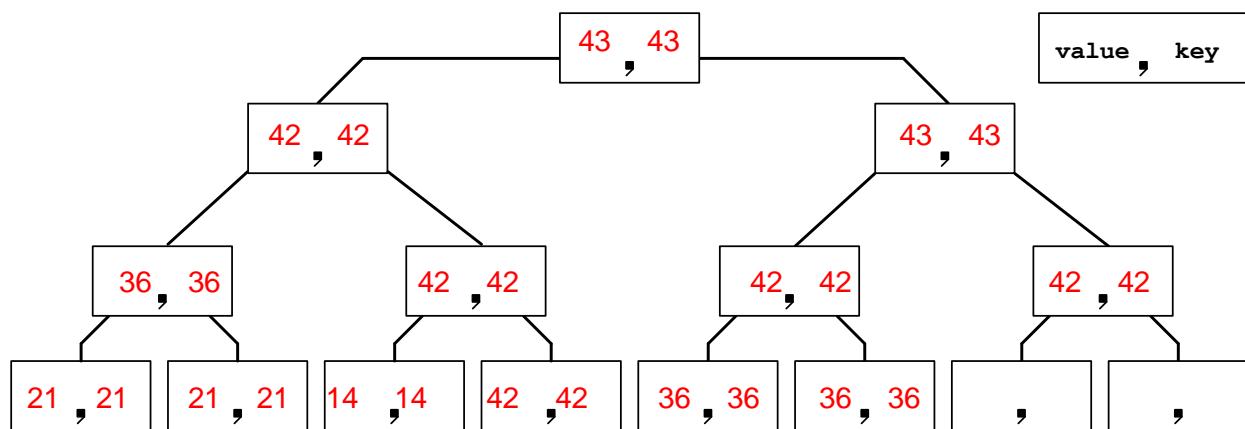


1.6) Dessinez l'état du monceau de la question 1.4) suite à deux appels consécutifs à `deleteRoot()` :

1.6.1) (1 pt) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de `value` et `key` de l'objet **Entry** contenues dans le monceau.

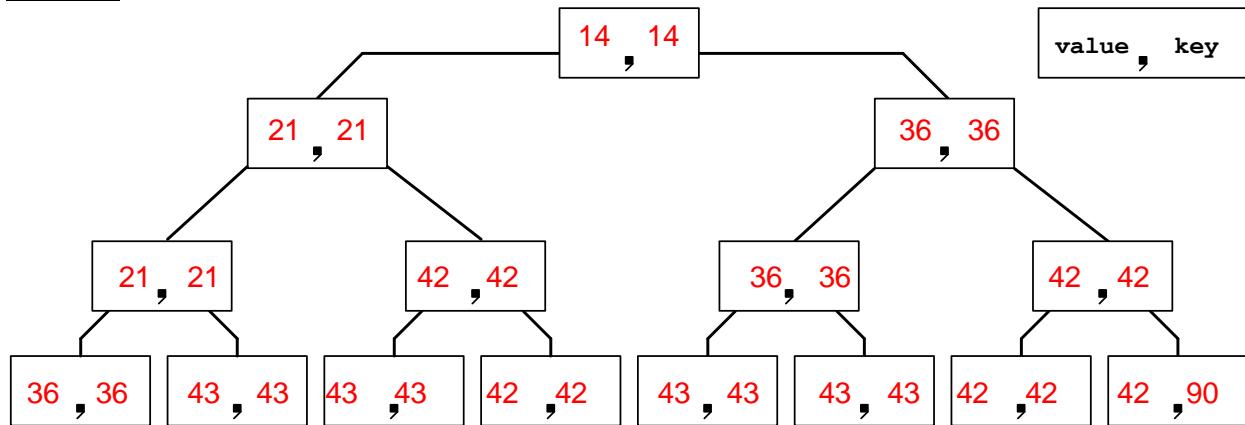


1.6.2) (1 pt) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de `value` et `key` de l'objet **Entry** contenues dans le monceau.



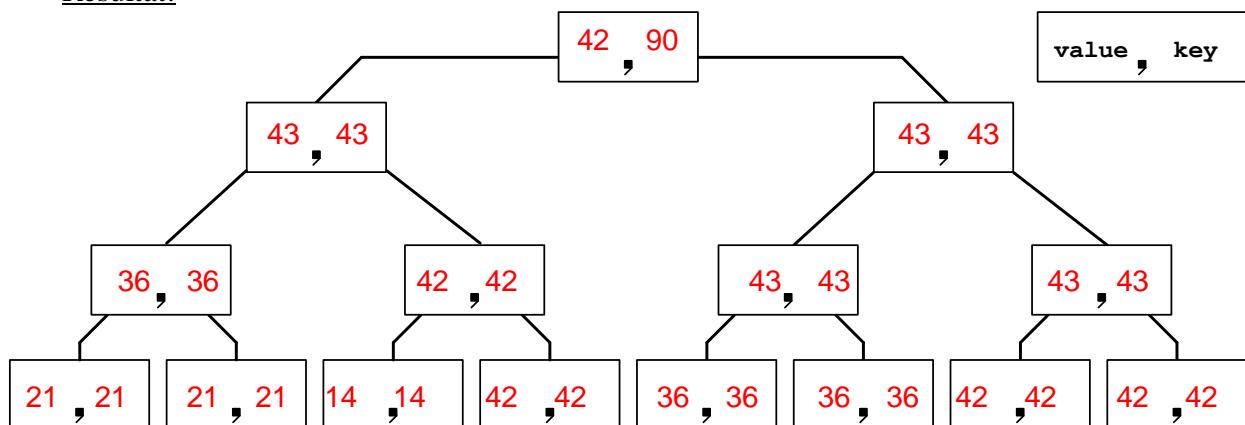
1.7) (4 pts) Exécutez `changeKey(15, 90)` sur le monceau obtenu en 1.3) :

Résultat:



1.8) (4 pts) Exécutez `changeKey(15, 90)` sur le monceau obtenu en 1.4) :

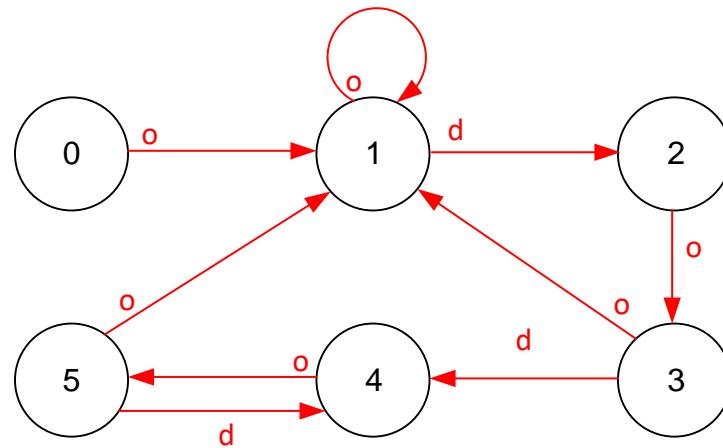
Résultat:



Question 2 : Recherche de patron**(12 points)**

On vous demande de retrouver le patron $P[1:5] = \text{« ododo »}$ dans un texte.

2.1) (3 pts) Dessinez le diagramme d'états de l'automate à états finis permettant de ce faire :



2.2) (3 pts) Donnez la table de transitions de l'automate recherché.

q\a	d	o	Autre
0	0	1	0
1	2	1	0
2	0	3	0
3	4	1	0
4	0	5	0
5	2	4	0

3. (6 pts) Quels sont respectivement le ou les états les plus visités et le ou les états les moins visités par l'automate une fois que le texte suivant aura été traité. L'automate débute à l'état 0.

$T[1 : 21] = \text{« odododododddododododo »}$

Aidez-vous de la table suivante (inscrivez l'état où se retrouve la machine en lisant le caractère $T[i]$):

$T[i]$	o	d	o	d	o	d	o
q	1	2	3	4	5	4	5
$T[i]$	d	o	d	d	d	o	d
q	4	5	4	0	0	1	2
$T[i]$	o	d	o	d	o	d	o
q	3	4	5	4	5	4	5

Le ou les états les plus visités : 4

Le ou les états les moins visités : 0, 1, 2, 3

Question 3 : Programmation dynamique**(20 points)**

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 2 \times 1 ; A_2 : 1 \times 4; A_3 : 4 \times 3; A_4 : 3 \times 1; A_5 : 1 \times 2$$

Considérez les tables **m** et **s** obtenues par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	8	18	17	21
2		0	12	15	17
3			0	12	18
4				0	6
5					0

s	1	2	3	4	5
1		1	1	1	1/4
2			2	3	4
3				3	4
4					4
5					

Complétez cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

3.1) (3 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_3 . Donnez son coût.

Parenthésage optimal: $A_1 ((A_2 A_3) A_4)$

Coût: **17**

3.2) (3 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal: $((A_2 A_3) A_4) A_5$

Coût: **17**

3.3) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $A_1 (((A_2 A_3) A_4) A_5)$ ou $(A_1 ((A_2 A_3) A_4)) A_5$

Coût: **21**

3.4 (10 pts) Si A_4 , et A_5 étaient des matrices 3×5 et 5×2 respectivement, quel serait le parenthésage optimal pour multiplier A_1 à A_5 ? Quel serait son coût? Aidez-vous des matrices suivantes pour répondre à la question.

m	1	2	3	4	5
1	0	8	18	37	41
2		0	12	27	37
3			0	60	54
4				0	30
5					0

s	1	2	3	4	5
1		1	1	1	1
2			2	3	4
3				3	3
4					4
5					

Parenthésage optimal: $A_1 ((A_2 A_3) A_4) A_5$

Coût: 41

Question 4 : Ordre topologique (24 points)

- 4.1) (5 pts) Donnez l'ordre topologique du graphe $G = \{V, E\}$ suivant en appliquant l'algorithme utilisant une file tel que vu en classe.

$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(a, f), (c, d), (f, d), (f, g), (e, b), (e, c)\}$$

Nœud	1	2	3	4	5	6	7
a	0	-	-	-	-	-	-
b	1	1	0	-	-	-	-
c	1	1	0	-	-	-	-
d	2	2	2	1	0	0	-
e	0	-	-	-	-	-	-
f	1	0	-	-	-	-	-
g	1	1	1	0	-	-	-
Entrée	a, e	f	b, c	g	-	d	-
Sortie	a	e	f	b	c	g	d

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	1	4	5	7	2	3	6

- 4.2) (6 pts) Ce graphe admet au moins trois autres ordres topologiques. Donnez-les.

Note : D'autres tables vous sont fournies à l'Annexe 2. Utilisez-les si nécessaire.

Ordre alternatif #1 :

Nœud	a	b	c	d	e	f	g
Ordre:	1	5	4	7	2	3	6

Ordre alternatif #2 :

Nœud	a	b	c	d	e	f	g
Ordre:	2	3	4	6	1	5	7

Ordre alternatif #3 :

Nœud	a	b	c	d	e	f	g
Ordre:	2	3	4	7	1	5	6

4.3) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant le parcours DFS post-ordre inverse. Partez du nœud a et visitez les nœuds alphabétiquement.

4.3.1) (3 pts) Donnez l'affichage DFS post-ordre obtenu :

d, g, f, a, b, c, e

4.3.2) (1 pt) Donnez l'affichage DFS post-ordre inverse obtenu :

e, c, b, a, f, g, d

4.3.3) (1 pt) Donnez l'ordre topologique trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre:	4	3	2	7	1	5	6

4.4) (3 pts) Combien de composantes fortement connexes G contient-il? Donnez-les.

7, chacun des nœuds

4.5) (5 pts) Combien de composantes fortement connexes contiendrait G si on lui ajoutait les arcs (d, a) et (a, e)? Donnez-les.

3 :

a, c, d, e, f

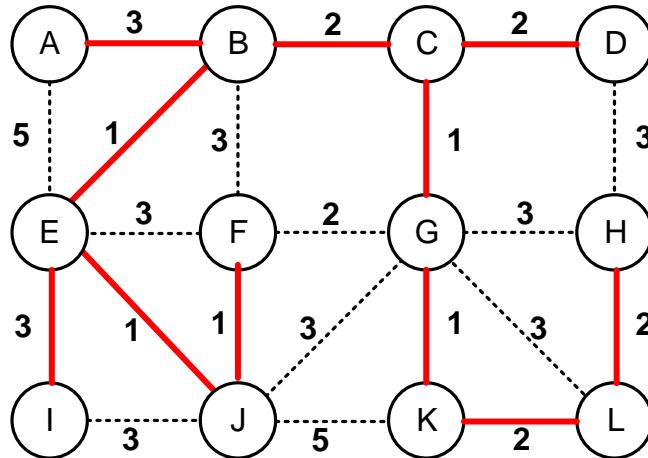
b

g

Question 5 : Arbre sous-tendant minimum (14 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Prim (le noeud de départ étant A) et de Kruskal. Respectez l'ordre alphabétique pour effectuer vos traitements (pour visiter les composantes, les nœuds voisins ou les arêtes). Aidez-vous des tables fournies pour ce faire.

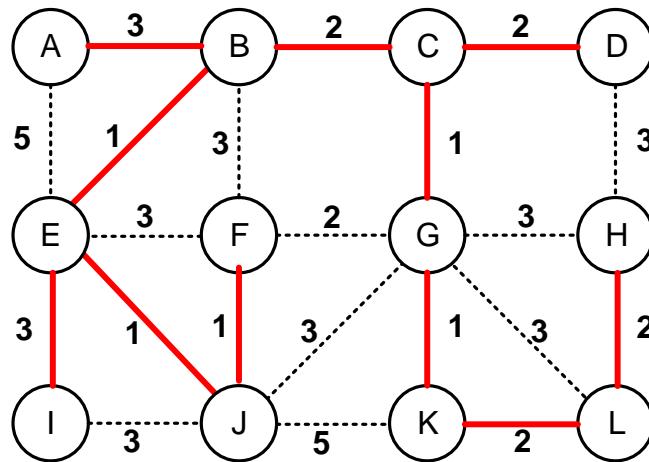
5.1) (7 pts) Par Prim:



Prim:

Nœud	Distance	Parent	Connu?
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			

5.2) (7 pts) Par Kruskal:



Kruskal:

Arête	Poids	Retenue?
(B, E)	1	
(C, G)	1	
(E, J)	1	
(F, J)	1	
(G, K)	1	
(B, C)	2	
(C, D)	2	
(F, G)	2	
(H, L)	2	
(K, L)	2	
(A, B)	3	
(B, F)	3	
(D, H)	3	
(E, F)	3	
(E, I)	3	
(G, H)	3	
(G, J)	3	
(G, L)	3	
(I, J)	3	
(A, E)	5	
(J, K)	5	

Question 6 : Généralités (10 points)

Répondez aux assertions suivantes par « vrai » ou par « faux ». Justifier votre réponse brièvement. Les réponses non justifiées ne seront pas considérées.

6.1) (2 pts) Soit $G = (V, E)$ un graphe dirigé. G possède au moins $|V| - 2$ composantes fortement. Alors G^T (le graphe transposé de G) possède également $|V| - 2$ composantes fortement connexes.

Vrai. Les composantes connexes se conservent lors de la transposition du graphe.

6.2) (2 pts) La solution vue en classe au problème de parenthésage des matrices est un algorithme de type diviser pour régner.

Faux. Il s'agit d'une approche par programmation dynamique.

6.3) (2 pts) Partant d'un sommet quelconque d'un graphe non dirigé, le parcours DFS permet de statuer sur la connexité du graphe (c.-à-d. qu'il permet d'affirmer que le graphe est connexe ou pas).

Vrai. Si le graphe est connexe, un DFS visitera tous les sommets sans interruption. C'est d'ailleurs l'algorithme vu en classe.

6.4) (2 pts) La structure de données d'ensembles disjoints vue en classe permet d'identifier l'ensemble auquel appartient un élément en $O(\lg(n))$ dans le meilleur cas.

Faux. $O(1)$ dans le meilleur cas.

6.5) (2 pts) La structure de données d'ensembles disjoints vue en classe permet d'identifier l'ensemble auquel appartient un élément en $O(\lg(n))$ dans le pire cas.

Faux, $O(n)$ dans le pire cas.



POLYTECHNIQUE
MONTRÉAL

Corrigé
examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Q7	
Total	

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20183
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo, responsable Tarek Ould Bachir, chargé de cours			5193
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Vendredi	14 décembre 2018	2 h 30	9 h 30 – 12 h 00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.
<i>Directives particulières</i>			
<input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.			
Important	Cet examen contient 7 questions sur un total de 14 pages (excluant cette page) La pondération de cet examen est de 40 % Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(20 points)**

Pour cette question, référez-vous au code Java donné à l'Annexe 1. Il s'agit de l'implémentation d'un monceau min telle que vue en classe (Weiss) et augmentée de la méthode :

```
public String printArray() {
    StringBuilder sb = new StringBuilder();
    for (int i=1; i<currentSize; i++ )
        sb.append(array[i].toString());
    return sb.toString();
}
```

Considérez la méthode `main` de l'Annexe 1 qui manipule un monceau `h` sur des objets de type `Character` ('A' < 'B' < 'C' < ...) et reproduisez les cinq (5) affichages qui y sont indiqués pour chacune des questions suivantes, tel que les exécute la méthode `main`.

Utilisez les cases du tableau pour écrire un caractère, comme illustré dans l'exemple suivant, résultant du 1^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 54 de la page 11 en Annexe 1 :

E	X	P											
---	---	---	--	--	--	--	--	--	--	--	--	--	--

Q1.1) **(4 points)** Donnez le résultat du 2^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 11 de la page 12 en Annexe 1.

A	E	E	X	M	N								
---	---	---	---	---	---	--	--	--	--	--	--	--	--

Q1.2) **(4 points)** Donnez le résultat du 3^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 19 de la page 12 en Annexe 1.

M	X	N											
---	---	---	--	--	--	--	--	--	--	--	--	--	--

Q1.3) **(4 points)** Donnez le résultat du 4^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 30 de la page 12 en Annexe 1.

A	F	N	I	L									
---	---	---	---	---	--	--	--	--	--	--	--	--	--

Q1.4) **(4 points)** Donnez le résultat du 5^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 38 de la page 12 en Annexe 1.

L	N												
---	---	--	--	--	--	--	--	--	--	--	--	--	--

Q1.5) **(4 points)** Donnez le résultat du 6^e affichage, soit le résultat de l'appel à `System.out.println(h.printArray());` à la ligne 46 de la page 12 en Annexe 1.

A	E	A	I	E	N	F	M	N	X	L			
---	---	---	---	---	---	---	---	---	---	---	--	--	--

Question 2 : Recherche de patron**(20 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier via un calcul de hachage. On considère l'alphabet $\Sigma = \{0, 1, 2, 3\}$ où $d = |\Sigma| = 4$. On admettra l'encodage suivant :

Symbol	0	1	2	3
Code	0	1	2	3

Le hachage est calculé en base $d = 4$ modulo $q = 17$. Par exemple, la séquence "2033010", encodée **2, 0, 3, 3, 0, 1, 0** produira une valeur de hash :

$$\begin{aligned} \text{mod}((((((2 \cdot d + 0) \cdot d + 3) \cdot d + 3) \cdot d + 0) \cdot d + 1) \cdot d + 0, q) &= \\ \text{mod}((((((2 \cdot 4 + 0) \cdot 4 + 3) \cdot 4 + 3) \cdot 4 + 0) \cdot 4 + 1) \cdot 4 + 0, 17) &= \\ &= 10. \end{aligned}$$

Sachant que pour deux entiers a et b pris dans l'intervalle $[0, 2^8]$,

$$\text{mod}(2^8 \cdot a + b, 17) = \text{mod}(a + b, 17)$$

Sachant que "1030" produit un hash de 8.

Sachant que "3010" produit un hash de 9.

2.1) (4 points) Donnez le hash p produit par $P[1:8] = "10303010"$. Justifiez votre réponse par un calcul.

$$\begin{aligned} p &= \text{mod}(4^8 + 9, 17) \\ &= \text{mod}(2^8 \cdot 4 + 9, 17) \\ &= \text{mod}(8 + 9, 17) = 0 \end{aligned}$$

Afin d'accélérer les calculs lors de la recherche du patron dans le texte, Rabin Karp exploite une formule récursive : $t_{s+1} = \text{mod}((t_s - hT[s+1])d + T[s+m+1], q)$, où $h = \text{mod}(d^{m-1}, q)$.

2.2) (4 points) Donnez la valeur de la variable h pour le patron $P[1 : 8]$

$$\begin{aligned} h &= \text{mod}(4^7, 17) = \text{mod}(4^4 \cdot 4^3, 17) = \text{mod}(2^8 \cdot 4^3, 17) = \\ &= \text{mod}(4^3 + 0, 17) = \text{mod}(2^6, 17) = \text{mod}(64, 17) = 13 \end{aligned}$$

Supposons maintenant que nous recherchions un patron P de longueur $m= 17$ dans un texte T de longueur $n = 19$;

- En admettant que le hash associé à P est $p = 0$;
- En admettant que $h = \text{mod}(d^{m-1}, q) = \text{mod}(4^{16}, 17) = 1$;
- En admettant que P débute par "03" ($P = "03\dots"$) ;
- En admettant que T est la concaténation de P et de "03", c.-à-d. $T = P \cdot "03"$.

2.3) (3 points) Donnez le hash t_0 produit par $T[1:m]$ en justifiant le résultat par un calcul.

$$t_0 = p = 0$$

2.4) (3 points) Donnez le hash t_1 produit par $T[2:m+1]$ en justifiant le résultat par un calcul.

$$t_1 = \text{mod}((0 - 1 \cdot 0) \cdot 4 + 0, 17) = 0$$

2.5) (3 points) Donnez le hash t_2 produit par $T[3:m+2]$ en justifiant le résultat par un calcul.

$$t_2 = \text{mod}((0 - 1 \cdot 3) \cdot 4 + 3, 17) = 8$$

2.6) (3 points) Combien de faux positifs seront détectés en recherchant P dans T ?

Un seul faux positif est détecté, il s'agit de $T[2:m+1]$ qui débute par '3' mais produit un hash égal à p (P débute par '0').

Question 3 : Programmation dynamique (16 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 1 \times 2 ; A_2 : 2 \times 3; A_3 : 3 \times 3; A_4 : 3 \times 4; A_5 : 4 \times 1$$

Considérez les tables **m** et **s** obtenues par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	6	15	27	29
2		0	18	42	27
3			0	36	21
4				0	12
5					0

s	1	2	3	4	5
1		1	2	3	1
2			2	3	2
3				3	3
4					4
5					

Complétez cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

Donnez le parenthésage optimal pour multiplier A_1 à A_5 et son coût

Parenthésage optimal: $A_1 (A_2 (A_3 (A_4 A_5)))$

Coût: 29

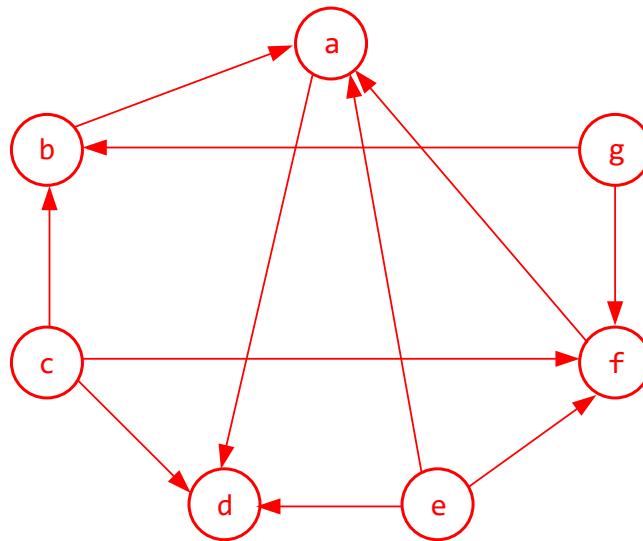
Question 4 : Ordre topologique (16 points)

On veut connaître l'ordre topologique du graphe dirigé acyclique suivant :

$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(a, d), (b, a), (c, b), (c, d), (c, f), (e, a), (e, d), (e, f), (f, a), (g, b), (g, f)\}$$

4.1) (2 pts) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (6 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant une liste de travail vu en classe. Utilisez une file (FIFO) comme liste de travail. Visitez les voisins par ordre alphabétique.

Nœud	1	2	3	4	5	6	7
a	3	3	2	2	1	0	-
b	2	1	1	0	-	-	-
c	0	-	-	-	-	-	-
d	3	2	1	1	1	1	0-
e	0	-	-	-	-	-	-
f	3	2	1	0	-	-	-
g	0	-	-	-	-	-	-
Nouveaux nœuds	c, e, g	-	-	b, f	-	a	d
Nœud retiré	c	e	g	b	f	a	d

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	6	4	1	7	2	5	3

4.3) (4 pts) Aurions-nous pu utiliser une pile (LIFO) à la place de la file pour exécuter l'algorithme précédent et obtenir un résultat correct ? Justifiez brièvement.

Oui, c'est possible. L'ordre d'entrée et de sortie dans la liste de travail n'importe pas.

4.4) (4 pts) Proposez un ordre topologique alternatif à G :

Nœud	a	b	c	d	e	f	g
Ordre :	6	4	2	7	1	5	3

Il en existe d'autres aussi.

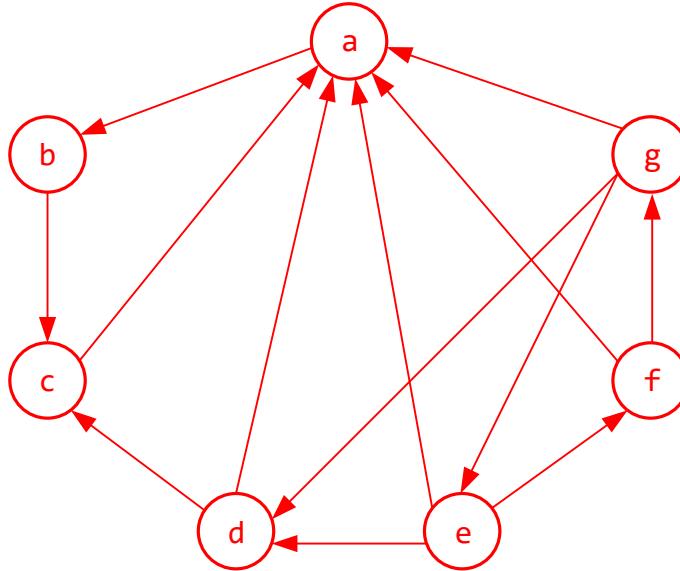
Question 5 : Composantes fortement connexes (12 points)

On veut connaître les composantes fortement connexes du graphe dirigé suivant :

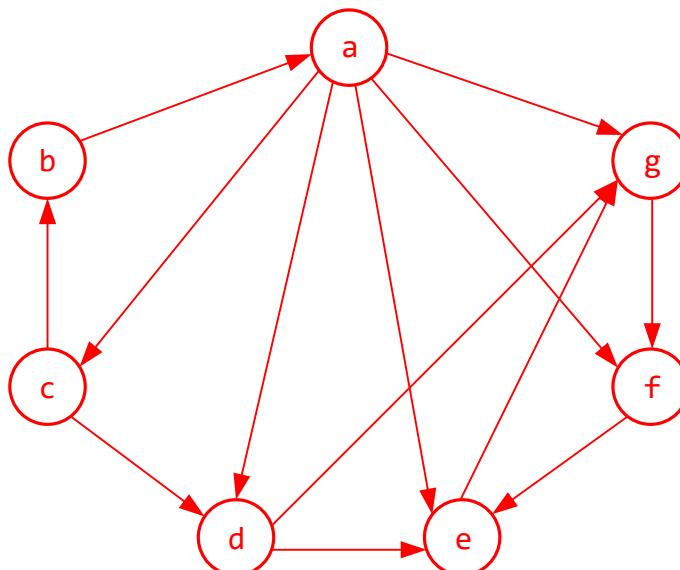
$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(a, b), (b, c), (c, a), (d, a), (d, c), (e, a), (e, d), (e, f), (f, a), (f, g), (g, a), (g, d), (g, e)\}$$

5.1) (2 points) Reproduisez graphiquement le graphe $G = (V, E)$:



5.2) (2 points) Donnez G^T , le graphe transposé de G :



5.3) (4 points) Donnez les composantes fortement connexes de G en associant chacun des nœuds a à g à une composante. Les composantes sont numérotées incrémentalement et la numérotation débute à 1. Laissez les colonnes inutilisées vides.

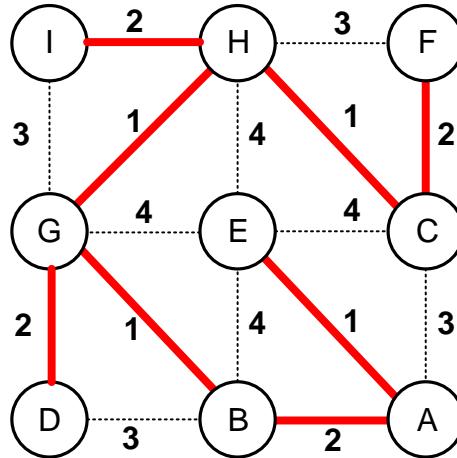
Nœud	Composante						
	1	2	3	4	5	6	7
a	X						
b	X						
c	X						
d		X					
e			X				
f			X				
g			X				

5.4) (4 points) Est-il vrai que le graphe G^T admet les mêmes composantes fortement connexes que G ? Expliquez brièvement pourquoi ?

C'est vrai. La transposition d'un graphe conserve les cycles, et conserve les composantes connexes par conséquent.

Question 6 : Arbre sous-tendant minimum (11 points)

6.1) (6 points) Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal en noircissant les arêtes retenues dans le graphe ci-après. Donnez le coût de l'arbre ainsi obtenu.



Coût: 12

Kruskal :

Arête	Retenue?
(A, B)	✓
(A, C)	
(A, E)	✓
(B, D)	
(B, E)	
(B, G)	✓
(C, E)	
(C, F)	✓
(C, H)	✓
(D, G)	✓
(E, G)	
(E, H)	
(F, H)	
(G, H)	✓
(G, I)	
(H, I)	✓

6.2) (3 points) Quelle structure de données serait appropriée pour choisir l'arête de plus faible poids dans l'exécution de l'algorithme de Kruskal par ensembles disjoints ?

Monceau pour choisir le minimum.

6.3) (2 points) Le graphe précédent admet-il un autre arbre sous-tendant minimum? Si oui, dites lequel.

OUI :	NON : ✓
Arbre alternatif :	

Question 7 : Compréhension de concepts**(5 points)**

Soit l'algorithme suivant, s'exécutant sur un graphe non dirigé $G = (V, E)$:

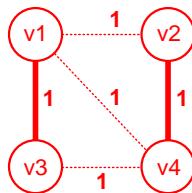
1. $A = \{ \}$
2. Pour chaque sommet v dans V
3. Sélectionner l'arête de plus faible poids e_{\min} parmi les arêtes reliant v à ses voisins
4. $A = A \cup \{ e_{\min} \}$
5. Retourner $T = (V, A)$

Remarques : Si à l'étape 3, plusieurs arêtes de plus faible poids se présentent, l'algorithme choisit une parmi elles au hasard.

Répondez aux questions suivantes en justifiant brièvement vos réponses à chaque fois.

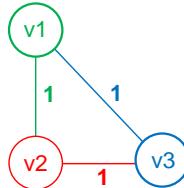
7.1) (1 points) Le graphe T est-il NÉCESSAIREMENT connexe ? Justifiez brièvement.

Pas nécessairement. Voici un contre-exemple.



7.2) (1 points) Le graphe T peut-il contenir un ou plusieurs cycles ? Justifiez brièvement.

Oui. Soit un graphe connexe à 3 sommets et à arêtes unitaires. Si on prend une arête différente pour chaque sommet, on crée un cycle.



7.3) (1 points) Si le graphe T est acyclique, peut-il être une forêt d'arbres ? Justifiez brièvement.

Oui, voir exemple de 7.1.

7.4) (1 points) Si le graphe T est connexe et acyclique, peut-il être un arbre ? Justifiez brièvement.

Oui. En vérité, si T est connexe et acyclique, il est forcément un arbre.

7.5) (1 points) Si le graphe T est connexe et acyclique, est-il toujours un arbre sous-tendant minimum ? Justifiez brièvement.

Oui. Si T est connexe et acyclique, il est forcément un arbre. Il est alors un arbre sous-tendant minimum par construction.



POLYTECHNIQUE
MONTRÉAL

Corrigé examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :		Prénom :
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre		
INF2010 – Structures de données et algorithmes		Tous	20171		
Professeur			Local		
Ettore Merlo, responsable		B-638	5193		
Tarek Ould Bachir, chargé de cours					
Jour	Date	Durée	Heures		
Vendredi	05 mai 2017	2 h 30	9 h 30 – 12 h 00		
Documentation		Calculatrice			
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Aucune	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.			
<input type="checkbox"/> Toute	<input type="checkbox"/> Toutes				
<input checked="" type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable				
Directives particulières					
<p><input type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.</p>					
Important	Ce corrigé contient 6 questions sur un total de 13 pages (excluant cette page)				
	La pondération de cet examen est de 40 %				
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux				
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non				

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux **(20 points)**

Pour cette question, référez-vous au code Java donné à l'Annexe 1.

- 1.1) (1 pts)** Donnez la complexité asymptotique en cas moyen d'un `findMin()`.

O(1)

- 1.2) (1 pts)** Donnez la complexité asymptotique en cas moyen d'un `deleteMin()`.

O(lg(n))

- 1.3) (1 pts)** Donnez la complexité asymptotique en cas moyen d'un `buildHeap()`.

O(n)

- 1.4) (1 pts)** Donnez la complexité asymptotique en pire cas d'un `buildHeap()`.

O(n)

- 1.5) (1 pts)** Donnez la complexité asymptotique en cas moyen d'un `insert(AnyType x)`.

O(1)

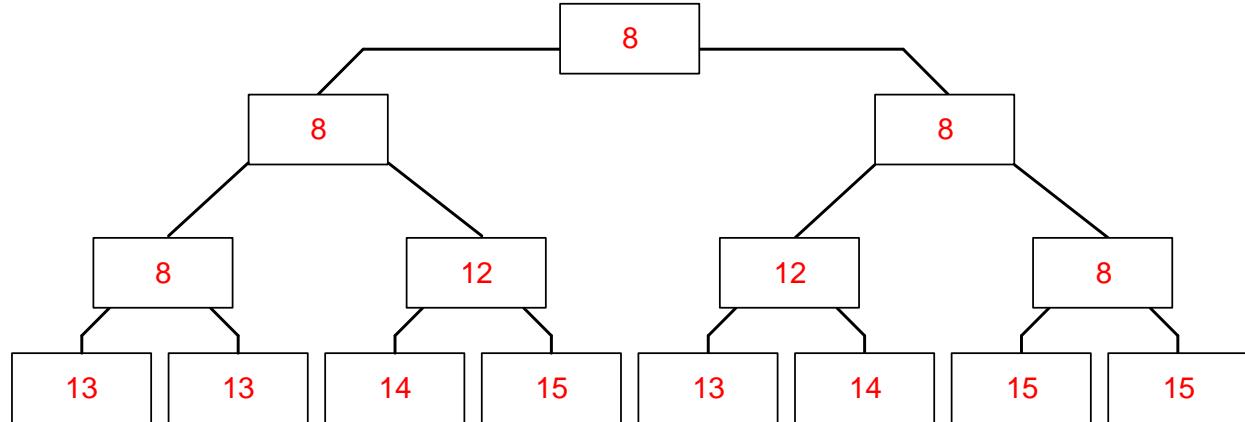
- 1.6) (1 pts)** Donnez la complexité asymptotique en pire cas d'un `insert(AnyType x)`.

O(lg(n))

1.7) (2 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après.

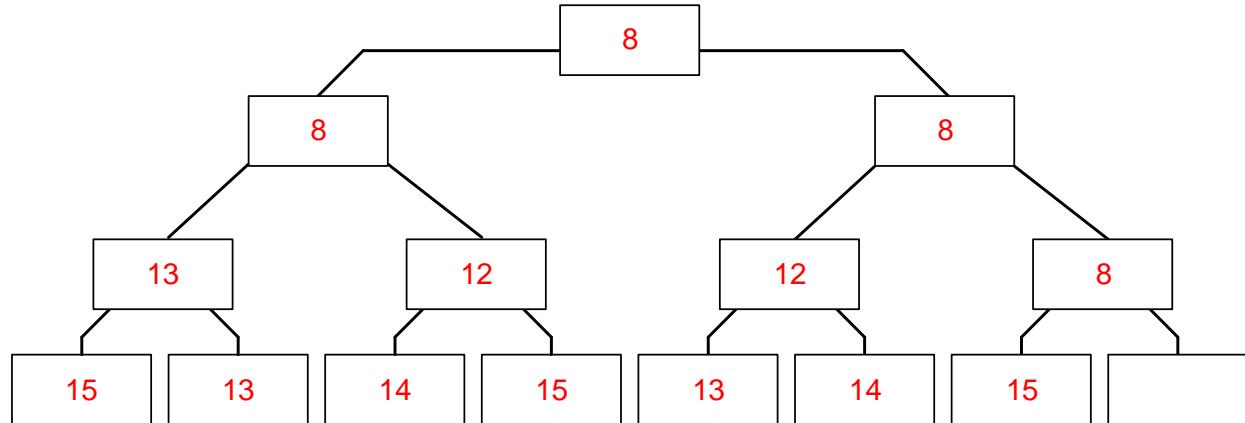
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	8	8	8	8	12	12	8	13	13	14	15	13	14	15	15

Votre réponse :



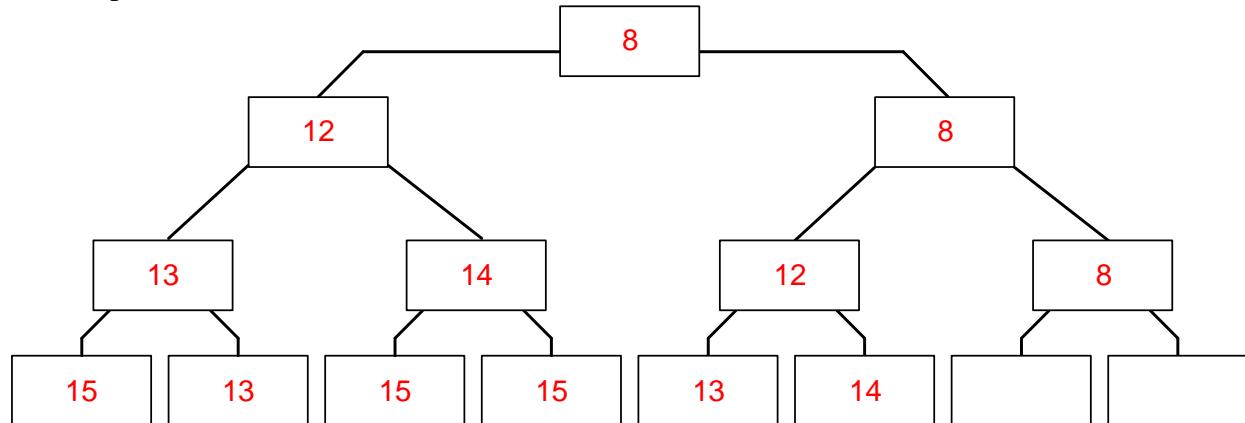
1.8) (2 pts) En partant du monceau de 1.7), effectuez un `deleteMin()`.

Votre réponse :



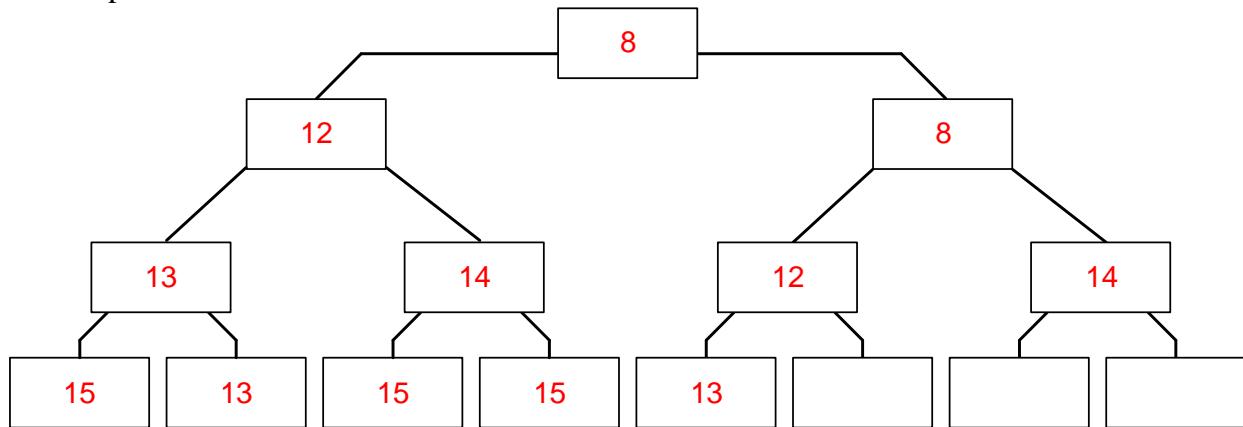
1.9) (2 pts) En partant du monceau de 1.8), effectuez un `deleteMin()`.

Votre réponse :



1.10) (2 pts) En partant du monceau de 1.9), effectuez un `deleteMin()`.

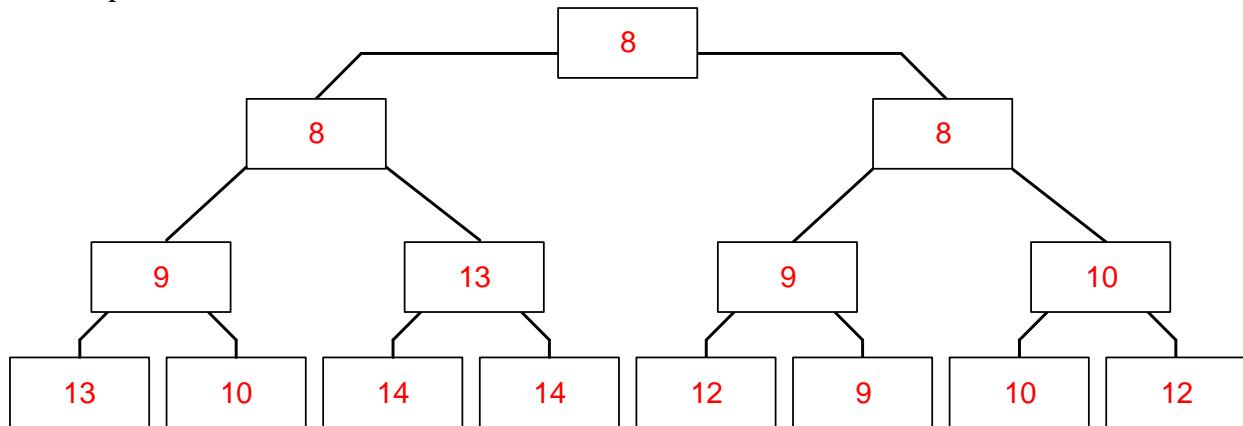
Votre réponse :



1.11) (2 pts) Dessinez le monceau résultant d'appel à `BinaryHeap(AnyType[] items)` où `items` est le tableau suivant :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	13	14	12	13	8	9	10	9	10	8	14	8	9	10	12	

Votre réponse :



- 1.12) (4 pts)** Dessinez le monceau résultant d'appel à `BinaryHeap(AnyType[] items)` où `items` est le tableau suivant :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	13	14	12	13	8	9	10	9	10	8	14	8	9	10	12	

Et où la méthode `percolateDown` a été modifiée comme suit :

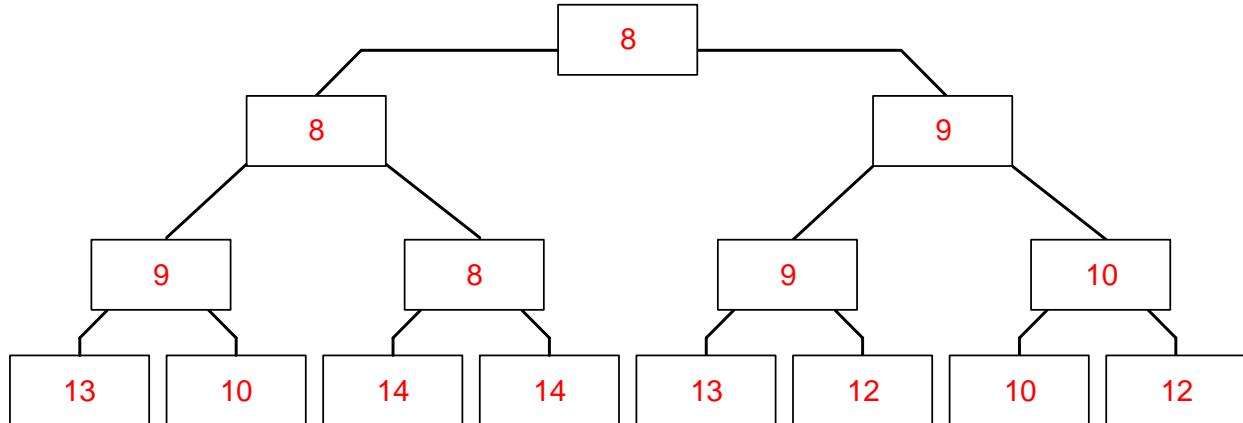
```
private void percolateDown( int hole ) {
    int child;
    AnyType tmp = array[ hole ];

    for( ; hole * 2 <= currentSize; hole = child ) {
        child = hole * 2;

        if( child != currentSize &&
            array[ child ].compareTo( array[ child+1 ] ) >= 0 )
            child++;

        if( array[ child ].compareTo( tmp ) < 0 )
            array[ hole ] = array[ child ];
        else
            break;
    }
    array[ hole ] = tmp;
}
```

Votre réponse :



Question 2 : Recherche de patron**(20 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier via un calcul de hachage. Une implémentation de Rabin Karp vous est proposée à l'Annexe 2. On y considère l'alphabet $\Sigma = \{G, C, A, T\}$ où $d = |\Sigma| = 4$. On admettra l'encodage suivant :

Symbol	G	C	A	T
Code	0	1	2	3

Le hachage est calculé en base d modulo $q = 2^7 - 1 = 127$. Par exemple, la séquence « GATAC », encodée 0, 2, 3, 2, 1, produira une valeur de hash :

$$\text{mod}((0 \cdot d + 2) \cdot d + 3) \cdot d + 2) \cdot d + 1, q) = 58.$$

2.1) (2 pts) Donnez le hash associé au patron « ATGCTTC » :

86

Afin d'accélérer les calculs lors de la recherche du patron dans le texte, Rabin Karp exploite une formule récursive : $t_{s+1} = \text{mod}(t_s - hT[s])d + T[s+m+1], q)$, où $h = \text{mod}(d^{m-1}, q)$. Sachant que $\text{mod}(2^m, 2^p-1) = 2^{\text{mod}(m, p)}$, nous aurons

$$\begin{aligned} h &= \text{mod}(d^{m-1}, q) \\ &= \text{mod}(4^{m-1}, 127) \\ &= \text{mod}(2^{2(m-1)}, 2^7-1) \\ &= 2^{\text{mod}(2(m-1), 7)} \end{aligned}$$

2.2) (2 pts) Donnez la valeur de la variable h pour le patron « ATGCTTC » :

Votre réponse :

h = 32

2.3) (16 pts) Combien de faux positifs seront détectés si le patron « ATGCTTC » est recherché dans le texte « CGAATGCTTCTCAA ». Fiez-vous au code java donné à l’Annexe 2. Remarquez que dans l’implémentation donnée de la fonction updateHash, le calcul du modulo renvoie toujours une valeur positive.

```
private int updateHash(int oldHash, char oldChar, char newChar) {
    if( !ENCODING.containsKey( oldChar ) ) throw new IllegalArgumentException();
    if( !ENCODING.containsKey( newChar ) ) throw new IllegalArgumentException();

    int newHash = oldHash;
    newHash -= h*ENCODING.get( oldChar );
    if( newHash < 0 ) newHash += q; // valeur positive
    newHash *= d;
    newHash += ENCODING.get( newChar );
    newHash %= q;
    return newHash;
}
```

Vous pouvez vous aider de la table ci-après :

Décalage	Sous-séquence	t_s
0	CGAATGC	86
1	GAATGCT	92
2	AATGCTT	117
3	ATGCTTC	86
4	TGCTTCT	91
5	GCTTCTC	108
6	CTTCTCA	53
7	TTCTCAA	86

2.3.a) (4 pts) Nombre de faux positifs détectés :

2

2.3.b) (12 pts) Les sous-séquences associées aux faux positifs détectés ET les décalages correspondants :

Décalage 0, CGAATGC

Décalage 7, TTCTCAA

Question 3 : Programmation dynamique (12 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 3 \times 1 ; A_2 : 1 \times 3; A_3 : 3 \times 2; A_4 : 2 \times 1; A_5 : 1 \times 3$$

Considérez les tables **m** et **s** obtenues par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	9	12	11	20
2		0	6	8	11
3			0	6	15
4				0	6
5					0

s	1	2	3	4	5
1		1	1	1	1 ou 4
2			2	3	4
3				3	4
4					4
5					

Complétez cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

3.1) (3 pts) Donnez le parenthésage optimal pour multiplier A_3 à A_5 . Donnez son coût.

Parenthésage optimal: **(A_3A_4) A_5**

Coût: **15**

3.2) (4 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal: **((A_2A_3) A_4) A_5**

Coût: **11**

3.3) (5 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: **$A_1(((A_2A_3)A_4)A_5)$ ou $(A_1((A_2A_3)A_4))A_5$**

Coût: **20**

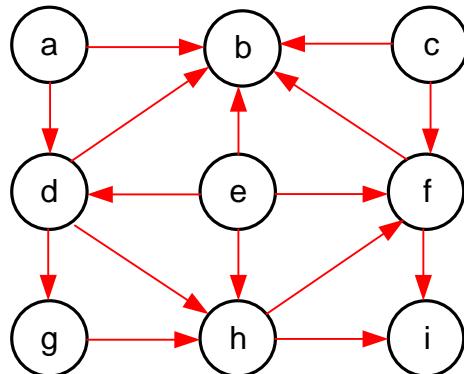
Question 4 : Ordre topologique (16 points)

On veut connaître l'ordre topologique du graphe suivant :

$$V = \{a, b, c, d, e, f, g, h, i\}$$

$$E = \{(a, b), (a, d), (c, b), (c, f), (d, b), (d, g), (d, h), (e, b), (e, d), (e, f), (e, h), (f, b), (f, i), (g, h), (h, f), (h, i)\}$$

4.1) (2 pts) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (7 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant une file vu en classe.

Nœud	1	2	3	4	5	6	7	8	9
a	0	-	-	-	-	-	-	-	-
b	5	4	3	2	1	1	1	0	-
c	0	-	-	-	-	-	-	-	-
d	2	1	1	0	-	-	-	-	-
e	0	-	-	-	-	-	-	-	-
f	3	3	2	1	1	1	0	-	-
g	1	1	1	1	1	0	-	-	-
h	3	3	3	2	1	0	-	-	-
i	2	2	2	2	2	2	1	0	-
Entrée	a,c,e	-	-	d	g	h	f	b,i	-
Sortie	a	c	e	d	g	h	f	b	i

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g	h	i
Ordre :	1	8	2	4	3	7	5	6	9

4.3) (7 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant le parcours DFS post-ordre inverse. Partez du nœud a et visitez les nœuds alphabétiquement.

4.3.a) (2 pts) Donnez l'affichage DFS post-ordre obtenu :

b, i, f, h, g, d, a, c, e

4.3.b) (2 pts) Donnez l'affichage DFS post-ordre inverse obtenu :

e, c, a, d, g, h, f, i, b

4.3.c) (3 pts) Donnez l'ordre topologique trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g	h	i
Ordre :	3	9	2	4	1	7	5	6	8

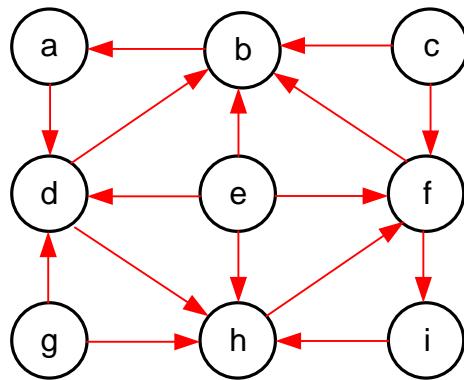
Question 5 : Composantes Fortement Connexes (18 points)

On veut connaître l'ordre topologique du graphe suivant :

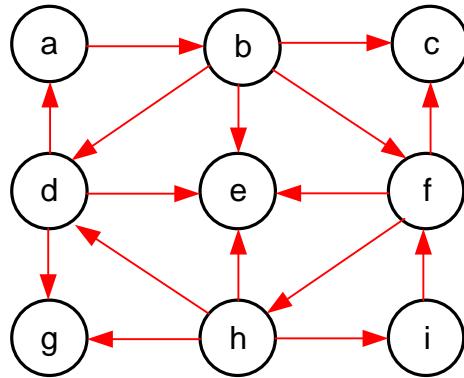
$$V = \{a, b, c, d, e, f, g, h, i\}$$

$$E = \{(a, d), (b, a), (c, b), (c, f), (d, b), (d, h), (e, b), (e, d), (e, f), (e, h), (f, b), (f, i), (g, d), (g, h), (h, f), (i, h)\}$$

5.1) (2 pts) Reproduisez graphiquement le graphe $G = (V, E)$:



5.2) (2 pts) Donnez le graphe G^T , c'est-à-dire le graphe transposé de G :



5.3) (4 pts) Donnez l'affichage DFS post-ordre inverse de G^T . Visitez les nœuds et les voisins dans l'ordre alphabétique.

a, b, f, h, i, d, g, e, c

5.4) (4 pts) Donnez l'affichage DFS post-ordre de G suivant l'ordre d'affichage obtenu en 5.3). Visitez les voisins dans l'ordre alphabétique.

b, i, f, h, d, a, g, e, c

5.5) (6 pts) En vous servant du parcours réalisé dans 5.4), identifiez les composantes fortement connexes du graphe $G = (V, E)$.

Note : Le fait que la table comporte 5 lignes ne signifie pas qu'il y ait 5 composantes fortement connexes. Laissez les lignes inutilisées vides.

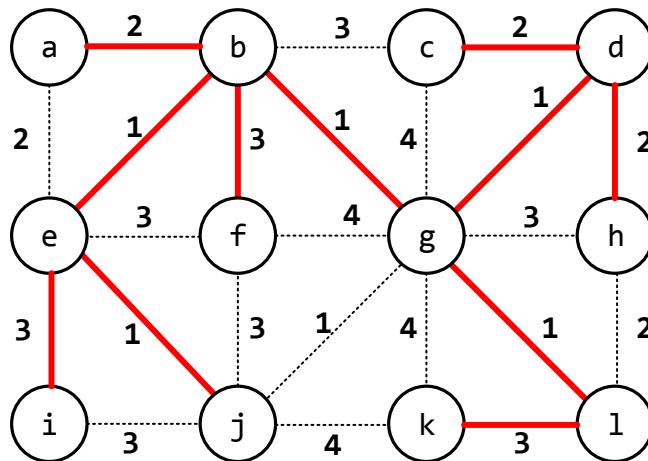
Composante	Nœuds
1	a, b, d, f, h, i
2	c
3	e
4	g
5	

Question 6 : Arbre sous-tendant minimum (14 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Prim (le nœud de départ étant a) et Kruskal en reliant les arêtes retenues dans les graphes données ci-après.

Respectez l'ordre alphabétique pour visiter les nœuds voisins ou les arêtes. Utilisez les tables fournies pour ce faire (le remplissage des tables ne compte pas dans l'attribution des points pour la question).

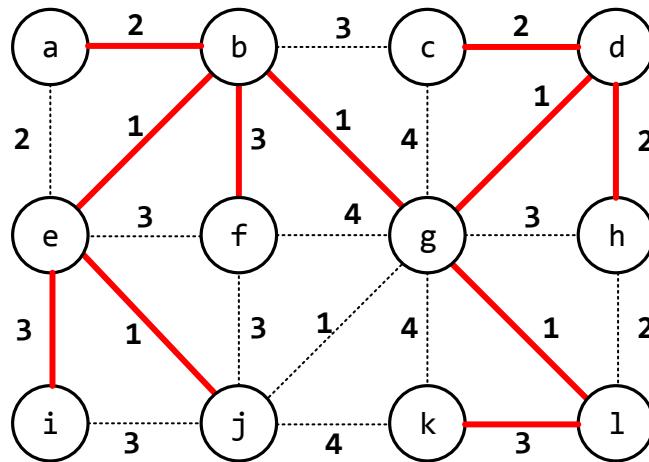
Par Prim (7 pts) :



Prim:

Nœud	Distance	Parent	Connu?
a			
b			
c			
d			
e			
f			
g			
h			
i			
j			
k			
l			

Par Kruskal (7 pts) :



Kruskal :

Arête	Poids	Retenue?
(a, b)	2	
(a, e)	2	
(b, c)	3	
(b, e)	1	
(b, f)	3	
(b, g)	1	
(c, d)	2	
(c, g)	4	
(d, g)	1	
(d, h)	2	
(e, f)	3	
(e, i)	3	
(e, j)	1	
(f, g)	4	
(f, j)	3	
(g, h)	3	
(g, i)	1	
(g, k)	4	
(g, l)	1	
(h, l)	2	
(i, j)	3	
(j, k)	4	
(k, l)	3	



POLYTECHNIQUE
MONTRÉAL

Corrigé examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Q6	
Total	

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 – Structures de données et algorithmes		Tous	20173
Professeur		Local	Téléphone
Ettore Merlo, responsable Tarek Ould Bachir, chargé de cours			5193
Jour	Date	Durée	Heures
Mardi	12 décembre 2017	2 h 30	9 h 30 – 12 h 00
Documentation		Calculatrice	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
Directives particulières			
<input type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.			
Important	Ce corrigé contient 6 questions sur un total de 14 pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux			
Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non			

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux (18 points)

Pour cette question, référez-vous au code Java donné à l'Annexe 1.

- 1.1) (1 pt)** Donnez la complexité asymptotique en pire cas d'un `findMin()`.

O(1)

- 1.2) (1 pt)** Donnez la complexité asymptotique en pire cas d'un `deleteMin()`.

O(lg(n))

- 1.3) (1 pt)** Donnez la complexité asymptotique en meilleur cas d'un `buildHeap()`.

O(n)

- 1.4) (1 pt)** Donnez la complexité asymptotique en pire cas d'un `buildHeap()`.

O(n)

- 1.5) (1 pt)** Donnez la complexité asymptotique en cas moyen d'un `insert(AnyType x)`.

O(1)

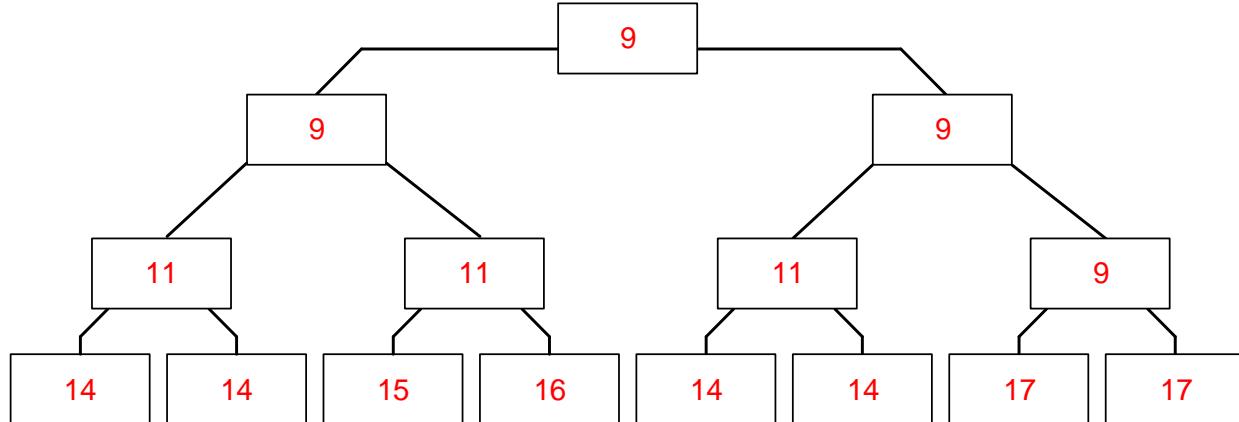
- 1.6) (1 pt)** Donnez la complexité asymptotique en meilleur cas d'un `insert(AnyType x)`.

O(1)

1.7) (1.5 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après.

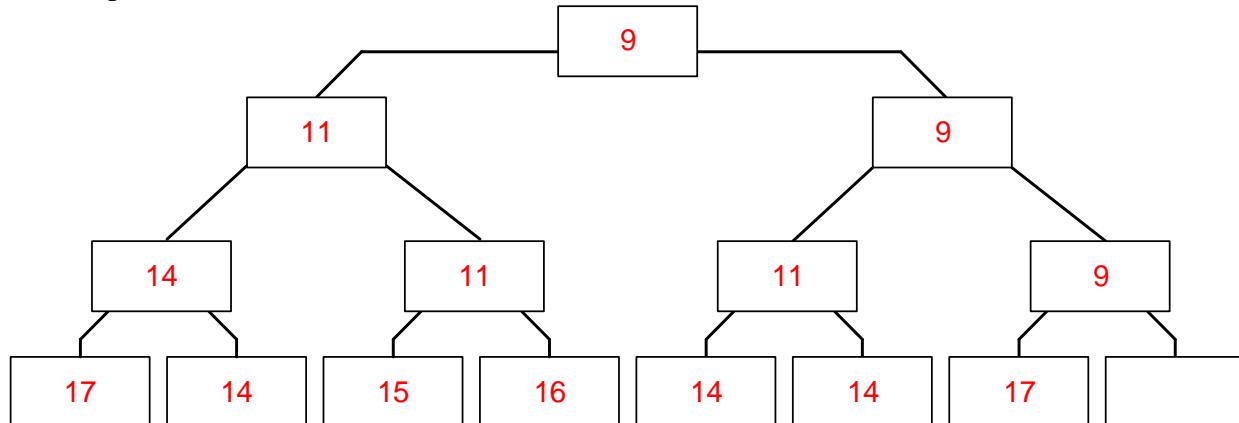
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	9	9	9	11	11	11	9	14	14	15	16	14	14	17	17

Votre réponse :



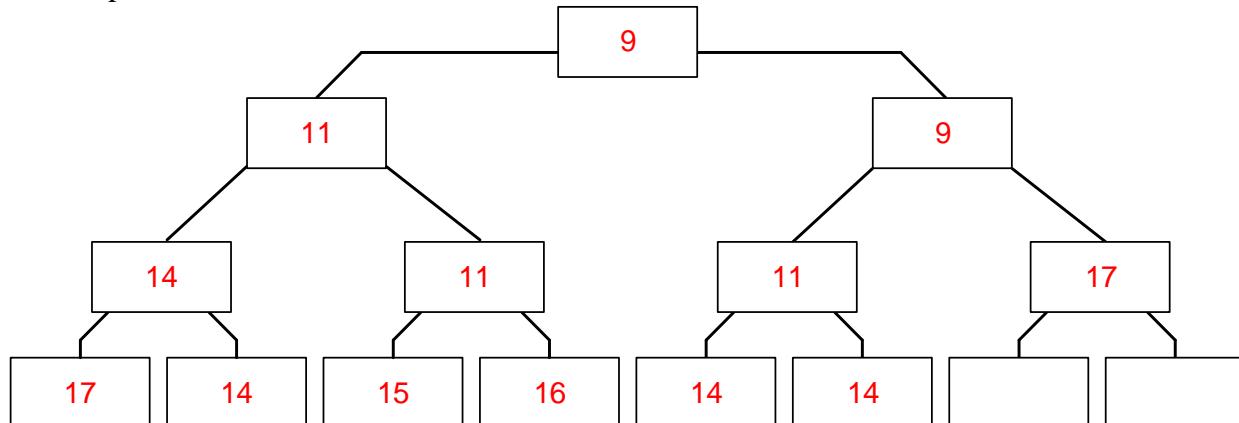
1.8) (1.5 pts) En partant du monceau de 1.7), effectuez un `deleteMin()`.

Votre réponse :



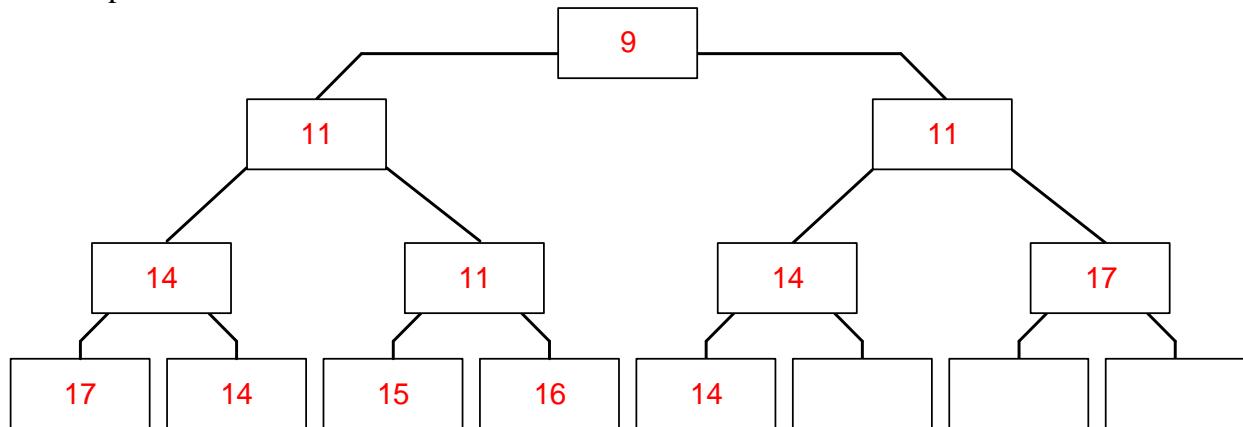
1.9) (1.5 pts) En partant du monceau de 1.8), effectuez un `deleteMin()`.

Votre réponse :



1.10) (1.5 pts) En partant du monceau de 1.9), effectuez un `deleteMin()`.

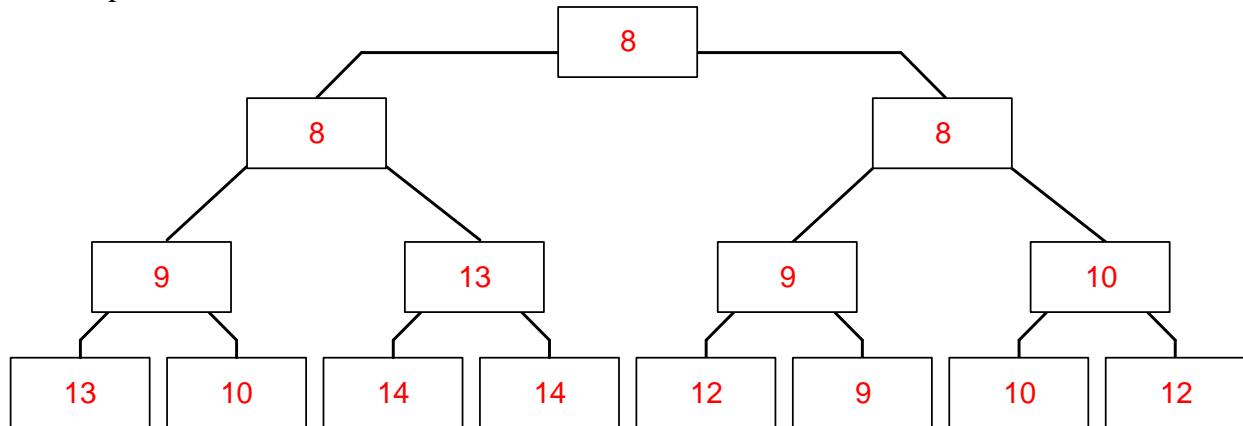
Votre réponse :



1.11) (2 pts) Dessinez le monceau résultant d'appel à `BinaryHeap(AnyType[] items)` où `items` est le tableau suivant :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	13	14	12	13	8	9	10	9	10	8	14	8	9	10	12	

Votre réponse :



- 1.12) (4 pts)** Dessinez le monceau résultant d'appel à `BinaryHeap` où `items` est le tableau suivant :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	16	15	11	14	9	10	11	10	11	7	15	9	10	11	13	

Et où la méthode `percolateDown` a été modifiée comme suit :

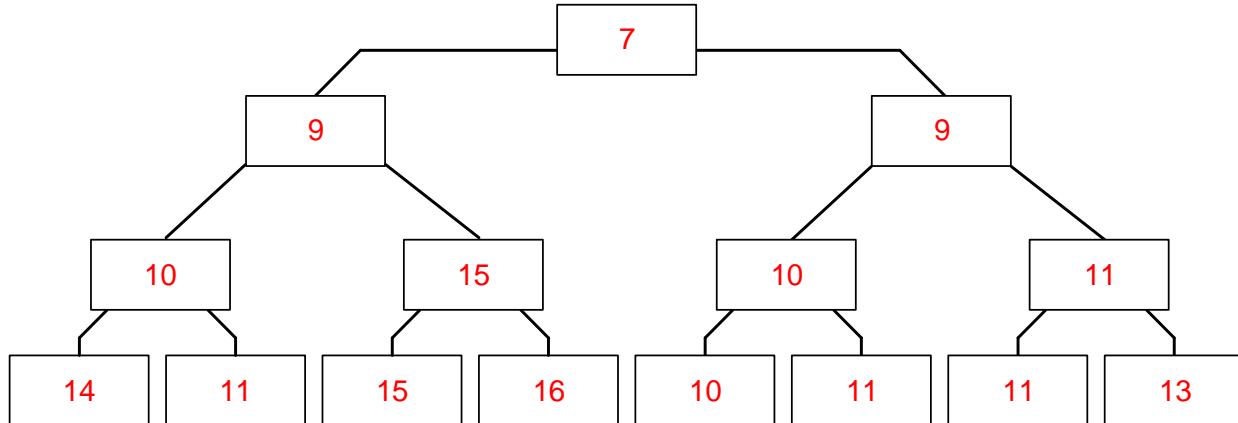
```
private void percolateDown( int hole ) {
    int child;
    AnyType tmp = array[ hole ];

    for( ; hole*2 <= currentSize; hole = child ) {
        child = hole*2;

        if( child != currentSize &&
            array[child+1].compareTo( array[child] ) <= 0 )
            child++;

        if( array[ child ].compareTo( tmp ) < 0 )
            array[ hole ] = array[ child ];
        else
            break;
    }
    array[ hole ] = tmp;
}
```

Votre réponse :



Question 2 : Recherche de patron**(14 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un entier via un calcul de hachage. Une implémentation de Rabin Karp vous est proposée à l'Annexe 2. On y considère l'alphabet $\Sigma = \{A, C, G, T\}$ où $d = |\Sigma| = 4$. On admettra l'encodage suivant :

Symbol	A	C	G	T
Code	0	1	2	3

Le hachage est calculé en base d modulo $q = 2^8 + 1 = 257$. Par exemple, la séquence « GATTACA », encodée **2, 0, 3, 3, 0, 1, 0** produira une valeur de hash :

$$\text{mod}(((((2 \cdot d + 0) \cdot d + 3) \cdot d + 3) \cdot d + 0) \cdot d + 1, q) = 161.$$

Sachant que pour deux entiers a et b pris dans l'intervalle $[0, 2^8]$,

$$\text{mod}(2^8 \cdot a + b, 257) = 257 + b - a \text{ si } a \geq b, \text{ et } b - a \text{ si } a < b.$$

Sachant que pour deux entiers a et b pris dans l'intervalle $[0, 2^{16}]$,

$$\text{mod}(2^{16} \cdot a + b, 257) = \text{mod}(a + b, 257)$$

2.1) (4 pts) Donnez le hash associé au patron « ATGATTACA ». Justifiez votre réponse par un calcul.

ATGATTACA peut être décomposé en trois parties A TGAT TACA

Le hash associé à A donne 0

Le hash associé à TGAC donne $\text{mod}((3 \times 4 + 2) \times 4 + 0) \times 4 + 3, 257 = 227$

Le hash associé à TACA donne $\text{mod}((3 \times 4 + 0) \times 4 + 1) \times 4 + 0, 257 = 196$

$$p = \text{mod}(0 \times 2^{16} + 227 \times 2^8 + 196, 257) = 257 + 196 - 227 = 226$$

Afin d'accélérer les calculs lors de la recherche du patron dans le texte, Rabin Karp exploite une formule récursive : $t_{s+1} = \text{mod}((t_s - hT[s])d + T[s+m+1], q)$, où $h = \text{mod}(d^{m-1}, q)$.

2.2) (2 pts) Donnez la valeur de la variable h pour le patron « ATGATTACA » :

Votre réponse :

$$\begin{aligned} h &= \text{mod}(4^{9-1}, 257) \\ &= \text{mod}(4^8, 257) \\ &= \text{mod}(2^{16}, 257) \\ &= 1 \end{aligned}$$

2.3) (2 pts) Simplifiez la formule $t_{s+1} = \text{mod}((t_s - hT[s])d + T[s+m+1], q)$ pour le cas où le patron recherché est « ATGATTACA » en utilisant le résultat de 2.2):

Votre réponse :

$$t_{s+1} = \text{mod}(4(t_s - T[s]) + T[s+m+1], 257)$$

2.4) (6 pts) Combien de faux positifs seront détectés si le patron « GAAAAAAAC » est recherché dans le texte « TAAAAAAAATGAAAAAAC ». Fiez-vous au code java donné à l'Annexe 2. Remarquez que dans l'implémentation donnée de la fonction updateHash, le calcul du modulo renvoie toujours une valeur positive. Aidez-vous de la table de la page suivante.

```
private int updateHash(int oldHash, char oldChar, char newChar) {
    if( !ENCODING.containsKey( oldChar ) ) throw new IllegalArgumentException();
    if( !ENCODING.containsKey( newChar ) ) throw new IllegalArgumentException();

    int newHash = oldHash;
    newHash -= h*ENCODING.get( oldChar );
    if( newHash < 0 ) newHash += q; // valeur positive
    newHash *= d;
    newHash += ENCODING.get( newChar );
    newHash %= q;
    return newHash;
}
```

Décalage	Sous-séquence	t_s	Faux Positif?
0	TAAAAAAA	3	✓
1	AAAAAAAAT	3	✓
2	AAAAAAATG	14	
3	AAAAAATGA	56	
4	AAAATGAA	224	
5	AAATGAAA	125	
6	AAATGAAAA	243	
7	AATGAAAAA	201	
8	ATGAAAAAA	33	
9	TGAAAAAAA	132	
12	GAAAAAAAC	3	

2.4.a) (3 pts) Nombre de faux positifs détectés :

2

2.4.b) (3 pts) Les sous-séquences associées aux faux positifs détectés,

TAAAAAAA et AAAAAAAAT

Question 3 : Programmation dynamique (16 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 3 \times 1 ; A_2 : 1 \times 4; A_3 : 4 \times 1; A_4 : 1 \times 1; A_5 : 1 \times 3$$

Considérez les tables **m** et **s** obtenues par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	12	7	8	17
2		0	4	5	8
3			0	4	15
4				0	3
5					0

s	1	2	3	4	5
1		1	1	1	1 ou 4
2			2	3	4
3				3	3
4					4
5					

Complétez cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

3.1) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

Parenthésage optimal: $A_1 ((A_2 A_3) A_4)$

Coût: 8

3.2) (4 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal: $((A_2 A_3) A_4) A_5$

Coût: 8

3.3) (1 pts) Donnez le coût minimal de la multiplication matricielle suivante. Justifiez par un calcul.

$$A_1(A_2A_3A_4A_5),$$

Coût : $3 \times 1 \times 3 + 8 = 17$

3.4) (1 pts) Donnez le coût minimal de la multiplication matricielle suivante. Justifiez par un calcul.

$$(A_1A_2)(A_3A_4A_5),$$

Coût : $3 \times 4 \times 3 + 12 + 15 = 63$

3.5) (1 pts) Donnez le coût minimal de la multiplication matricielle suivante. Justifiez par un calcul.

$$(A_1A_2A_3)(A_4A_5),$$

Coût : $3 \times 1 \times 3 + 7 + 3 = 19$

3.6) (1 pts) Donnez le coût minimal de la multiplication matricielle suivante. Justifiez par un calcul.

$$(A_1A_2A_3A_4)A_5,$$

Coût : $3 \times 1 \times 3 + 8 = 17$

3.7) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $A_1 (((A_2 A_3) A_4) A_5)$ ou $(A_1 ((A_2 A_3) A_4)) A_5$

Coût: 17

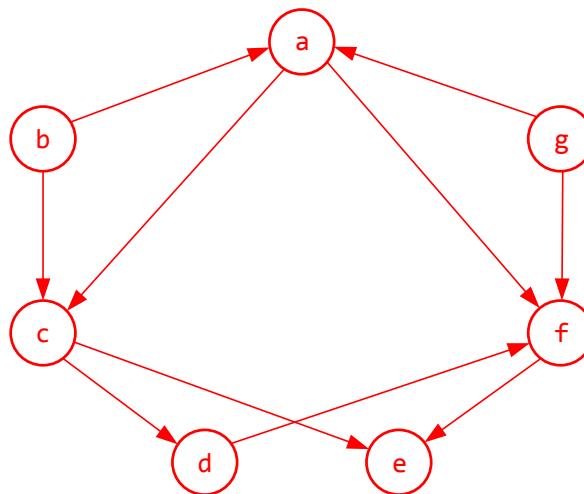
Question 4 : Algorithmes sur les graphes (24 points)

On veut connaître l'ordre topologique du graphe dirigé suivant :

$$V = \{a, b, c, d, e, f, g\}$$

$$E = \{(a, c), (a, f), (b, a), (b, c), (c, d), (c, e), (d, f), (f, e), (g, a), (g, f)\}$$

4.1) (2 pts) Reproduisez graphiquement le graphe $G = (V, E)$:



4.2) (7 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant une file vu en classe.

Nœud	1	2	3	4	5	6	7
a	2	1	0	-	-	-	-
b	0	-	-	-	-	-	-
c	2	1	1	0	-	-	-
d	1	1	1	1	0	-	-
e	2	2	2	2	1	1	0
f	3	3	2	1	1	0	-
g	0	-	-	-	-	-	-
Entrée	b, g	-	a	c	d	f	e
Sortie	b	g	a	c	d	f	e

Ordre trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	3	1	4	5	7	6	2

4.3) (7 pts) Donnez l'ordre topologique du graphe G en appliquant l'algorithme utilisant le parcours DFS post-ordre inverse. Partez du nœud a et visitez les nœuds alphabétiquement.

4.3.a) (2 pts) Donnez l'affichage DFS post-ordre obtenu :

e, f, d, c, a, b, g

4.3.b) (2 pts) Donnez l'affichage DFS post-ordre inverse obtenu :

g, b, a, c, d, f, e

4.3.c) (3 pts) Donnez l'ordre topologique trouvé (débutez la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	3	2	4	5	7	6	1

4.4) (2 pts) Combien de composantes fortement connexes G contient-il? Donnez-les.

Il en existe 7, soit chacun des sommets individuellement.

4.5 (3 pts) Combien de composantes fortement connexes G contiendrait-il si on y ajoutait l'arc (e, g)? Donnez-les

Il n'y en aurait plus que 2, b d'un côté et l'ensemble des sommets restants ensemble.

4.6) (3 pts) Proposez l'ajout de deux arcs dirigés à G pour que le graphe devienne connexe.

Réponse :

Arcs à ajouter :

Arc 1 : (e , g)

Arc 2 : (e , b)

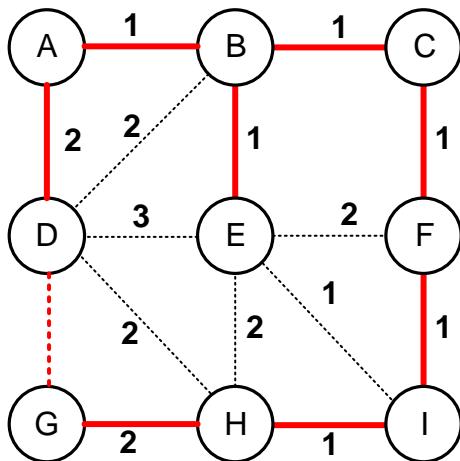
D'autres solutions existent.

Question 5 : Arbre sous-tendant minimum (12 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Prim (le nœud de départ étant A) et Kruskal en reliant les arêtes retenues dans les graphes données ci-après.

Respectez l'ordre alphabétique pour visiter les nœuds voisins ou les arêtes. Utilisez les tables fournies pour ce faire (le remplissage des tables ne compte pas dans l'attribution des points pour la question).

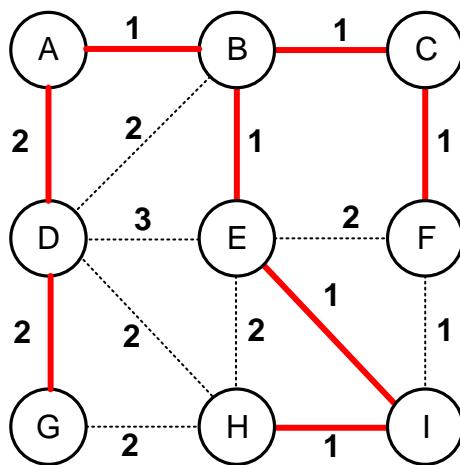
Par Prim (6 pts) :



Prim:

Nœud	Distance	Parent	Connu?
A	0	-	✓
B	1	A	✓
C	1	B	✓
D	2	A	✓
E	1	B	✓
F	1	C	✓
G	2	H	✓
H	1	I	✓
I	1	F	✓

Par Kruskal (6 pts) :



Kruskal :

Arête	Poids	Retenue?
(A, B)	1	✓
(A, D)	2	✓
(B, C)	1	✓
(B, D)	2	
(B, E)	1	✓
(C, F)	1	✓
(D, E)	3	
(D, G)	2	✓
(D, H)	2	
(E, F)	2	
(E, H)	2	
(E, I)	1	✓
(F, I)	1	
(G, H)	2	
(H, I)	1	✓

Question 6 : Généralités (16 points)

Répondez aux assertions suivantes par « vrai » ou par « faux ». Justifier votre réponse brièvement. Les réponses non justifiées ne seront pas considérées.

6.1) (2 points) Le tri monceau est un tri interne.

Vrai, puisqu'il n'utilise pas de mémoire supplémentaire.

6.2) (2 points) Soit $G = (V, E)$ un graphe dirigé. G^T (le graphe transposé de G) possède au moins deux composantes fortement connexes si et seulement si G est connexe.

Faux. Si G est connexe, alors G^T aussi, et il ne possède qu'une seule CFC.

6.3) (2 points) Les algorithmes issus des méthodes de programmation dynamique requièrent un espace mémoire proportionnel à $O(n^2)$ (les besoins en mémoire croissent quadratiquement avec la taille du problème).

Faux. Cela varie d'un problème à l'autre. Exemple Fibonacci peut-être résolu avec un espace mémoire (n), et même $O(1)$.

6.4) (2 points) Un algorithme glouton est un algorithme dont les besoins en mémoire sont proportionnels à $O(n^2)$ (croissent quadratiquement avec la taille du problème).

Faux. $O(n)$ pour Kruskal par exemple.

6.5) **(2 points)** La solution vue en classe au problème de parenthésage des matrices est un algorithme glouton.

Faux, algorithme issu des méthodes de programmation dynamique.

6.6) **(2 points)** Partant d'un sommet quelconque d'un graphe dirigé, le parcours DFS permet de statuer sur la connexité du graphe (dire si le graphe est fortement connexe ou pas).

Vrai. Si le graphe est fortement connexe, DFS visitera tous les nœuds en partant d'un sommet quelconque. Si ce n'est pas le cas lorsque le graphe n'est pas connexe.

6.7) **(2 points)** Si la longueur m d'un patron $P[1 :m]$ est le quart de la longueur n du texte $T[1 :n]$ dans lequel il est recherché ($m=n/4$), alors pour des valeurs de n suffisamment grandes, l'utilisation d'un automate est plus rapide que l'algorithme naïf, même en incluant le temps de construction de l'automate.

Faux. La construction du monceau se fait en $O(d^m)$. Si m est proportionnel à n , l'algorithme a une complexité exponentielle.

6.8) **(2 points)** La structure de données d'ensembles disjoints vue en classe permet d'identifier l'ensemble auquel appartient un élément en $O(\lg(n))$ dans le pire cas.

Faux. $O(n)$ en pire cas.



POLYTECHNIQUE
MONTRÉAL

Corrigé examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 – Structures de données et algorithmes		Tous	20161
Professeur		Local	Téléphone
Ettore Merlo, responsable Tarek Ould Bachir, chargé de cours		M-4105	5193
Jour	Date	Durée	Heures
Mardi	26 avril 2016	2 h 30	13 h 30 – 16 h 00
Documentation		Calculatrice	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
Directives particulières			
<input type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.			
Important	Cet examen contient 6 questions sur un total de 10 pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux		
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

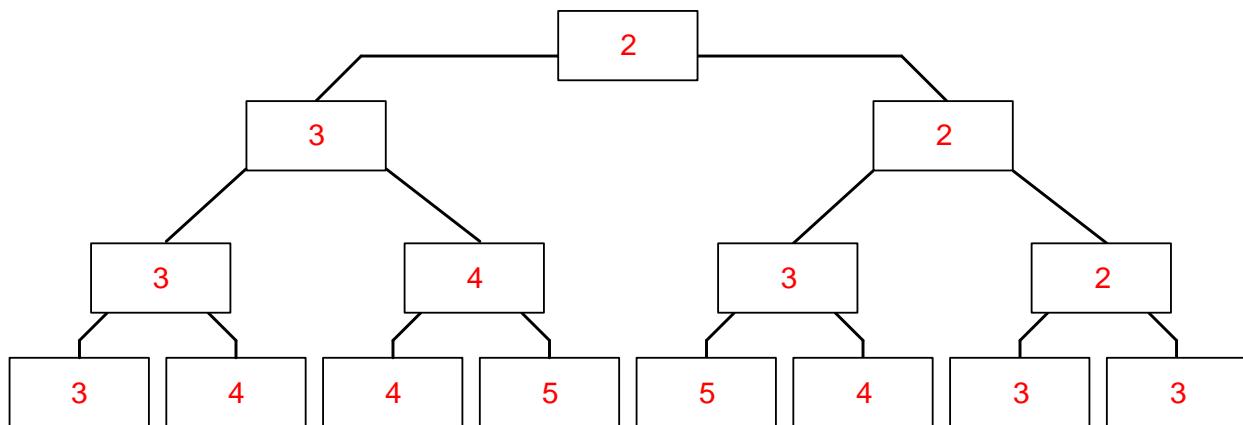
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux (15 points)

a) (3 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après.

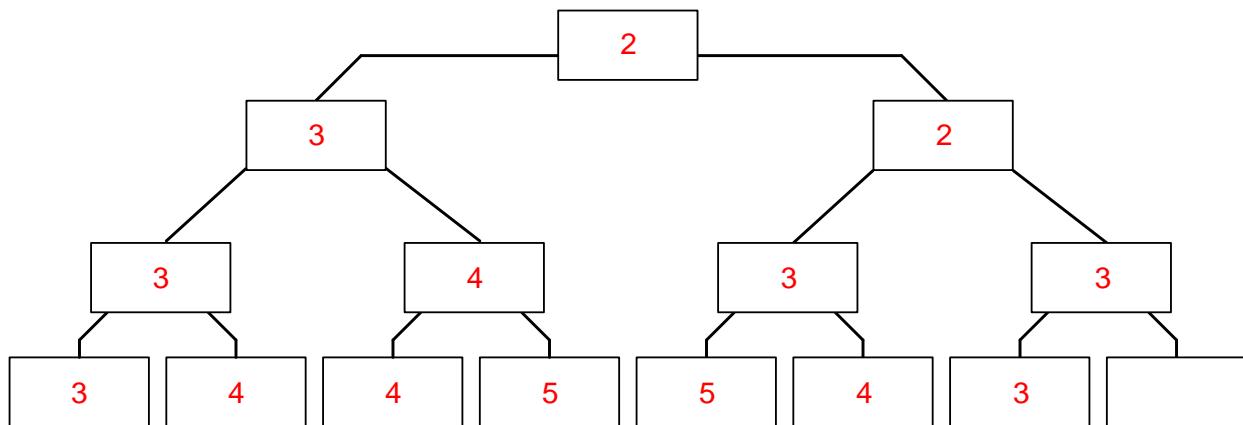
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	2	3	2	3	4	3	2	3	4	4	5	5	4	3	3

Votre réponse :



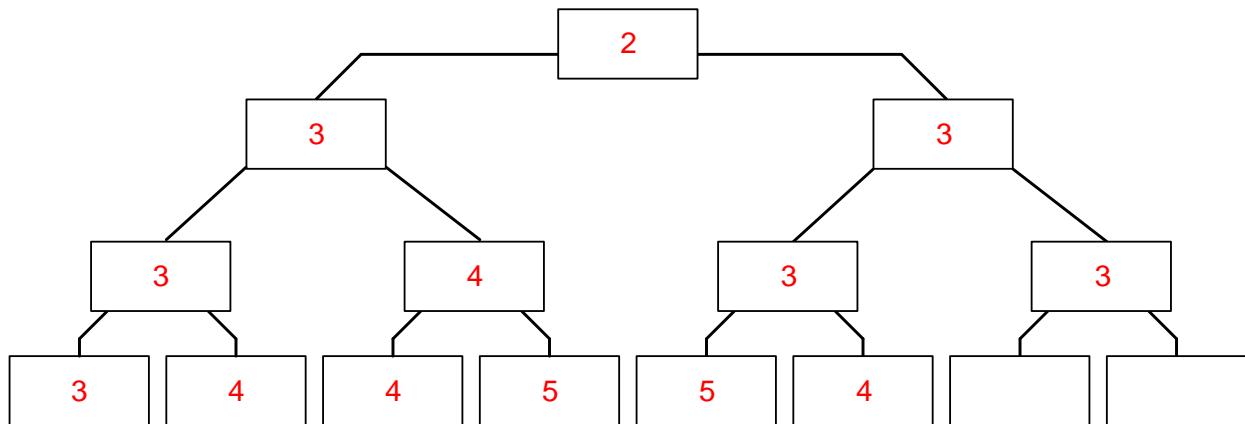
b) (4 pts) En partant du monceau de 1.a), effectuez un `deleteRoot()`.

Votre réponse :



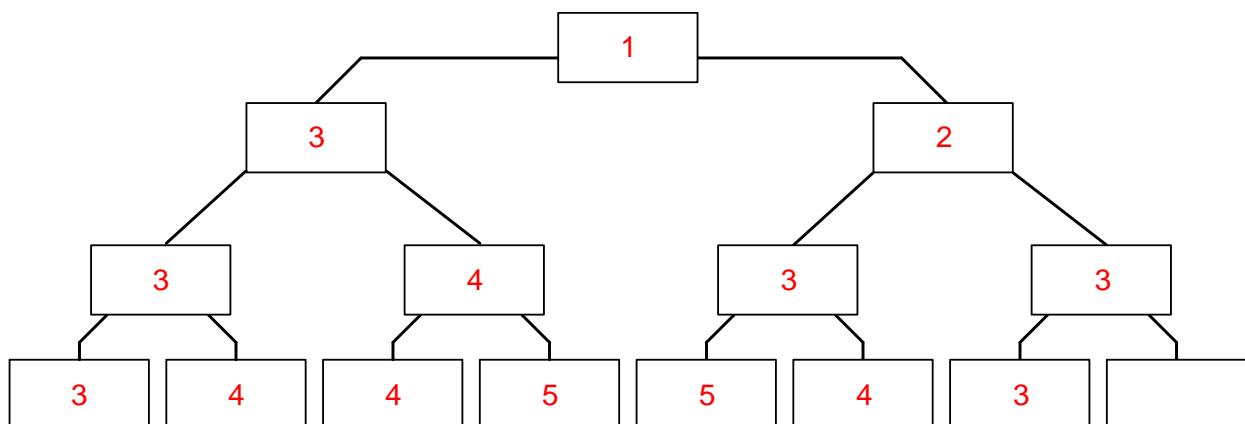
c) (4 pts) En partant du monceau de 1.b), effectuez un `deleteRoot()`.

Votre réponse :



d) (4 pts) En partant du monceau de 1.c), effectuez un `insert(1)`.

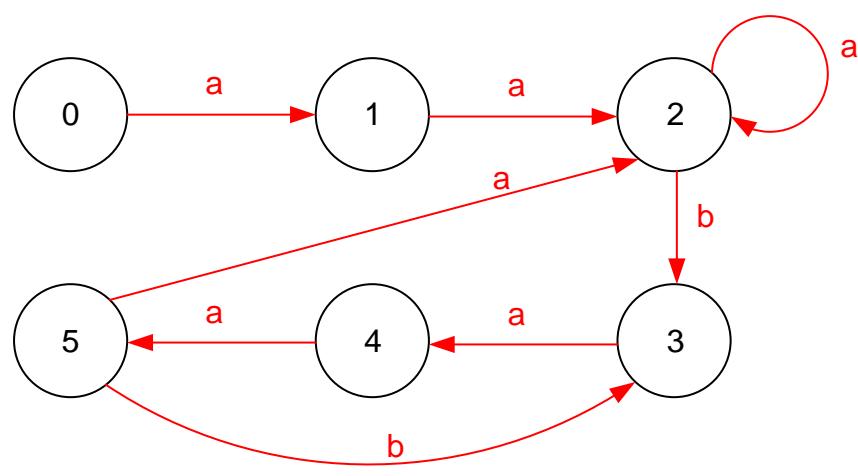
Votre réponse :



Question 2 : Recherche de patron**(15 points)**

On vous demande de retrouver le patron $P[1:5] = \text{«aabaa»}$ dans un texte.

a) (3 pts) Dessinez le diagramme d'états de l'automate à états finis permettant de ce faire :



Arcs absents mènent vers 0

b) (2 pts) Donnez la table de transitions de l'automate recherché.

$q \backslash a$	a	b	Autre
0	1	0	0
1	2	0	0
2	2	3	0
3	4	0	0
4	5	0	0
5	2	3	0

- c) (10 pts) Quels seront respectivement le ou les états les plus visités et le ou les états les moins visités par l'automate une fois arrivé à la fin du texte suivante. Combien de décalages seront retournés?

$T[1 : 30] = \text{« aabaaabbaabbbaabaabaaabaaaabaaa »}$

Aidez-vous de la table suivante:

T[i]	a	a	b	a	a	a	b	b
q	1	2	3	4	5	2	3	0
T[i]	a	a	b	b	b	a	a	b
q	1	2	3	0	0	1	2	3
T[i]	a	a	b	a	a	a	b	a
Q	4	5	3	4	5	2	3	4
T[i]	a	a	b	a	a	a		
q	5	2	3	4	5	2		

Le ou les états les plus visités : 2, 3 (4 pts)

Le ou les états les moins visités : 0, 1 (4 pts)

Nombre de décalages retournés : 5 (2 pts)

Question 3 : Programmation dynamique (16 points)

En utilisant l'algorithme vu en classe, donnez la longueur de la PLSC des séquences de caractères X = "babbaabbbaabb" et Y = "abaabbaaabbb". Aidez-vous du tableau donné ci-après. Inscrivez votre réponse au bas de cette page.

X\Y		a	b	a	a	b	b	a	a	a	b	b	b
	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0												
a	0												
b	0												
b	0												
a	0												
a	0												
b	0												
b	0												
b	0												
a	0												
a	0												
a	0												
b	0												
b	0												

Longueur de la PLSC : **11** (8 pts)

PLCS trouvée : **abaabbaaabbb** (8 pts)

Question 4 : Programmation dynamique (15 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 3 \times 1 ; A_2 : 1 \times 1; A_3 : 1 \times 3; A_4 : 3 \times 1; A_5 : 1 \times 3$$

Considérez les tables **m** et **s** obtenues par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	3	12	7	16
2		0	3	4	7
3			0	3	6
4				0	9
5					0

s	1	2	3	4	5
1		1	1	1	1 ou 4
2			2	2	4
3				3	4
4					4
5					

Complétez cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

- a) (4 pts) Donnez le parenthésage optimal pour multiplier A_3 à A_5 . Donnez son coût.

Parenthésage optimal: $(A_3 A_4) A_5$

Coût: 6

- b) (5 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal: $(A_2 (A_3 A_4)) A_5$

Coût: 7

- c) (6 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $A_1 ((A_2 (A_3 A_4)) A_5) \text{ ou } (A_1 (A_2 (A_3 A_4))) A_5$

Coût: 16

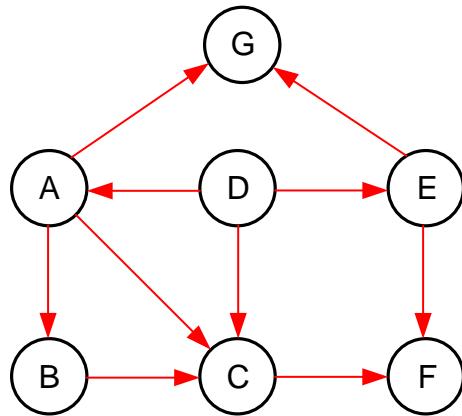
Question 5 : Ordre topologique (19 points)

On veut connaître l'ordre topologique du graphe suivant :

$$V = \{A, B, C, D, E, F, G\}$$

$$E = \{(A, B), (A, C), (A, G), (B, C), (C, F), (D, A), (D, C), (D, E), (E, F), (E, G)\}$$

- a) (3 pts) Reproduisez graphiquement le graphe :



- b) (8 pts) Donnez l'ordre topologique du graphe en appliquant l'algorithme utilisant une file vu en classe.

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Ordre trouvé (débutez la numérotation à 1) :

Nœud	A	B	C	D	E	F	G
Ordre :	2	4	6	1	3	7	5

c) (8 pts) Donnez l'ordre topologique du graphe en appliquant l'algorithme utilisant le parcours DFS post-ordre inverse. Partez du nœud A et visitez les nœuds alphabétiquement.

c.1 (4 pts) Donnez l'affichage post-ordre obtenu :

F, C, B, G, A, E, D

c.2 (4 pts) Donnez l'ordre topologique trouvé (débuter la numérotation à 1) :

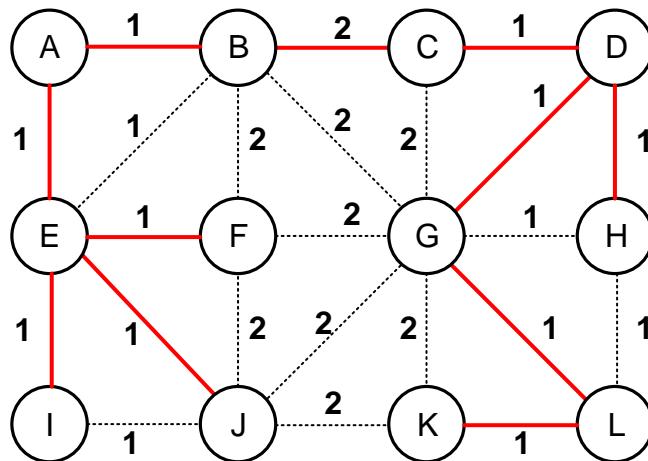
Nœud	A	B	C	D	E	F	G
Ordre :	3	5	6	1	2	7	4

Question 6 : Arbre sous-tendant minimum (20 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Prim (le nœud de départ étant A) et Kruskal en reliant les arêtes retenues dans les graphes données ci-après.

Respectez l'ordre alphabétique pour visiter les nœuds voisins ou les arêtes. Utilisez les tables fournies pour ce faire (le remplissage des tables ne compte pas dans l'attribution des points pour la question).

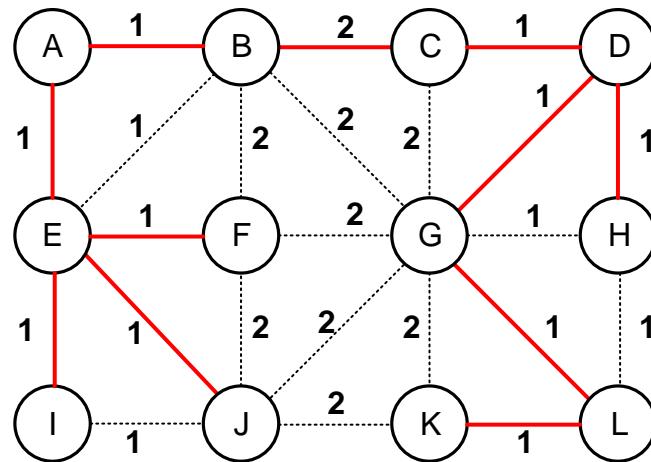
Par Prim (10 pts) :



Prim:

Nœud	Distance	Parent	Connu?
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			

Par Kruskal (10 pts) :



Kruskal :

Arête	Poids	Retenue?
(A,B)	1	
(A,E)	1	
(B, C)	2	
(B, E)	1	
(B, F)	2	
(B, G)	2	
(C, D)	1	
(C, G)	2	
(D, G)	1	
(D, H)	1	
(E, F)	1	
(E, I)	1	
(E, J)	1	
(F, G)	2	
(F, J)	2	
(G, H)	1	
(G, J)	2	
(G, K)	2	
(G, L)	1	
(H, L)	1	
(I, J)	1	
(J, K)	2	
(K, L)	1	



POLYTECHNIQUE
MONTRÉAL

Corrigé
examen final

INF2010

Sigle du cours

Q1	
Q2	
Q3	
Q4	
Q5	
Total	

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20151
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo, responsable / Tarek Ould Bachir, chargé		-	-
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Mercredi	22 avril 2014	2h30	9h30-12h00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
<i>Directives particulières</i>			
<input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.			
Important	Cet examen contient 5 questions sur un total de 29 pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux		
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Files de priorité (30 points)

Pour cette question, vous devez vous référer au code Java de l'Annexe 1.

On y trouve deux classes implémentant une file de priorité décrite par l'interface :

```
public interface PriorityQueue<AnyType> {
    int size();
    void clear();
    boolean isEmpty();
    boolean contains(AnyType x);
    AnyType peek() throws NoSuchElementException;
    AnyType remove() throws NoSuchElementException;
    boolean add(AnyType x, double priority);
    void updatePriority(AnyType x, double priority);
    AnyType getMax() throws NoSuchElementException;
}
```

La première classe s'appelle `HeapPriorityQueue`. Elle utilise un monceau tel que vu en classe. Un tableau contient tous les enregistrements. L'élément le plus prioritaire (plus petite valeur de priorité) se trouve au début.

La seconde classe s'appelle `ListPriorityQueue`. Elle fonctionne un peu comme la file idéale vue en classe. Un tableau contient tous les enregistrements. L'élément le plus prioritaire (plus petite valeur de priorité) se trouve à la fin.

Dans les deux cas, l'entrée 0 du tableau n'est pas utilisée. De plus une table de hachage est utilisée pour connaître la position de tous les éléments. Cette table de hachage est utilisée pour modifier la priorité d'un élément.

La modification de la priorité d'un élément se fait en ramenant l'élément modifié à la classe la plus prioritaire (début ou fin). On procède ensuite à son retrait puis à sa réinsertion avec la nouvelle priorité.

- a) (8 pts) Pour chacune des fonctions définies dans l'interface ci-haut et énumérées ci-après, indiquez la complexité asymptotique (en cas moyen) de la méthode en fonction de n , le nombre d'éléments présents. On supposera une distribution uniforme des priorités.

	HeapPriorityQueue	ListPriorityQueue
<code>remove()</code>	$O(\log(n))$	$O(1)$
<code>getMax()</code>	$O(n)$	$O(1)$
<code>add(...)</code>	$O(1)$	$O(n)$
<code>updatePriority(...)</code>	$O(\log(n))$	$O(n)$

Le programme suivant est exécuté :

```

1  public static void main(String[] args) {
2
3      HeapPriorityQueue<String> hpq = new HeapPriorityQueue<String>();
4      ListPriorityQueue<String> lpq = new ListPriorityQueue<String>();
5      int[] priorities = {4, 5, 6, 1, 2, 5, 3, 4, 2, 6, 3, 4, 6, 2, 5};
6
7      for(int i=0; i<priorities.length; i++){
8          String item = new String("v_" + (i+1));
9          System.out.println("insert "+item+" avec priorité "+priorities[i]);
10         hpq.add(item, priorities[i]);
11         lpq.add(item, priorities[i]);
12     }
13
14
15     System.out.println("modifie priorité de v_13 avec priorité 1");
16     hpq.updatePriority("v_13", 1);
17     lpq.updatePriority("v_13", 1);
18
19     System.out.println("\nFile de priorité de type monceau");
20
21     while(!hpq.isEmpty())
22         System.out.print(hpq.remove() + ", ");
23
24     System.out.println();
25
26     System.out.println("\nFile de priorité de type liste chaînée");
27
28     while(!lpq.isEmpty())
29         System.out.print(lpq.remove() + ", ");
30
31     System.out.println();
32 }
```

Les lignes 7 à 17 produisent l'affichage suivant :

```

insert v_1 avec priorité 4
insert v_2 avec priorité 5
insert v_3 avec priorité 6
insert v_4 avec priorité 1
insert v_5 avec priorité 2
insert v_6 avec priorité 5
insert v_7 avec priorité 3
insert v_8 avec priorité 4
insert v_9 avec priorité 2
insert v_10 avec priorité 6
insert v_11 avec priorité 3
insert v_12 avec priorité 4
insert v_13 avec priorité 6
insert v_14 avec priorité 2
insert v_15 avec priorité 5
modifie priorité de v_13 avec priorité 1
```

- b) (11 pts) Sachant que `remove()` retourne l'élément le plus prioritaire de la file après l'avoir retiré, donnez ci-après l'affichage résultant de l'exécution des lignes 19 à 22.

File de priorité de type monceau
`v_4, v_13, v_5, v_9, v_14, v_11, v_7, v_8, v_1, v_12, v_2, v_6, v_15, v_3, v_10,`

- c) (11 pts) Sachant que `remove()` retourne l'élément le plus prioritaire de la file après l'avoir retiré, donnez ci-après l'affichage résultant de l'exécution des lignes 26 à 29.

File de priorité de type liste chaînée
`v_13, v_4, v_14, v_9, v_5, v_11, v_7, v_12, v_8, v_1, v_15, v_6, v_2, v_10, v_3,`

Question 2 : Programmation dynamique (20 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 2 \times 3 ; A_2 : 3 \times 2; A_3 : 2 \times 1; A_4 : 1 \times 3; A_5 : 3 \times 2$$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	12	12	18	22
2		0	6	15	18
3			0	6	10
4				0	6
5					0

s	1	2	3	4	5
1		1	1	3	3
2			2	3	3
3				3	3
4					4
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

a) (5 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_3 . Donnez son coût.

Parenthésage optimal: $A_1 (A_2 A_3)$

Coût: 12

b) (5 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

Parenthésage optimal: $(A_1 (A_2 A_3)) A_4$

Coût: 18

c) (10 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $(A_1 (A_2 A_3))(A_4 A_5)$

Coût: 22

Question 3 : Ordre topologique**(10 points)**

NOTE IMPORTANTE : Les questions 3 à 5 font référence au code Java de l'Annexe 2. Pour les questions 4 et 5, ce code faire intervenir la file de priorité ListPriorityQueue discutée à la question 1 dont l'implémentation est donnée à l'Annexe 1.

La classe `Graph` de l'Annexe 2 permet d'implémenter un graph dirigé ou non dirigé. L'option est donnée à la construction du graphe via un booléen : `public Graph(boolean isDirected)`. Le constructeur par défaut met le booléen à vrai : `public Graph(){ this(true); }`

Le code qui suit crée un graphe dirigé et valué pour lequel on désire trouver un ordre topologique.

```

1  public static void main(String[] args) {
2      // On crée un graphe dirigé
3      Graph graph = new Graph();
4
5      graph.addVortex("a");
6      graph.addVortex("b");
7      graph.addVortex("c");
8      graph.addVortex("d");
9      graph.addVortex("e");
10     graph.addVortex("f");
11     graph.addVortex("g");
12
13     graph.addEdge("a", "b", 1.0);
14     graph.addEdge("a", "c", 1.0);
15     graph.addEdge("a", "d", 3.0);
16     graph.addEdge("a", "g", 5.0);
17     graph.addEdge("b", "d", 2.0);
18     graph.addEdge("b", "e", 1.0);
19     graph.addEdge("c", "b", 1.0);
20     graph.addEdge("c", "d", 1.0);
21     graph.addEdge("c", "e", 3.0);
22     graph.addEdge("c", "f", 3.0);
23     graph.addEdge("d", "g", 2.0);
24     graph.addEdge("e", "g", 3.0);
25     graph.addEdge("f", "d", 2.0);
26     graph.addEdge("f", "e", 1.0);
27
28     System.out.println( "Graph dirigé: " );
29     System.out.println( graph );
30
31     System.out.println( "Son ordre topologique: " );
32     System.out.println( graph.printTopologicalOrder() + "\n" );
33 }
```

Les lignes 28 et 29 produisent l'affichage suivant :

```
Graph dirigé:
a: b, 1.0; c, 1.0; d, 3.0; g, 5.0;
b: d, 2.0; e, 1.0;
c: b, 1.0; d, 1.0; e, 3.0; f, 3.0;
d: g, 2.0;
e: g, 3.0;
f: d, 2.0; e, 1.0;
g:
```

- a) (8 pts) Donnez le résultat de l'affichage résultant de l'exécution des lignes 31 et 32. Vous pouvez vous aider du tableau suivant (le remplissage du tableau n'est pas noté).

Nœud	1	2	3	4	5	6	7
a							
b							
c							
d							
e							
f							
g							
Entrée							
Sortie							

Affichage obtenu :

Son ordre topologique:

a, c, b, f, d, e, g,

- b) (2 pts) Donnez l'ordre trouvé (débuter la numérotation à 1) :

Nœud	a	b	c	d	e	f	g
Ordre :	1	3	2	5	6	4	7

Question 4 : Plus court chemin d'un graphe valué (20 points)

NOTE IMPORTANTE : Les questions 3 à 5 font référence au code Java de l'Annexe 2. Pour les questions 4 et 5, ce code faire intervenir la file de priorité ListPriorityQueue discutée à la question 1 dont l'implémentation est donnée à l'Annexe 1. Il est fortement suggéré d'avoir complété la question 1 pour cette question-ci.

La classe `Graph` de l'Annexe 2 permet d'implémenter un graph dirigé ou non dirigé. L'option est donnée à la construction du graphe via un booléen : `public Graph(boolean isDirected)`. Le constructeur par défaut met le booléen à vrai : `public Graph(){ this(true); }`

Le code qui suit crée un graphe dirigé et valué pour lequel on désire trouver tous les chemins partant du sommet « a ». Ce graphe est identique à celui de la question 3.

```

1  public static void main(String[] args) {
2      // On crée un graphe dirigé
3      Graph graph = new Graph();
4
5      graph.addVortex("a");
6      graph.addVortex("b");
7      graph.addVortex("c");
8      graph.addVortex("d");
9      graph.addVortex("e");
10     graph.addVortex("f");
11     graph.addVortex("g");
12
13     graph.addEdge("a", "b", 1.0);
14     graph.addEdge("a", "c", 1.0);
15     graph.addEdge("a", "d", 3.0);
16     graph.addEdge("a", "g", 5.0);
17     graph.addEdge("b", "d", 2.0);
18     graph.addEdge("b", "e", 1.0);
19     graph.addEdge("c", "b", 1.0);
20     graph.addEdge("c", "d", 1.0);
21     graph.addEdge("c", "e", 3.0);
22     graph.addEdge("c", "f", 3.0);
23     graph.addEdge("d", "g", 2.0);
24     graph.addEdge("e", "g", 3.0);
25     graph.addEdge("f", "d", 2.0);
26     graph.addEdge("f", "e", 1.0);
27
28     System.out.println( "Graph dirigé: " );
29     System.out.println( graph );
30
31     System.out.println( "Les chemins depuis \"a\": " );
32     System.out.println( graph.printPrintPathsFrom("a") + "\n" );
33 }
```

Les lignes 28 et 29 produisent l'affichage suivant :

```
Graph dirigé:
a: b, 1.0; c, 1.0; d, 3.0; g, 5.0;
b: d, 2.0; e, 1.0;
c: b, 1.0; d, 1.0; e, 3.0; f, 3.0;
d: g, 2.0;
e: g, 3.0;
f: d, 2.0; e, 1.0;
g:
```

- a) (10 pts) Donnez le résultat de l'affichage résultant de l'exécution des lignes 31 et 32. Vous pouvez vous aider du tableau suivant (le remplissage du tableau n'est pas noté).

Nœud	Connu	Dist min.	Parent
a		0,	
b		∞ ,	
c		∞ ,	
d		∞ ,	
e		∞ ,	
f		∞ ,	
g		∞ ,	

Affichage obtenu :

```
Les chemins depuis "a":
a : 0.0
a->b : 1.0
a->c : 1.0
a->c->d : 2.0
a->b->e : 2.0
a->c->f : 4.0
a->c->d->g : 4.0
```

b) (4 pts) Détaillez chacun des chemins les plus courts trouvés :

Destination	Le plus court chemin	Distance parcourue
b	a → b	1
c	a → c	1
d	a → c → d	2
e	a → b → e	2
f	a → c → f	4
g	a → c → d → g	4

c) (6 pts) Dans l'exécution de l'Algorithme de Dijkstra par la fonction `printPathsFrom(...)`, la file de prioritaire utilisée pour traiter les sommets est `ListPriorityQueue<Virtex>`. Aurait-il été préférable d'utiliser `HeapPriorityQueue<Virtex>`? Justifiez clairement mais brièvement votre réponse.

Note : On fait référence ici à la ligne :

```
// Execute Djikstra
PriorityQueue<Virtex> q = new ListPriorityQueue<Virtex>();
```

Trois méthodes sont principalement appelées dans ce code. `remove()`, `add(...)` et `updatePriority(...)` dont voici les complexités :

	HeapPriorityQueue	ListPriorityQueue
<code>remove()</code>	$O(\log(n))$	$O(1)$
<code>add(...)</code>	$O(1)$	$O(n)$
<code>updatePriority(...)</code>	$O(\log(n))$	$O(n)$

`remove()` est appelée $O(|V|)$ fois, tandis que `add(...)` et `updatePriority(...)` sont appelées $O(|E|)$ fois. Le graphe étant peu dense, $O(|E|)$ est proportionnel à $O(|V|)$. Il apparaît donc clairement que `HeapPriorityQueue` est un meilleur choix puisqu'on obtiendrait un complexité $O(|V|\log(|V|))$ au lieu de $O(|V|^2)$.

Question 5 : Arbre sous-tendant minimum**(20 points)**

NOTE IMPORTANTE : Les questions 3 à 5 font référence au code Java de l'Annexe 2. Pour les questions 4 et 5, ce code faire intervenir la file de priorité ListPriorityQueue discutée à la question 1 dont l'implémentation est donnée à l'Annexe 1. Il est fortement suggéré d'avoir complété la question 1 pour cette question-ci.

La classe Graph de l'Annexe 2 permet d'implémenter un graph dirigé ou non dirigé. L'option est donnée à la construction du graphe via un booléen : **public Graph(boolean isDirected)**. Le constructeur par défaut met le booléen à vrai : **public Graph(){ this(true); }**

Le code qui suit crée un graphe dirigé et valué pour lequel on désire trouver tous les chemins partant du sommet « a ». Ce graphe est identique à celui de la question 3.

```

1  public static void main(String[] args) {
2      // On crée un graphe non dirigé
3      Graph graph = new Graph(false);
4
5      graph.addVortex("A");
6      graph.addVortex("B");
7      graph.addVortex("C");
8      graph.addVortex("D");
9      graph.addVortex("E");
10
11     graph.addEdge("A", "B", 2.0);
12     graph.addEdge("A", "C", 1.0);
13     graph.addEdge("B", "E", 2.0);
14     graph.addEdge("C", "E", 3.0);
15     graph.addEdge("D", "B", 2.0);
16     graph.addEdge("D", "C", 1.0);
17
18     System.out.println( "Graph non dirigé: " );
19     System.out.println( graph );
20
21     System.out.println( graph.printPrimMinSpanningTree() + "\n" );
22
23     System.out.println( graph.printKruskalMinSpanningTree() + "\n" );
24 }
```

Les lignes 18 et 19 produisent l'affichage suivant :

```
Graph non dirigé:
A: B, 2.0; C, 1.0;
B: A, 2.0; E, 2.0; D, 2.0;
C: A, 1.0; E, 3.0; D, 1.0;
D: B, 2.0; C, 1.0;
E: B, 2.0; C, 3.0;
```

- a) (10 pts) Donnez le résultat de l'affichage résultant de l'exécution de la ligne 21. Vous pouvez vous aider du tableau suivant (le remplissage du tableau n'est pas noté).

Nœud	Connu	Dist min.	Parent
A		0,	
B		∞ ,	
C		∞ ,	
D		∞ ,	
E		∞ ,	

Affichage obtenu :

```
Coût = 6.0
A: B, 2.0; C, 1.0;
B: A, 2.0; E, 2.0;
C: A, 1.0; D, 1.0;
D: C, 1.0;
E: B, 2.0;
```

- b) (10 pts) Donnez le résultat de l'affichage résultant de l'exécution de la ligne 23. Vous pouvez vous aider du tableau suivant (le remplissage du tableau n'est pas noté).

Ordre des arêtes dans la file de priorité

Arête	Poids	Retenue?

Affichage obtenu :

Coût = 6.0
A: C, 1.0;
B: E, 2.0; D, 2.0;
C: A, 1.0; D, 1.0;
D: B, 2.0; C, 1.0;
E: B, 2.0;



Corrigé
examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 – Structures de données et algorithmes		Tous	20141
Professeur		Local	Téléphone
Ettore Merlo, responsable / Tarek Ould Bachir, chargé		M-5028	7128 / 5193
Jour		Date	Durée
Samedi		19 avril 2014	2h30
Documentation		Calculatrice	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.

Directives particulières	
<p><input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.</p>	

Important	Cet examen contient 6 questions sur un total de 15 pages (excluant cette page)
	La pondération de cet examen est de 40 %
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non

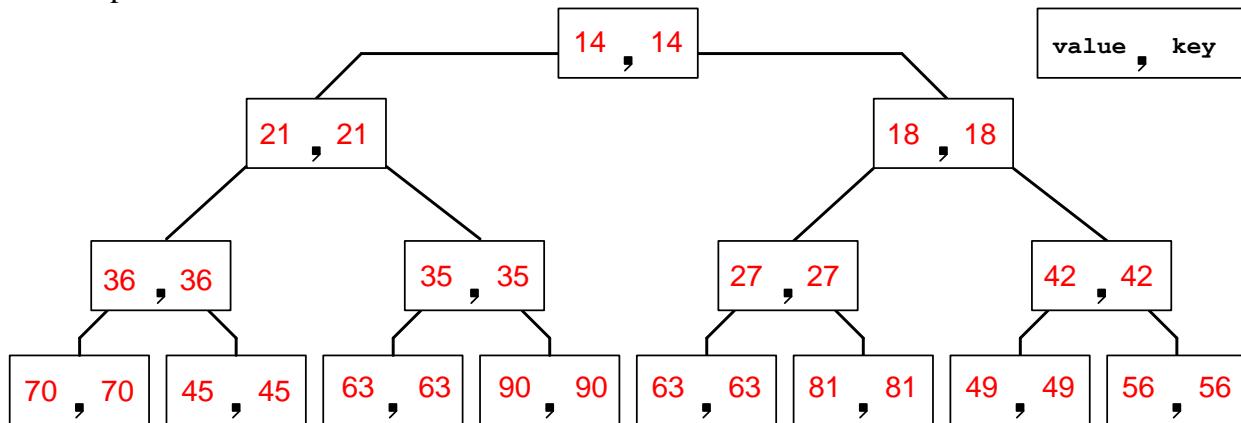
Question 1 : Monceaux (20 points)

Pour cette question, vous pouvez vous référer au code Java de l'Annexe 1.

- a) (2 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	14	21	18	36	35	27	42	70	45	63	90	63	81	49	56

Votre réponse :



- b) Considérez la fonction `changeKey(int location, int newKey)` donnée à l'Annexe 1 permettant de modifier la clé d'une entrée du monceau.

- b.1) (2 pts) Quelle est la complexité asymptotique de cette fonction en pire cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

En pire cas, cette fonction a une complexité $O(\lg(n))$ car la percolation vers le bas et l'insertion ont un pire cas en $O(\lg(n))$.

- b.2) (2 pts) Quelle est la complexité asymptotique de cette fonction en meilleur cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

En meilleur cas, cette fonction a une complexité $O(1)$ car la percolation vers le bas et l'insertion ont un meilleur cas en $O(1)$. Ce cas survient par exemple lorsqu'on change la clé de la dernière entrée pour une clé supérieure ou égale à sa clé courante.

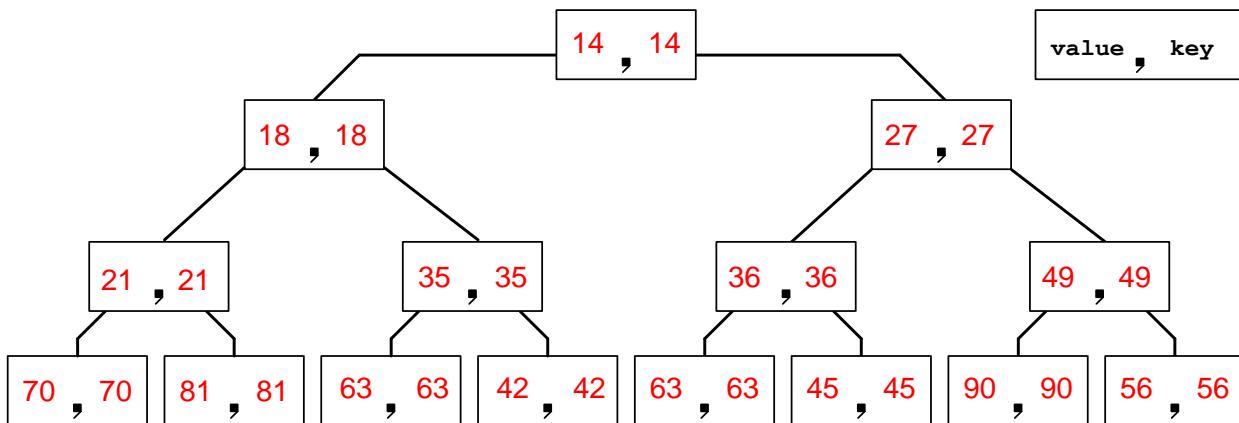
- c) (3 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel : **BinaryHeap(values_1, true)**

Indiquez dans les cases les valeurs de `value` et `key` de l'objet `Entry` contenues dans le monceau.

Le tableau **values** 1est le suivant:

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	36	18	45	81	63	27	90	70	21	35	42	63	14	49	56

Monceau résultant :



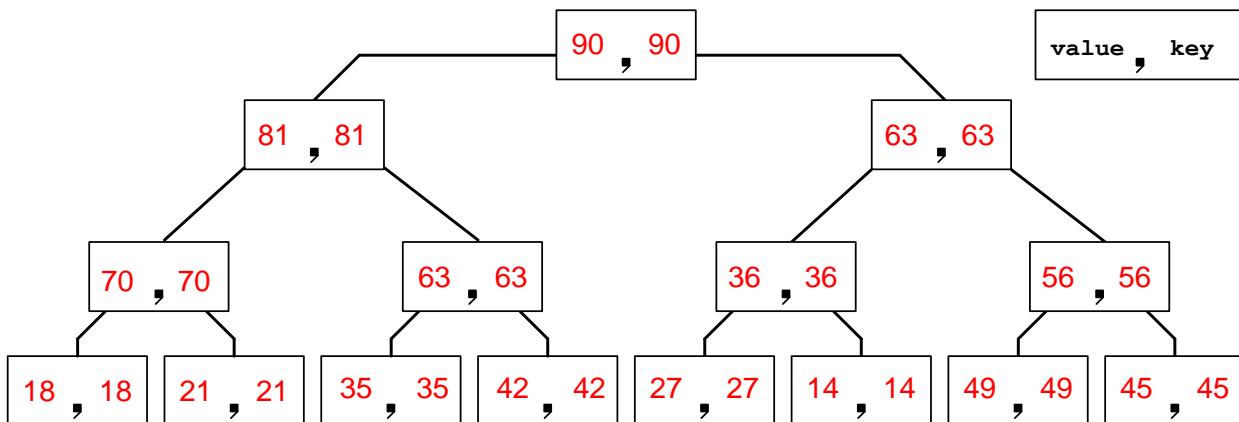
- d) (3 pts) En vous fiant au code donné à l'Annexe 1, dessinez le monceau résultant de l'appel : **BinaryHeap(values_1, false)**

Indiquez dans les cases les valeurs de `value` et `key` de l'objet `Entry` contenues dans le monceau.

Le tableau **values** 1 est le suivant:

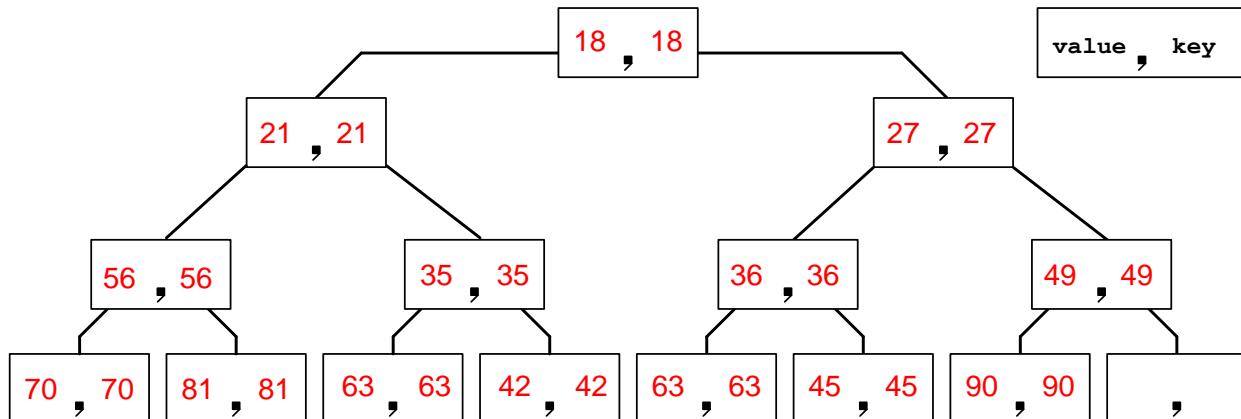
Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	36	18	45	81	63	27	90	70	21	35	42	63	14	49	56

Monceau résultant :

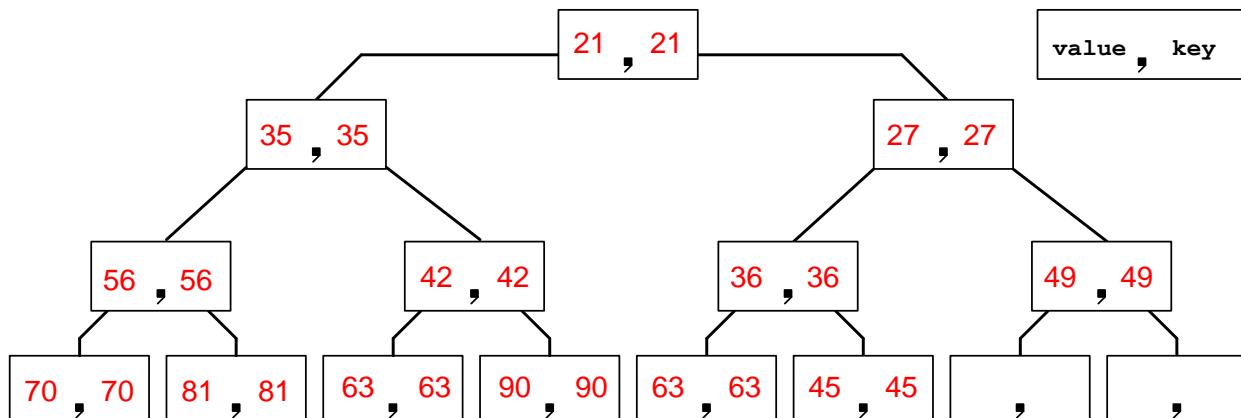


e) Dessinez l'état du monceau de la question 1.c) suite à deux appels consécutifs à `deleteRoot()` :

e.1) (1 pts) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

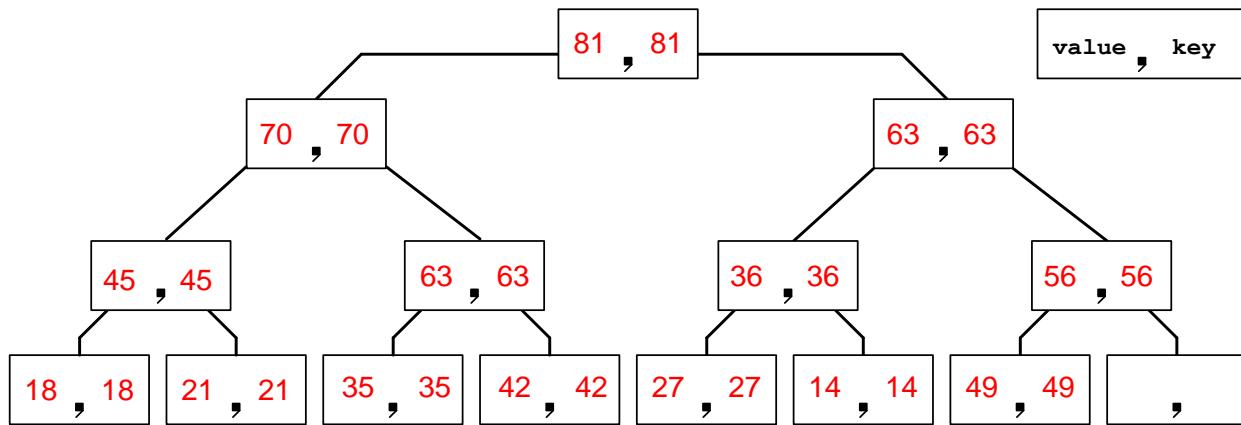


e.2) (1 pts) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

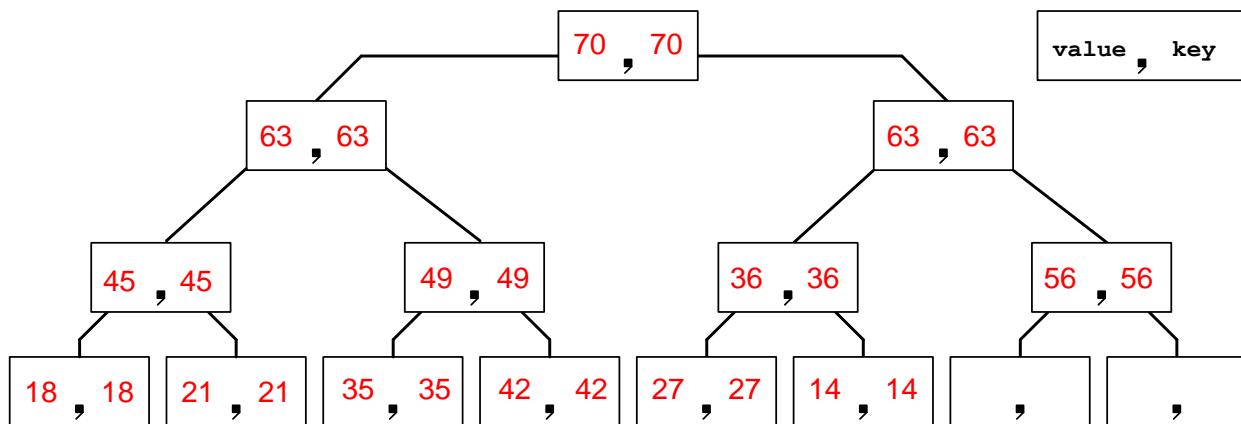


f) Dessinez l'état du monceau de la question 1.d) suite à deux appels consécutifs à `deleteRoot()` :

f.1) (1 pts) Monceau résultant du premier `deleteRoot()`. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.



f.2) (1 pts) Monceau résultant du second `deleteRoot()`. Indiquez dans les cases les valeurs de **value** et **key** de l'objet **Entry** contenues dans le monceau.

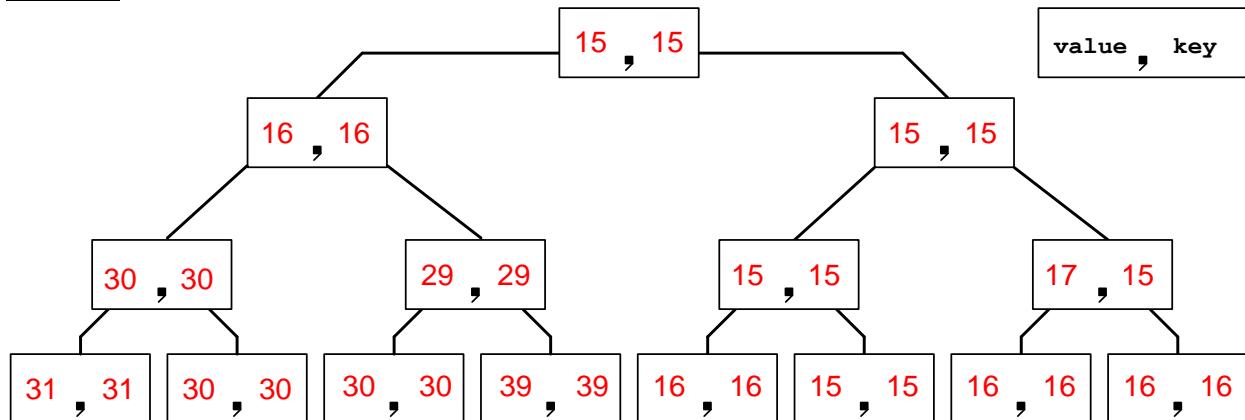


g) (4 pts) Exécutez `changeKey(15, 15)` sur le monceau suivant construit après un appel : `BinaryHeap(values_2)`. Indiquez dans les cases les valeurs de `value` et `key` de l'objet `Entry` contenues dans le monceau.

Le tableau `values_2` est le suivant:

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	15	16	15	30	29	15	16	31	30	30	39	16	15	16	17

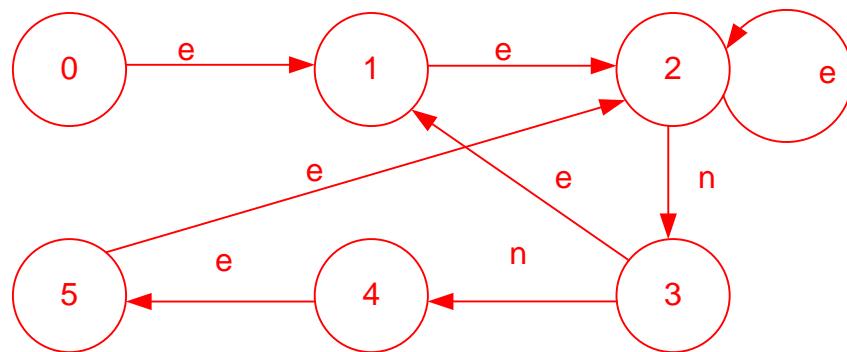
Résultat:



Question 2 : Recherche de patron**(14 points)**

On vous demande de retrouver le patron $P[1:8] = \text{« eenne »}$ dans un texte.

a) (4 pnts) Dessiner le diagramme d'états de l'automate à états finis permettant de ce faire :



b) (4 pnts) Donner la table de transitions de l'automate recherché.

$q \backslash a$	e	n	Autre
0	1	0	0
1	2	0	0
2	2	3	0
3	1	4	0
4	5	0	0
5	2	0	0

- c) (6 pts) Quels seront respectivement le ou les états les plus visités et le ou les états les moins visités par l'automate une fois arrivé à la fin du texte suivante.

$T[1 : 34] = \text{« eeeeeeeennnnnneeeeeennnnneeeeeennnneeee »}$

Aidez-vous de la table suivante:

$T[i]$	e	e	e	e	e	e	e
q	1	2	2	2	2	2	2
$T[i]$	e	n	n	n	e	n	e
q	2	3	4	0	1	0	1
$T[i]$	e	e	e	e	n	n	e
q	2	2	2	2	3	4	5
$T[i]$	e	e	e	n	n	e	e
q	2	2	2	3	4	5	2
$T[i]$	e	n	n	e	e	e	
q	2	3	4	5	2	2	

Le ou les états les plus visités : 2, visité 18 fois

Le ou les états les moins visités : 0. visité 2 fois, (Autre réponse acceptée, 0 1 et 5, visités 3 fois si on considère l'état initial de l'automate)

Question 3 : Programmation dynamique (15 points)

a) (10 points) En utilisant l'algorithme vu en classe, donnez la longueur de la PLSC des séquences de caractères X = "abcbcaacbabc" et Y = "bbacbabcbcabc".

Aidez-vous du tableau donné ci-après. Inscrivez votre réponse à la page suivante.

X\Y		a	b	c	b	c	a	a	c	b	a	b	c
	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	H, 0	D, 1	G, 1	D, 1					D, 1		D, 1	
b	0		D, 1		D, 2	G, 2				D		D	
a	0	D, 1	H, 1				D, 3	D			D		
c	0			D, 2		D, 3	H, 3		D			D	
b	0		D		D, 3	G, 3	H, 3			D		D	
a	0	D					D	D, 4			D		
b	0		D		D			H, 4		D		D	
c	0			D		D			D, 5				D
b	0		D		D					D, 6		D	
c	0			D		D			D	H, 6			D
a	0	D					D	D			D, 7		
b	0		D		D					D		D, 8	
c	0			D		D			D				D, 9
a	0	D					D	D			D		H, 9

Longueur de la PLSC : **bbaacbabc**

PLCS trouvée : **9**

b) (5 pts) Il existe une autre PLSC pour les séquences X et Y que celle trouvée à la question 3.a.
Donnez-la:

2^e PLCS : **bcbacbababc**

3^e PLCS : **acbacadabc**

Question 4 : Programmation dynamique (20 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 2 \times 1 ; A_2 : 1 \times 1; A_3 : 1 \times 3; A_4 : 3 \times 1; A_5 : 1 \times 6$$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	2	8	12	24
2		0	3	10	16
3			0	3	15
4				0	18
5					0

s	1	2	3	4	5
1		1	2	1	4
2			2	2	4
3				3	4
4					4
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

a) (3 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_3 . Donnez son coût.

Parenthésage optimal: $(A_1 \quad A_2) \quad A_3$

Coût: 8

b) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

Parenthésage optimal: $A_1 \quad (A_2 \quad (A_3 \quad A_4))$

Coût: 12

c) (5 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $(A_1 \quad (A_2 \quad (A_3 \quad A_4))) \quad A_5$

Coût: 24

d) (8 pts) Si A_1, A_2 et A_3 étaient des matrices 2×2 , 2×3 et 3×3 respectivement, quel serait le parenthésage optimal pour multiplier A_1 à A_4 ? Quel serait son coût? Aidez-vous des matrices suivantes pour répondre à la question.

m	1	2	3	4	5
1	0	12	30	19	
2		0	18	15	
3			0	9	
4				0	
5					0

s	1	2	3	4	5
1		1	2	1	
2			2	2	
3				3	
4					4
5					

Parenthésage optimal: $A_1 \text{ (} A_2 \text{ (} A_3 \text{ } A_4 \text{)) }$

Coût: **19**

Question 5 : Ordre topologique (13 points)

- a) (7 pts) Donnez l'ordre topologique du graphe suivant en appliquant l'algorithme utilisant une file vu en classe.

$$V = \{A, B, C, D, E, F, G\}$$

$$E = \{(A, D), (C, F), (D, F), (D, G), (E, B), (E, C)\}$$

Nœud	1	2	3	4	5	6	7
A	0	-	-	-	-	-	-
B	1	1	0	-	-	-	-
C	1	1	0	-	-	-	-
D	1	0	-	-	-	-	-
E	0	-	-	-	-	-	-
F	2	2	2	1	1	0	-
G	1	1	1	0	-	-	-
Entrée	A, E	D	B, C	G	-	F	-
Sortie	A	E	D	B	C	G	F

Ordre trouvé (débuter la numérotation à 1) :

Nœud	A	B	C	D	E	F	G
Ordre :	1	4	5	3	2	7	6

- b) (6 pts) Ce graphe admet au moins trois autres ordres topologiques. Donnez-les. D'autres tables vous sont fournies à l'Annexe 2. Utilisez-les si nécessaire.

Ordre alternatif #1 :

Nœud	A	B	C	D	E	F	G
Ordre:	1	5	4	3	2	6	7

Ordre alternatif #2 :

Nœud	A	B	C	D	E	F	G
Ordre:	2	3	4	5	1	6	7

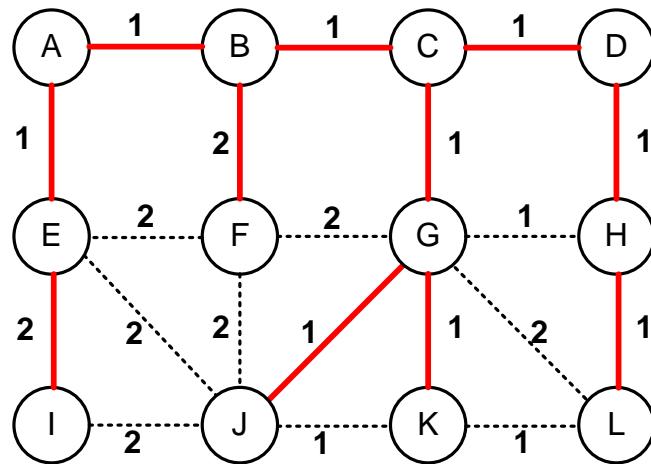
Ordre alternatif #3 :

Nœud	A	B	C	D	E	F	G
Ordre:	2	4	3	5	1	6	7

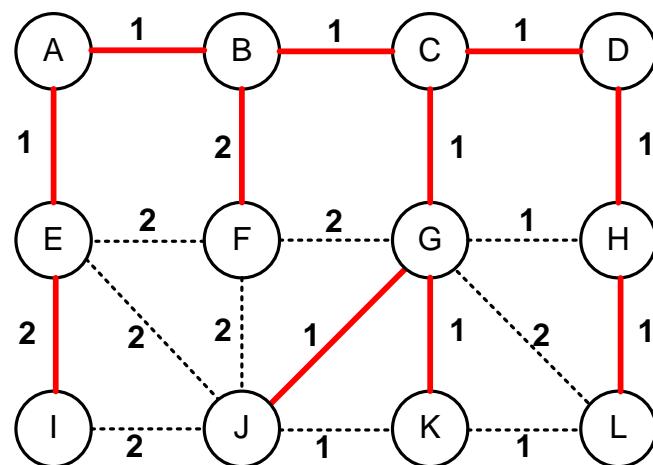
Question 6 : Arbre sous-tendant minimum (18 points)

Donnez les arbres sous-tendant minimum obtenus par les algorithmes de Boruvka, Prim (le noeud de départ étant A) et de Kruskal. Respectez l'ordre alphabétique pour effectuer vos traitements (pour visiter les composantes, les nœuds voisins ou les arêtes). Utilisez les tables fournies pour ce faire (le remplissage des tables compte dans l'attribution des points pour la question, sauf dans Boruvka où seule la réponse finale compte).

Par Boruvka:



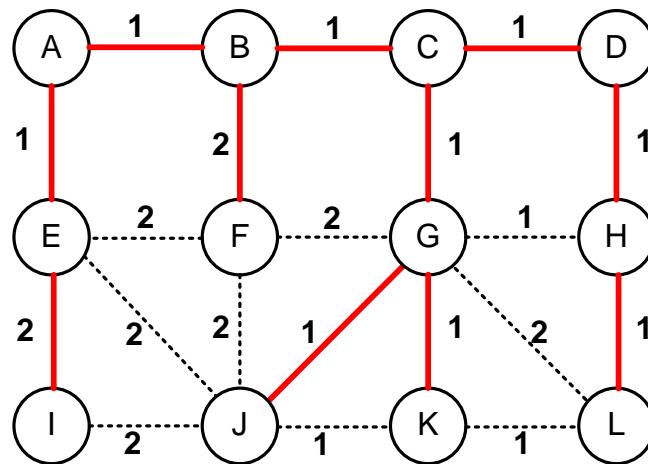
Par Prim:



Prim:

Nœud	Distance	Parent	Connu?
A	$\infty, 0$	-	✓
B	$\infty, 1$	A	✓
C	$\infty, 1$	B	✓
D	$\infty, 1$	C	✓
E	$\infty, 1$	A	✓
F	$\infty, 2$	B	✓
G	$\infty, 1$	C	✓
H	$\infty, 1$	D	✓
I	$\infty, 1$	E	✓
J	$\infty, 2$	G	✓
K	$\infty, 1$	G	✓
L	$\infty, 2, 1$	G, H	✓

Par Kruskal:



Kruskal:

Arête	Poids	Retenue?
(A,B)	1	✓
(A,E)	1	✓
(B, C)	1	✓
(B, F)	2	✓
(C, D)	1	✓
(C, G)	1	✓
(D, H)	1	✓
(E, F)	2	
(E, I)	2	✓
(E, J)	2	
(F, G)	2	
(F, J)	2	
(G, H)	1	
(G, J)	1	✓
(G, K)	1	✓
(G, L)	2	
(H, L)	1	✓
(I, J)	2	
(J, K)	1	
(K, L)	1	



POLYTECHNIQUE
MONTRÉAL

Corrigé
examen final

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 – Structures de données et algorithmes		Tous	20131
Professeur		Local	Téléphone
Ettore Merlo, responsable / Tarek Ould Bachir, chargé		M-5028	7128 / 5193
Jour	Date	Durée	Heures
Vendredi	26 avril 2012	2h30	9h30-12h00
Documentation	Calculatrice		
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
Directives particulières			
<p><input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.</p>			
Important	Cet examen contient 6 questions sur un total de 16 pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux			
Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non			

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

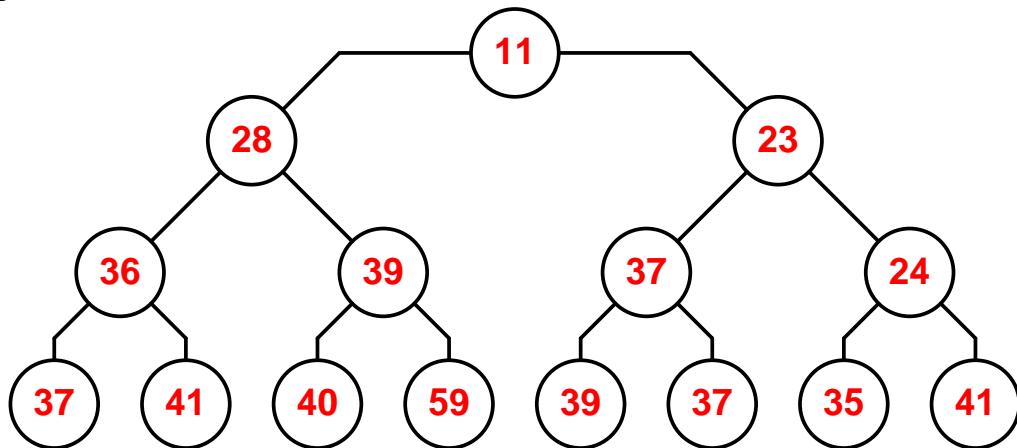
Question 1 : Monceaux (20 points)

Pour cette question, vous pouvez vous référer au code Java de l'annexe 1.

- a) (3 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	11	28	23	36	39	37	24	37	41	40	59	39	37	35	41

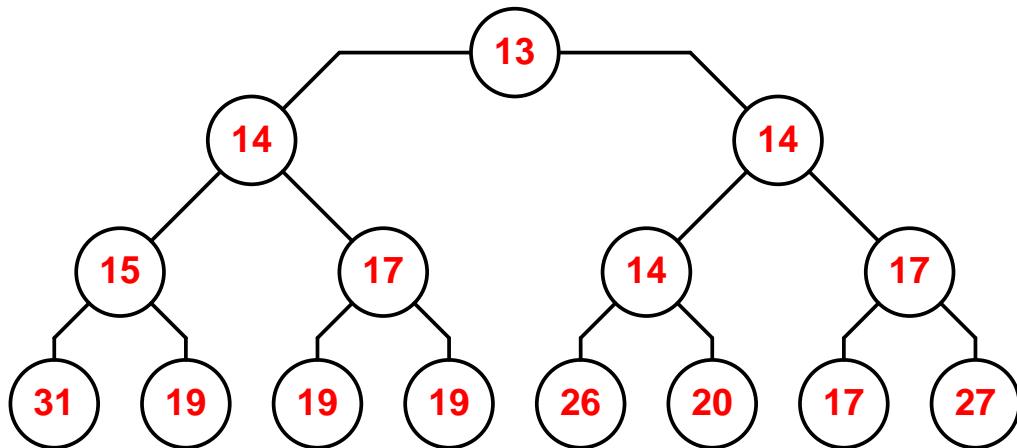
Votre réponse :



- b) (3 pts) Construisez, selon la technique vue en cours, un monceau MIN à partir du tableau de données suivant.

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu	14	15	17	19	19	20	27	31	14	17	19	26	14	17	13

Monceau résultant :

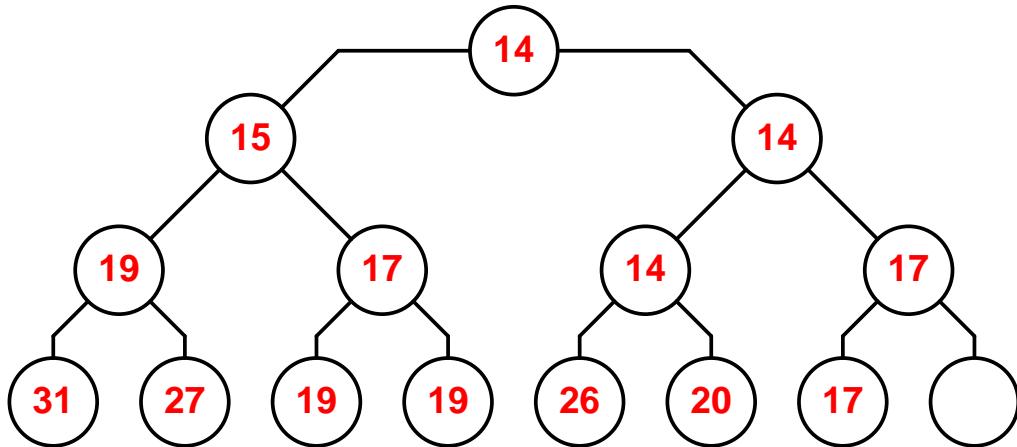


On se souviendra que l'on appelle ici le constructeur :

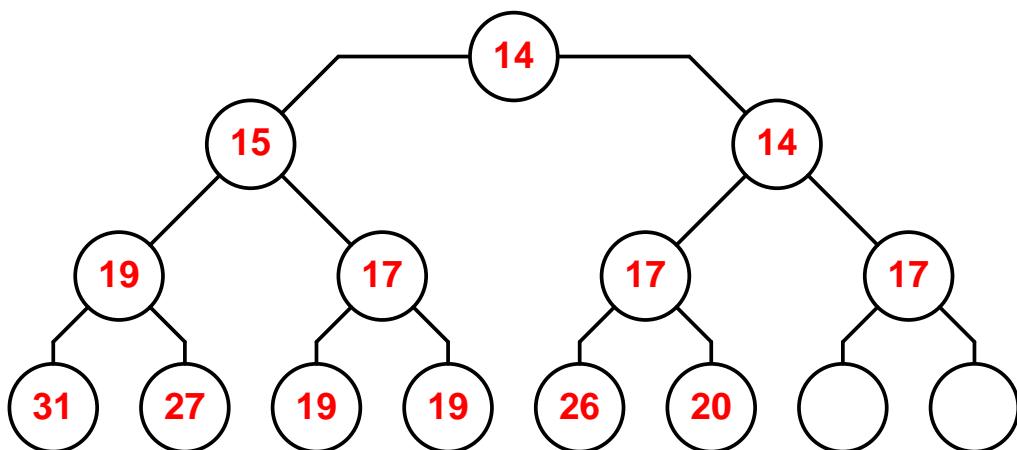
```
public BinaryHeap( AnyType [ ] items ).
```

c) Dessinez l'état du monceau de la question 1b) — monceau MIN que vous avez obtenu — suite à deux appels consécutifs à `deleteMin()` :

c.1) (4 pts) Monceau résultant du premier `deleteMin()`



c.2) (4 pts) Monceau résultant du second `deleteMin()`

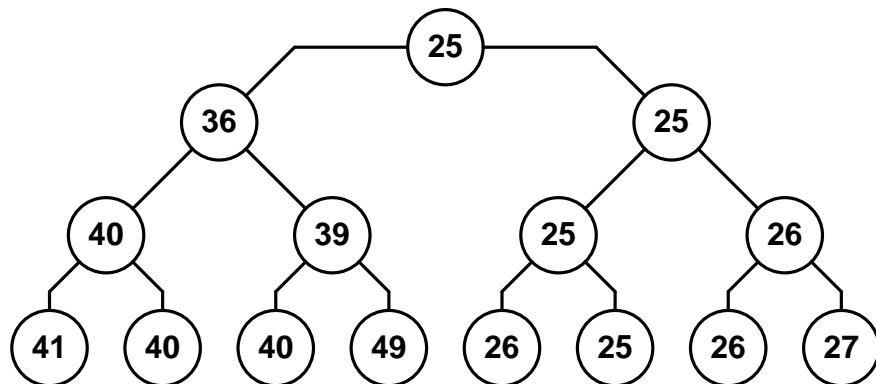


- d) Considérez la fonction `changeKey(int location, int newKey)` donnée à l'Annexe 1 permettant de modifier la clé d'une entrée du monceau.

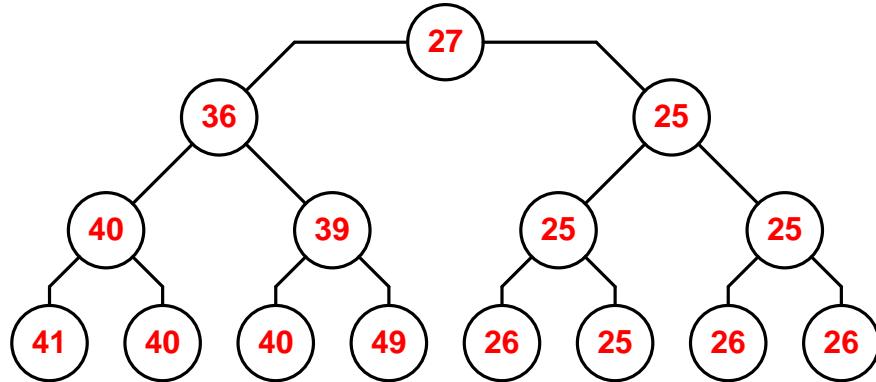
d.1) (3 pts) Quelle est la complexité asymptotique de cette fonction en pire cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

La fonction a une complexité $O(\lg(n))$ en pire cas car elle fait intervenir un percolateDown ($O(\log(n))$ en pire cas) suivi d'un insert ($O(\log(n))$ en pire cas).

d.2) (3 pts) Exécutez `changeKey(15, 14)` sur le monceau suivant:



Résultat:



Question 2 : Recherche de patron

(20 points)

On désire détecter dans un texte des patrons de type **nnnnnnnnnnnn** (où **n** est un chiffre entre '0' et '9' et **p** est le point '.') pour être en mesure de reconnaître des adresses IP telles que 132.207.201.172. On supposera qu'il faut impérativement que les séquences de chiffres comportent trois caractères. Autrement dit, on supposera que l'IP 132.207.72.1 est invalide.

- a) (7 pts) Donnez la fonction de transition de l'automate à utiliser en complétant le tableau suivant. N'utilisez que les cases nécessaires.

- b) (7 pts) Modifiez la fonction de transition de l'automate pour être en mesure de reconnaître aussi des adresses IP comportant des séquences de chiffres de moins de trois caractères, par exemple 132.207.72.1 ou 172.16.254.1. N'utilisez que les cases nécessaires.

État	n	p	Autre
0	1	0	0
1	2	4	0
2	3	4	0
3	3	4	0
4	5	0	0
5	6	8	0
6	7	8	0
7	3	8	0
8	9	0	0
9	10	12	0
10	11	12	0
11	3	12	0
12	13	0	0
13	14	12	0
14	15	12	0
15	3	12	0

- b) (6 pts) Complétez le tableau suivant où l'on considère le texte $T[1:25]$, et où l'on cherche à trouver l'état dans lequel se trouve votre automate après avoir reçu chacun des caractères de T .

Décalage s	Texte $T[1:25]$	État après la lecture	Décalages retenus
0	'M'	0	NON
1	'o'	0	NON
2	'n'	0	NON
3	''	0	NON
4	T'	0	NON
5	'P'	0	NON
6	''	0	NON
7	'e'	0	NON
8	's'	0	NON
9	't'	0	NON
10	::	0	NON
11	'1'	1	NON
12	'3'	2	NON
13	'2'	3	NON
14	''	4	NON
15	'2'	5	NON
16	'0'	6	NON
17	'7'	7	NON
18	''	8	NON
19	'7'	9	NON
20	'2'	10	NON
21	''	12	NON
22	'2'	13	NON
23	'0'	14	NON
24	'1'	15	OUI

Question 3 : Programmation dynamique (20 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 2 \times 1 ; A_2 : 1 \times 3; A_3 : 3 \times 1; A_4 : 1 \times 4; A_5 : 4 \times 2$$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	6	5	13	17
2		0	3	7	13
3			0	12	14
4				0	8
5					0

s	1	2	3	4	5
1		1	1	3	1
2			2	3	3
3				3	3
4					4
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_k, p_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

a) (3 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

Parenthésage optimal: $(A_1 (A_2 A_3)) A_4$

Coût: **13**

b) (3 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

Parenthésage optimal: $(A_2 A_3)(A_4 A_5)$

Coût: **13**

c) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

Parenthésage optimal: $A_1 ((A_2 A_3)(A_4 A_5))$

Coût: _____

d) (3 pts) Le parenthésage de la réponse 3c) est-il unique? Justifiez votre réponse. Donnez un parenthésage optimal alternatif s'il existe.

Votre réponse:

En calculant la solution pour $m[1,5]$, on trouve deux minimums à coûts équivalents. L'algorithme est supposé prendre le premier rencontré. Le second pose la séparation à 3.

Parenthésage optimal alternatif (s'il existe): $(A_1 (A_2 A_3)) (A_4 A_5)$

e) (4 pts) Si A_4 et A_5 étaient des matrices 1×2 et 2×2 respectivement, quel serait le parenthésage optimal pour multiplier A_1 à A_4 ? Quel serait son coût? Aidez-vous des matrices suivantes pour répondre à la question.

m	1	2	3	4	5
1	0	6	5	1	
2		0	3	5	
3			0	6	
4				0	
5					0

s	1	2	3	4	5
1		1	1	6	
2			2	3	
3				3	
4					4
5					

Parenthésage optimal: $A_1 ((A_2 A_3) A_4)$

Coût: **9**

f) (3 pts) Le parenthésage de la réponse 3e) est-il unique? Justifiez votre réponse. Donnez un parenthésage optimal alternatif si il existe.

Votre réponse:

En calculant la solution pour $m[1,4]$, on trouve deux minimums à coûts équivalents. L'algorithme est supposé prendre le premier rencontré. Le second pose la séparation à 3.

Parenthésage optimal alternatif (s'il existe): $(A_1 (A_2 A_3)) A_4$

Question 4 : Ordre topologique (14 points)

- a) (8 pts) Donnez l'ordre topologique du graphe suivant en appliquant l'algorithme utilisant une file vu en classe.

$$V = \{A, B, C, D, E, F, G\}$$

$$E = \{(A, B), (A, G), (B, C), (B, F), (B, G), (C, E), (D, C), (E, G), (F, E)\}$$

Nœud	1	2	3	4	5	6	7
A	0	-	-	-	-	-	-
B	1	0	0	-	-	-	-
C	2	2	1	0	-	-	-
D	0	0	-	-	-	-	-
E	2	2	2	2	1	0	-
F	1	1	1	0	0	-	-
G	3	2	2	1	1	1	0
Entrée	A, D	B	-	C, F	-	E	G
Sortie	A	D	B	C	F	E	G

Ordre trouvé (débuter la numérotation à 1) :

Nœud	A	B	C	D	E	F	G
Ordre :	1	3	4	2	6	5	7

- b) (6 pts) Ce graphe admet trois autres ordres topologiques. Donnez-les. D'autres tables vous sont fournies à l'Annexe 2. Utilisez-les si nécessaire.

Ordre alternatif #1 :

Nœud	A	B	C	D	E	F	G
Ordre:	1	3	5	2	6	4	7

Ordre alternatif #2 :

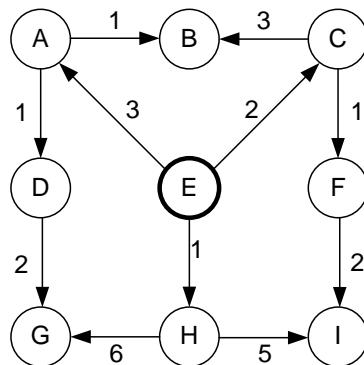
Nœud	A	B	C	D	E	F	G
Ordre:	2	3	4	1	6	5	7

Ordre alternatif #3 :

Nœud	A	B	C	D	E	F	G
Ordre:	2	3	5	1	6	4	7

Question 5 : Algorithm de Dijkstra**(14 points)**

Considérez le graphe suivant:



- a) (8 pts) Exécutez l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de E. Visitez les voisins d'un nœud par ordre alphabétique.

Nœud	Connu	Dist min.	Parent
A		∞ , 3	E
B		∞ , 5, 4	C, A
C		∞ , 2	E
D		∞ , 4	A
E		0,	-
F		∞ , 3	C
G		∞ , 7, 6	H, D
H		∞ , 1	E
I		∞ , 6, 5	H, F

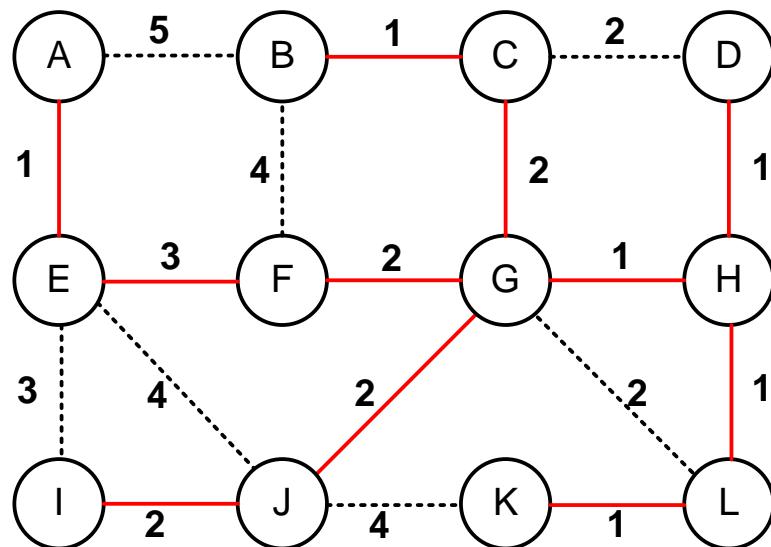
- b) (6 pts) Détaillez chacun des chemins les plus courts trouvés parmi ceux demandés :

Destination	Le plus court chemin	Distance parcourue
B	E → A → B	4
G	E → A → D → G	6
I	E → C → F → I	5

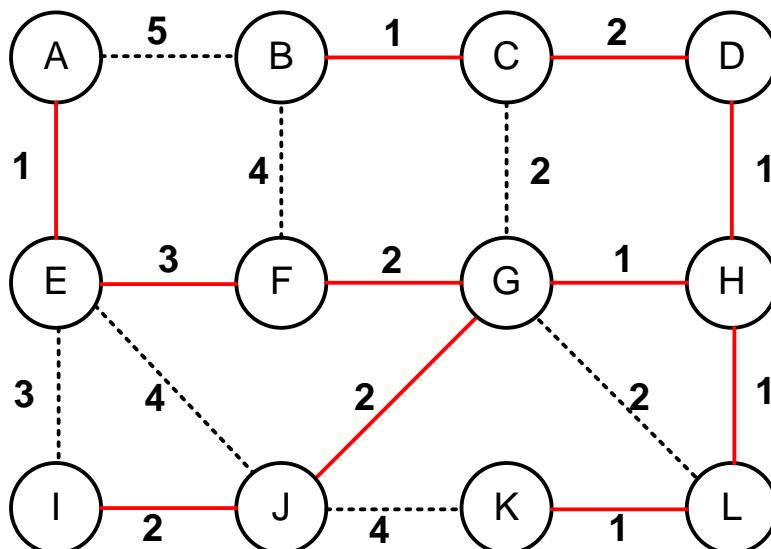
Question 6 : Arbre sous-tendant minimum (12 points)

Donnez les arbres sous-tendant minimum obtenus par l'algorithme de Prim (le noeud de départ étant A) et par l'algorithme de Kruskal. Traitez les nœuds voisins ou les arêtes par ordre alphabétique. Utilisez les tables de la page suivante pour ce faire (le remplissage des tables compte dans l'attribution des points pour la question).

Par Prim:



Par Kruskal:



Prim:

Nœud	Distance	Parent	Connu?
A	0	-	
B	$\infty, 5, 4, 1$	A, F, C	
C	$\infty, 2$	G	
D	$\infty, 1$	H	
E	$\infty, 1$	A	
F	$\infty, 3$	E	
G	$\infty, 2$	F	
H	$\infty, 1$	G	
I	$\infty, 3, 2$	E, J	
J	$\infty, 4, 2$	E, G	
K	$\infty, 1$	L	
L	$\infty, 2, 1$	G, H	

Kruskal:

Arête	Poids	Retenue?
(A,B)	5	✗
(A,E)	1	✓
(B, C)	1	✓
(B, F)	4	✗
(C, D)	2	✓
(C, G)	2	✗
(D, H)	1	✓
(E, F)	3	✓
(E, I)	3	✗
(E, J)	4	✗
(F, G)	2	✓
(G, H)	1	✓
(G, J)	2	✓
(G, L)	2	✗
(H, L)	1	✓
(I, J)	2	✓
(J, K)	4	✗
(K, L)	1	✓

Annexe 1

```

public class BinaryHeap<AnyType>
{
    public BinaryHeap( ) { this( DEFAULT_CAPACITY ); }

    @SuppressWarnings("unchecked")
    public BinaryHeap( int capacity )
    {
        currentSize = 0;
        array = new Entry[ capacity + 1 ];
    }

    @SuppressWarnings("unchecked")
    public BinaryHeap( AnyType [ ] items )
    {
        currentSize = items.length;
        array = new Entry[ ( currentSize + 2 ) * 11 / 10 ];

        int i = 1;
        for( AnyType item : items )
            array[ i++ ] = new Entry<AnyType>(item);

        buildHeap( );
    }

    public void insert( AnyType x )
    {
        if( currentSize == array.length - 1 )
            enlargeArray( array.length * 2 + 1 );

        // Percolate up
        int hole = ++currentSize;
        for( ; hole > 1 && x.hashCode() < array[ hole / 2 ].hashCode(); hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
        array[ hole ] = new Entry<AnyType>(x);
    }

    private void insert( Entry<AnyType> x )
    {
        if( currentSize == array.length - 1 )
            enlargeArray( array.length * 2 + 1 );

        // Percolate up
        int hole = ++currentSize;
        for( ; hole > 1 && x.hashCode() < array[ hole / 2 ].hashCode(); hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
        array[ hole ] = x;
    }

    @SuppressWarnings("unchecked")
    private void enlargeArray( int newSize )
    {
        Entry<AnyType>[] old = array;
        array = new Entry[ newSize ];
    }
}

```

```

    for( int i = 0; i < old.length; i++ )
        array[ i ] = old[ i ];
    }

public AnyType findMin( )
{
    if( isEmpty( ) ) return null;
    return array[ 1 ].value;
}

public AnyType deleteMin( )
{
    if( isEmpty( ) ) return null;

    AnyType minItem = findMin( );
    array[ 1 ] = array[ currentSize-- ];
    percolateDown( 1 );

    return minItem;
}

private void buildHeap( )
{
    for( int i = currentSize / 2; i > 0; i-- )
        percolateDown( i );
}

public boolean isEmpty( ) { return currentSize == 0; }

public void makeEmpty( ) { currentSize = 0; }

private void percolateDown( int hole )
{
    int child;
    Entry<AnyType> tmp = array[ hole ];

    for( ; hole * 2 <= currentSize; hole = child )
    {
        child = hole * 2;
        if( child != currentSize &&
            array[ child + 1 ].hashCode() < array[ child ].hashCode() )
            child++;
        if( array[ child ].hashCode() < tmp.hashCode() )
            array[ hole ] = array[ child ];
        else
            break;
    }
    array[ hole ] = tmp;
}

```

```
public void changeKey(int location, int newKey)
{
    if( location < 1 ) return;
    if( location > currentSize ) return;

    Entry<AnyType> tmp = array[location];
    tmp.setKey( newKey );

    array[location] = array[currentSize--];

    percolateDown( location );
    insert( tmp );
}

private static final int DEFAULT_CAPACITY = 10;

private int currentSize;      // Number of elements in heap
private Entry<AnyType> [] array; // The heap array

private static class Entry<AnyType>
{
    public int key;
    public AnyType value;

    public Entry(AnyType value) {this.key = value.hashCode(); this.value = value; }

    public Entry(int key, AnyType value) {this.key = key; this.value = value; }

    public void setKey(int key) { this.key = key; }

    public boolean equals(Object cmp)
    {
        return this.hashCode() == cmp.hashCode();
    }

    public int hashCode()
    {
        return key;
    }

    public String toString()
    {
        return "(v-" + value.toString() + ", k-" + key + ")";
    }
}
```

Annexe 2

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							

Nœud	1	2	3	4	5	6	7
A							
B							
C							
D							
E							
F							
G							
Entrée							
Sortie							



Corrigé examen final

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20121
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo – responsable / Tarek Ould Bachir - chargé		M-5028	7128 / 5193
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Mercredi	30 avril 2012	2h30	9h30-12h00

<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Aucune		
<input type="checkbox"/> Toute	<input type="checkbox"/> Toutes		Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.
<input checked="" type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable		

<i>Directives particulières</i>		
<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen. 		

Bonne chance à tous!

Important	Cet examen contient 6 questions sur un total de 15 pages (excluant cette page)
	La pondération de cet examen est de 40 %

Vous devez répondre sur : le questionnaire le cahier les deux

Vous devez remettre le questionnaire : oui non

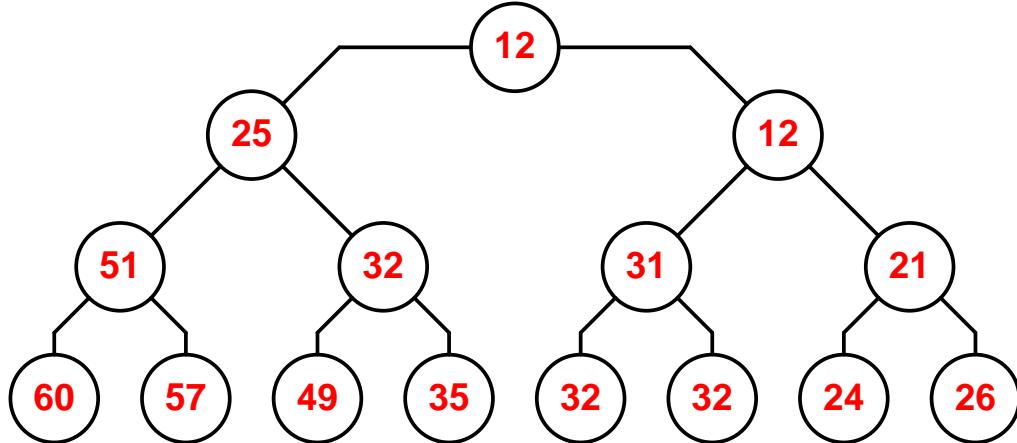
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(20 points)**

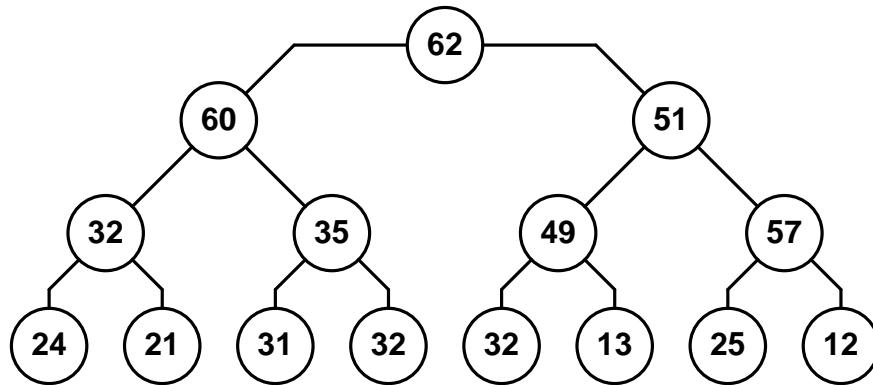
a) (5 pts) Dessinez le monceau contenu en mémoire dans le tableau ci-après :

Indice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Contenu	-	12	25	13	51	32	31	21	60	57	49	35	32	32	24	26

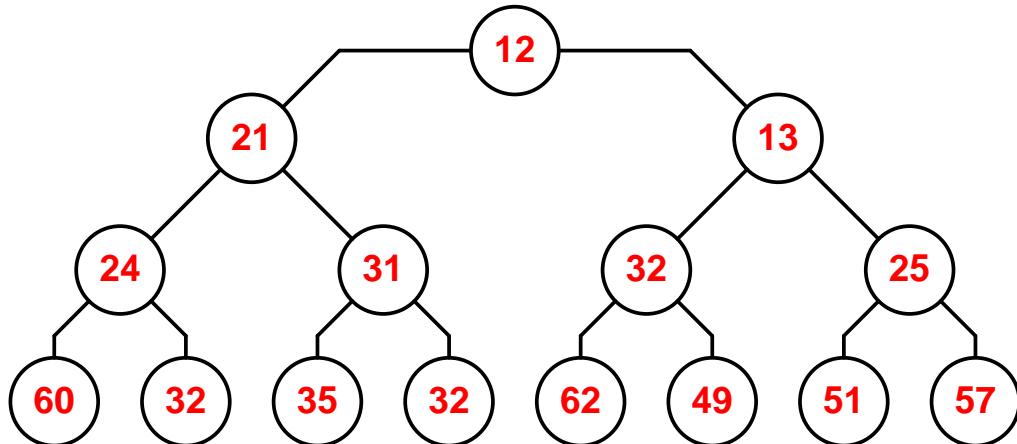
Votre réponse:



b) (7 pts) Construire, selon la technique vue en cours, un monceau MIN à partir de l'arbre binaire suivant :

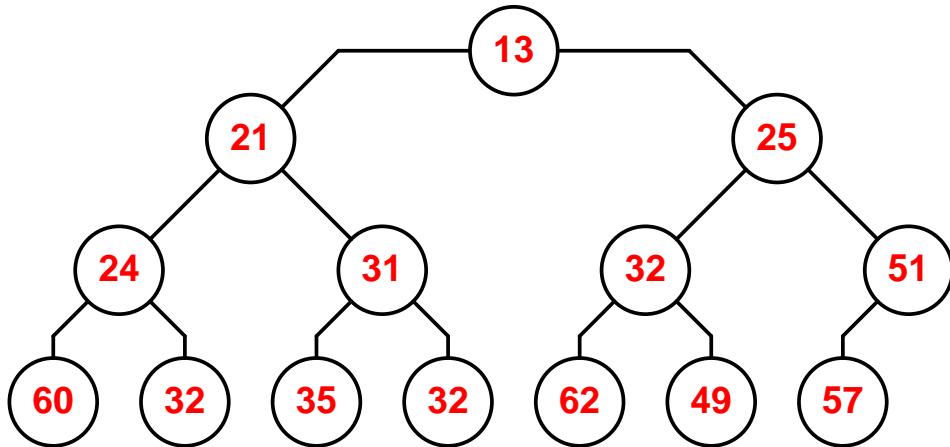


Monceau résultant :

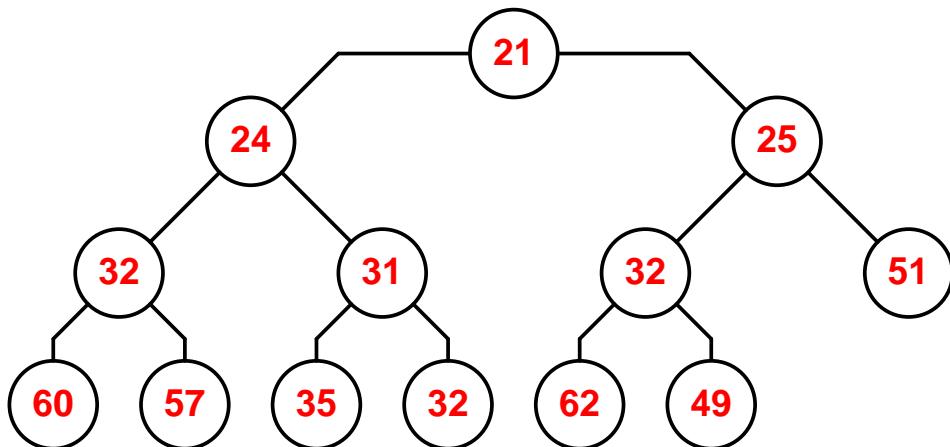


c) Dessiner l'état du monceau de la question 1.b) — monceau MIN que vous avez obtenu — suite à deux appels consécutifs à `deleteMin()` :

c.1) (1.5 pts) Monceau résultant du premier `deleteMin()`



c.2) (1.5 pts) Monceau résultant du second `deleteMin()`



d) (5 pts) Complétez la fonction `troisieme()`, qui retourne la troisième plus petite valeur d'un monceau MIN. Le code Java complet de l'implémentation du monceau MIN est donnée à l'Annexe 1 :

```

@SuppressWarnings("unchecked")
public AnyType troisieme() throws Exception
{
    if( currentSize < 3)
        throw new Exception("Le monceau a moins de 3 éléments");

    int maxIndex = Math.min(currentSize, 7); // COMPLÉTER ICI

    AnyType[] arr = (AnyType[]) new Comparable[3];

    arr[ 0 ] = array[ 1 ];

    arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
                array[ 2 ] : array[ 3 ];

    arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
                array[ 2 ] : array[ 3 ];

    for(int i=4; i <= maxIndex; i++)
    {
        if( array[ i ].compareTo( arr[ 1 ] ) < 0 )
        {
            % COMPLÉTER ICI
            arr[ 2 ] = arr[ 1 ];
            arr[ 1 ] = array[ i ];

        }
        else if( array[ i ].compareTo( arr[ 2 ] ) < 0 )
        {
            % COMPLÉTER ICI
            arr[ 2 ] = array[ i ];

        }
    }

    return arr[2];
}

```

Question 2 : Recherche de patron

(20 points)

On désire retrouver le patron P = "GAUGAUCHE" au moyen d'un automate.

a) (10 pts) Donnez la fonction de transition de l'automate à utiliser en complétant le tableau suivant. N'utiliser que les cases nécessaires.

b) (10 pts) Complétez le tableau suivant où l'on considère le texte $T[1:26]$, et où l'on cherche à trouver l'état dans lequel se trouve votre automate après avoir reçu chacun des caractères de T .

Décalage s	Texte T[1:26]	État après la lecture	Décalages retenus
0	'G'	1	
1	'A'	2	
2	'C'	0	
3	'H'	0	
4	T	0	
5	'S'	0	
6	'A '	0	
7	'G'	1	

(continué)

(suite)

Décalage s	Texte T[1:26]	État après la lecture	Décalages retenus
8	'A'	2	
9	'U'	3	
10	'G'	4	
11	'A'	5	
12	'U'	6	
13	'G'	4	
14	'A'	5	
15	'U'	6	
16	'G'	4	
17	'A'	5	
18	'U'	6	
19	'C'	7	
20	'H'	8	
21	'E'	9	✓
22	'G'	1	
23	'A'	2	
24	'G'	1	
25	'E'	0	

Question 3 : Plus Longue Sous-séquence Commune (PLSC) (20 points)

a) (10 pts) En vous aidant du tableau suivant, donnez la PLSC des deux mots "PETOCHE" et "EPTOGRAPHE"

		P	E	T	O	C	H	E
	0	0	0	0	0	0	0	0
E	0	0, H	1, D	1, G	1, G	1, G	1, G	1, D
P	0	1, D	1, H					
T	0	1, H	1, H	2, D	2, G	2, G	2, G	2, G
O	0	1, H	1, H	2, H	3, D	3, G	3, G	3, G
G	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
R	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
A	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
P	0	1, H	1, H	2, H	3, H	3, H	3, H	3, H
H	0	1, H	1, H	2, H	3, H	3, H	4, D	4, G
E	0	1, H	2, D	2, H	3, H	3, H	4, H	5, D

PLSC: **ETOHE**

Longueur de la PLSC: **5**

b) (5 pts) Si possible, proposez d'autres PLSC au couple de mots "PETOCHE" et "EPTOGRAPHE":

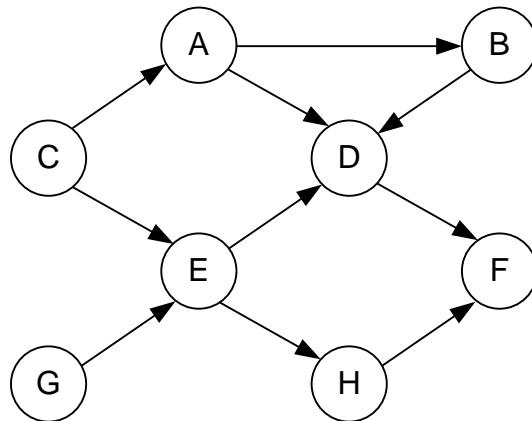
PTOHE

c) (5 pts) En utilisant les résultats tableau que vous avez obtenu en a), donnez l'ensemble des PLSC connues des mots "PETO" et "EPTO".

ETO, PTO

Question 4 : Ordre topologique**(10 points)**

a) (7 pts) Donnez l'ordre topologique du graphe suivant en appliquant l'algorithme utilisant une file vu en classe.



Noeud	1	2	3	4	5	6	7	8
A	1	0						
B	1	1	1	0				
C	0							
D	3	3	3	2	1	0		
E	2	1	0					
F	2	2	2	2	2	2	1	0
G	0							
H	1	1	1	1	0			
Entrée	C, G	A	E	B	H	D	-	F
Sortie	C	G	A	E	B	H	D	F

Ordre trouvé:

Nœud	A	B	C	D	E	F	G	H
Ordre:	3	5	1	7	4	8	2	6

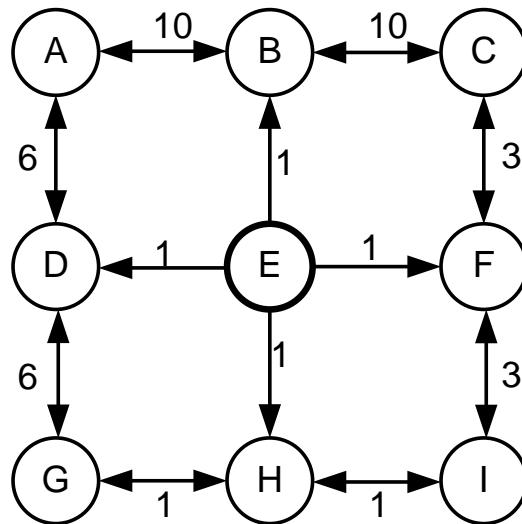
b) (3 pts) Proposez un ordre topologique alternatif à ce graphe:

Ordre alternatif:

Nœud	A	B	C	D	E	F	G	H
Ordre:	3	5	2	7	4	8	1	6

Question 5 : Algorithm de Dijkstra**(14 points)**

Considérez le graphe suivant. Les arcs à deux terminaisons indiquent des arcs double (de A à B et de B à A par exemple) de même poids.



- a) (9 pts) Exécutez l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de E. Visitez les voisins d'un nœud par ordre alphabétique.

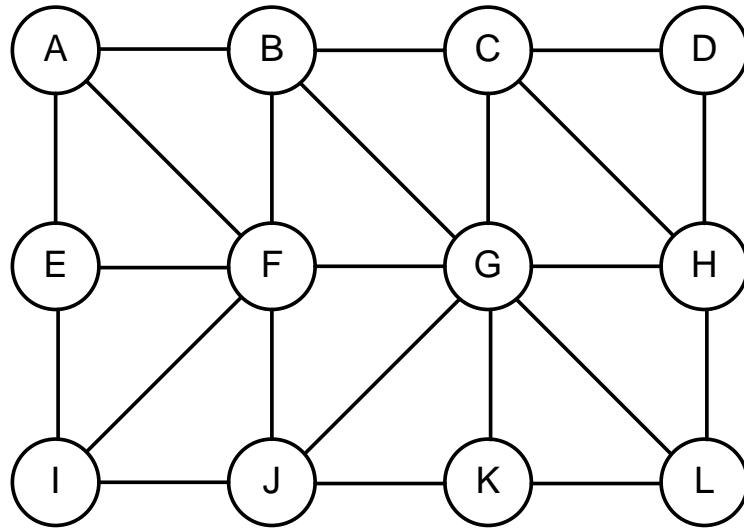
Nœud	Connu	Dist min.	Parent
A	✓	∞, 11, 7	B, D
B	✓	∞, 1	E
C	✓	∞, 11, 4	B, F
D	✓	∞, 1	E
E	✓	0,	-
F	✓	∞, 1	E
G	✓	∞, 7, 2	D, H
H	✓	∞, 1	E
I	✓	∞, 4, 2	F, H

b) (5 pts) Détaillez chacun des chemins les plus courts trouvés parmi ceux demandés :

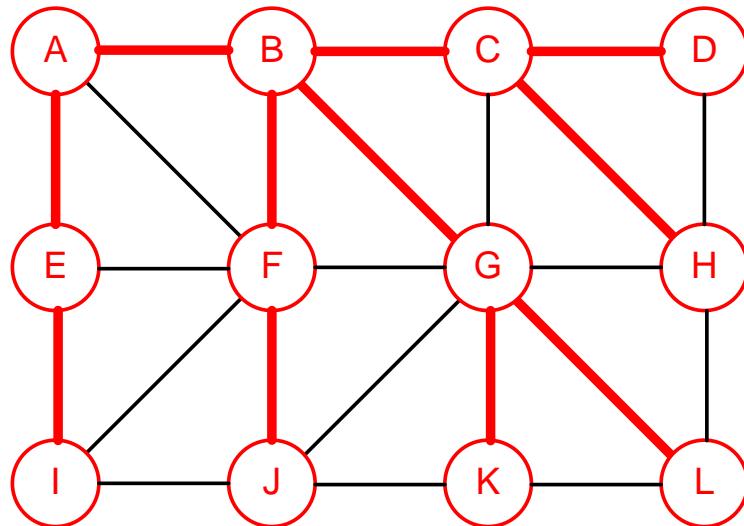
Destination	Le plus court chemin	Distance parcourue
A	E → D → A	7
C	E → F → C	4
G	E → H → G	2
I	E → H → I	2

Question 6 : Arbre sous-tendant minimum (16 points)

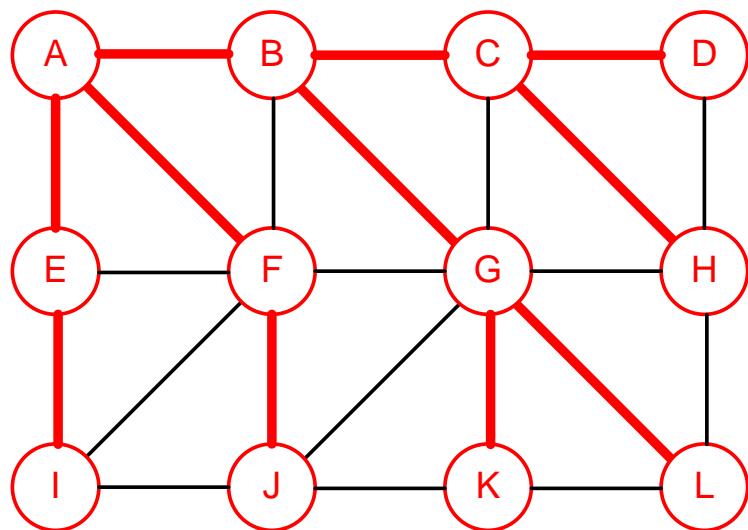
Considérez le graphe suivant dans lequel tous les arcs ont un poids unitaire.



- a) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe en utilisant l'algorithme de Prim. Partez du nœud A pour ce faire et visitez les voisins par ordre alphabétique.



b) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe précédent en utilisant l'algorithme de Kruskal. Traitez les arcs par ordre alphabétique. Par exemple, vu du nœud E, on traitera (A, E) avant (E, F), et (E, F) avant (E, I).



Annexe 1

```

public class BinaryHeap<AnyType extends Comparable<? super AnyType>>
{
    public BinaryHeap( )
    {
        this( DEFAULT_CAPACITY );
    }

    @SuppressWarnings("unchecked")
    public BinaryHeap( int capacity )
    {
        currentSize = 0;
        array = (AnyType[]) new Comparable[ capacity + 1 ];
    }

    @SuppressWarnings("unchecked")
    public BinaryHeap( AnyType [ ] items )
    {
        currentSize = items.length;
        array = (AnyType[]) new Comparable[ ( currentSize + 2 ) * 11 / 10 ];

        int i = 1;
        for( AnyType item : items )
            array[ i++ ] = item;
        buildHeap();
    }

    public void insert( AnyType x )
    {
        if( currentSize == array.length - 1 )
            enlargeArray( array.length * 2 + 1 );

        // Percolate up
        int hole = ++currentSize;
        for( ; hole > 1 && x.compareTo( array[ hole / 2 ] ) < 0; hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
        array[ hole ] = x;
    }

    @SuppressWarnings("unchecked")
    private void enlargeArray( int newSize )
    {
        AnyType [ ] old = array;
        array = (AnyType []) new Comparable[ newSize ];
        for( int i = 0; i < old.length; i++ )
            array[ i ] = old[ i ];
    }

    public AnyType findMin( )
    {
        if( isEmpty( ) )
            return null;
        return array[ 1 ];
    }
}

```

```

public AnyType deleteMin( )
{
    if( isEmpty( ) )
        return null;

    AnyType minItem = findMin( );
    array[ 1 ] = array[ currentSize-- ];
    percolateDown( 1 );

    return minItem;
}

private void buildHeap( )
{
    for( int i = currentSize / 2; i > 0; i-- )
        percolateDown( i );
}

public boolean isEmpty( )
{
    return currentSize == 0;
}

public void makeEmpty( )
{
    currentSize = 0;
}

private static final int DEFAULT_CAPACITY = 10;

private int currentSize;      // Number of elements in heap
private AnyType [ ] array; // The heap array

private void percolateDown( int hole )
{
    int child;
    AnyType tmp = array[ hole ];

    for( ; hole * 2 <= currentSize; hole = child )
    {
        child = hole * 2;
        if( child != currentSize &&
            array[ child + 1 ].compareTo( array[ child ] ) < 0 )
            child++;
        if( array[ child ].compareTo( tmp ) < 0 )
            array[ hole ] = array[ child ];
        else
            break;
    }
    array[ hole ] = tmp;
}

```

```
@SuppressWarnings("unchecked")
public AnyType troisieme() throws Exception
{
    if( currentSize < 3)
        throw new Exception("Le monceau possède moins de 3 éléments");

    int maxIndex = // masqué pour la question;

    AnyType[] arr = (AnyType[]) new Comparable[3];

    arr[ 0 ] = array[ 1 ];

    arr[ 1 ] = (array[ 2 ].compareTo( array[ 3 ] ) < 0 ) ?
        array[ 2 ] : array[ 3 ];

    arr[ 2 ] = (array[ 2 ].compareTo( array[ 3 ] ) >= 0 ) ?
        array[ 2 ] : array[ 3 ];

    for( int i=4; i <= maxIndex; i++)
    {
        if( array[ i ].compareTo( arr[ 1 ] ) < 0 )
        {
            // masqué pour la question
        }
        else if( array[ i ].compareTo( arr[ 2 ] ) < 0 )
        {
            // masqué pour la question
        }
    }

    return arr[2];
}
```

**Corrigé
examen final**

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20111
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo – responsable / Tarek Ould Bachir - chargé		A-416	7821
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Mercredi	04 mai 2011	2h30	13h30-16h00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Aucune	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
<input type="checkbox"/> Toute	<input type="checkbox"/> Toutes		
<input checked="" type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable		

Directives particulières

- Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Toutes vos réponses doivent être faites sur le questionnaire. Le cahier supplémentaire n'est pas à remettre à la fin de l'examen.

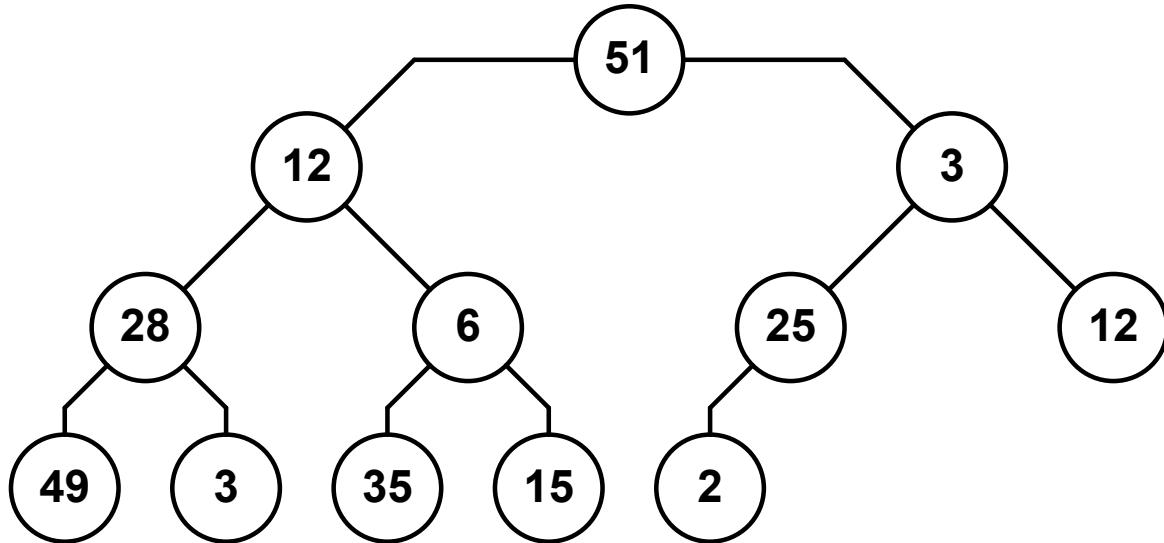
Bonne chance à tous!

Important	Cet examen contient 5 questions sur un total de 18 pages (excluant cette page) La pondération de cet examen est de 40 % Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non
------------------	---

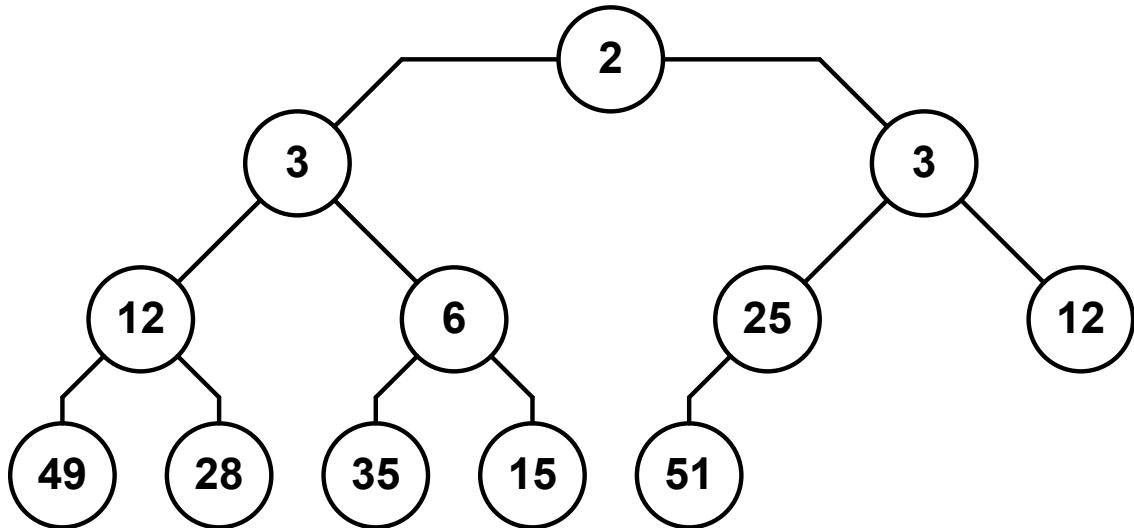
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(20 points)**

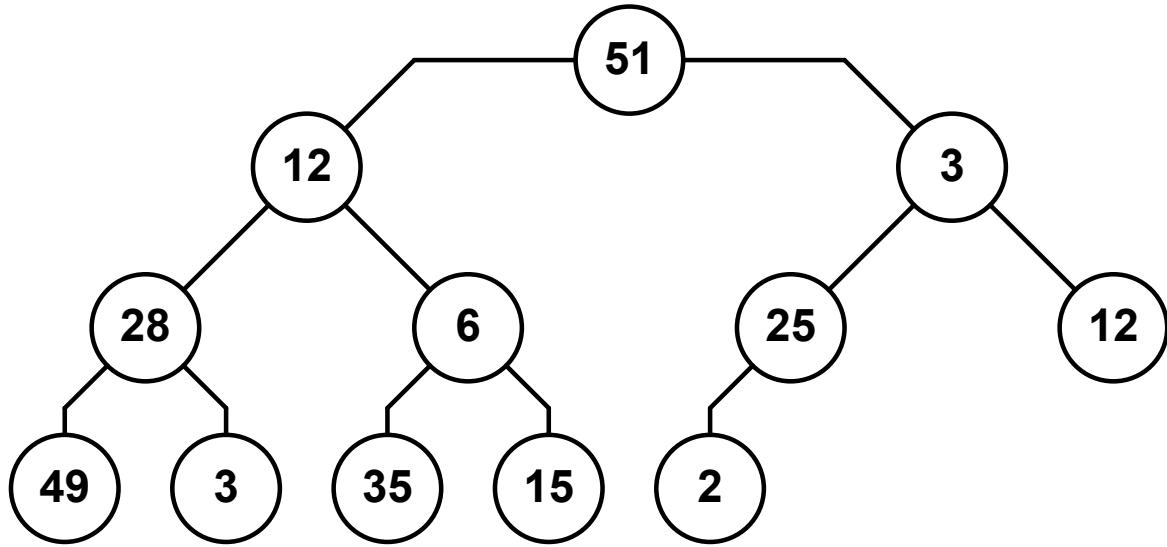
a) (5 pts) Construire, selon la technique vue en cours, un monceau MIN à partir de l'arbre binaire suivant :



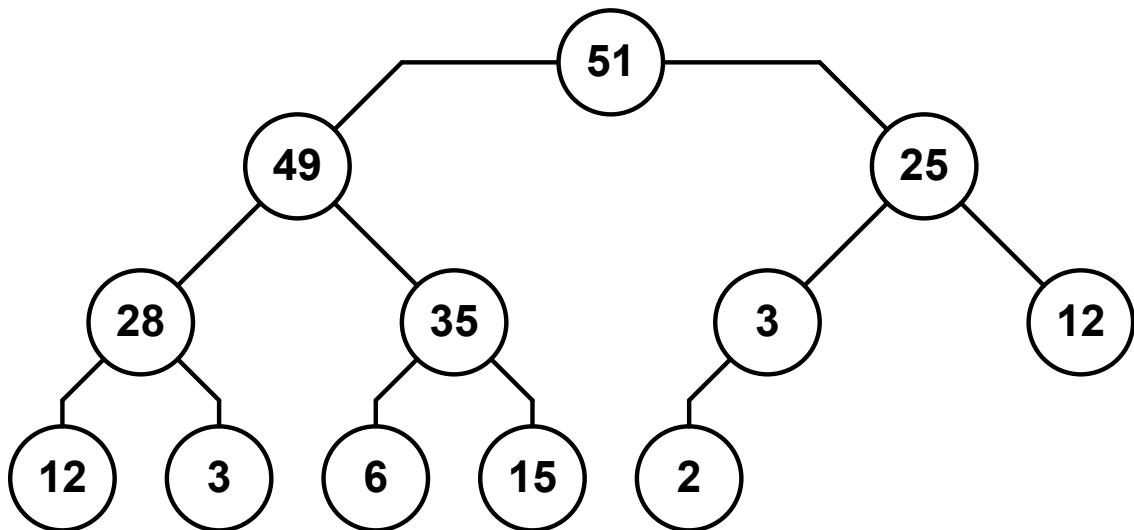
Monceau résultant :



a) (5 pts) Construire, selon la technique vue en cours, un monceau MAX à partir de l'arbre binaire suivant :

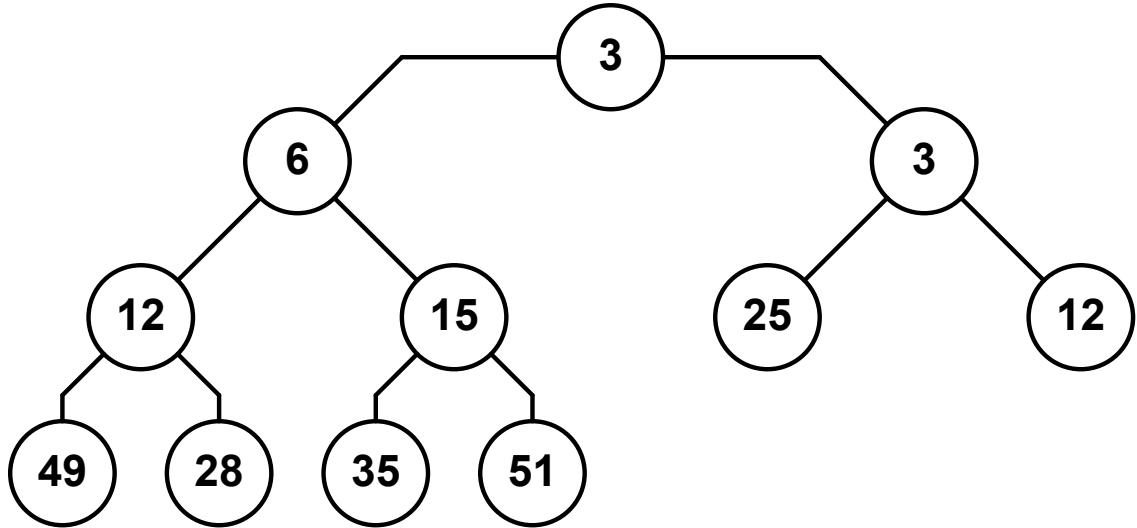


Monceau résultant :

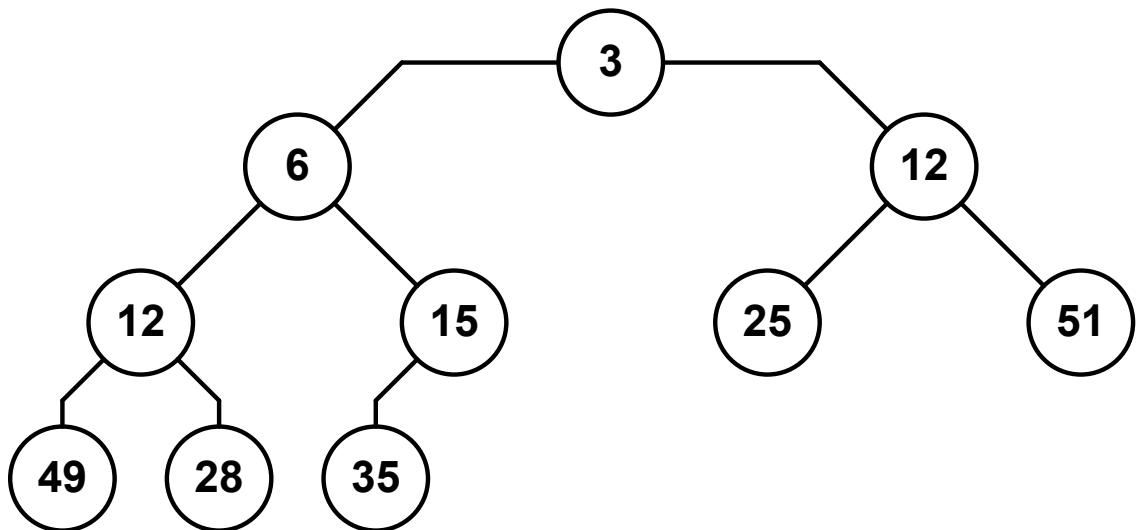


c) Dessiner l'état du monceau de la question 1.a) — monceau MIN que vous avez obtenu — suite à deux appels consécutifs à `deleteMin()` :

c.1) (2.5 pts) Monceau résultant du premier `deleteMin()`

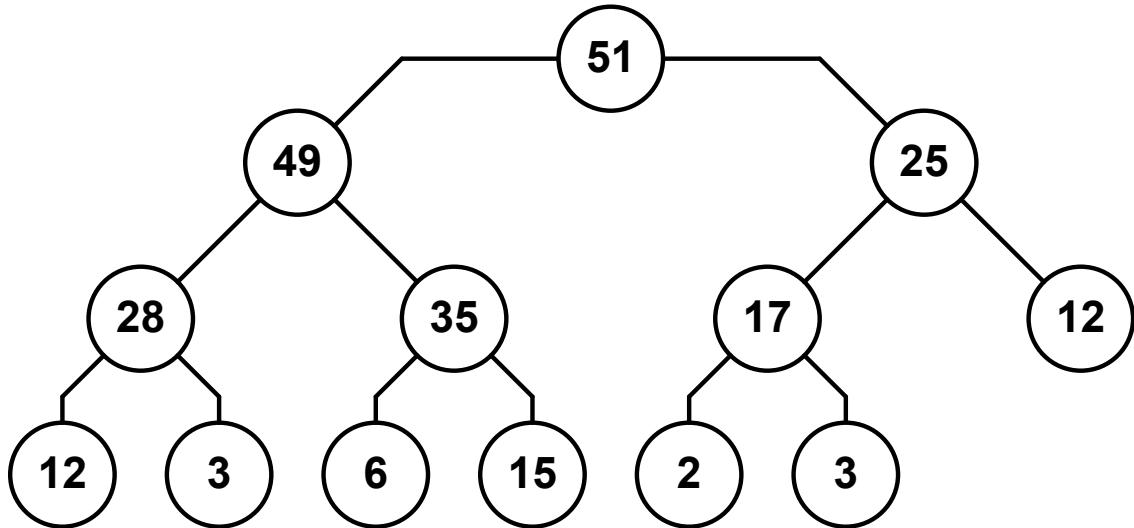


c.2) (2.5 pts) Monceau résultant du second `deleteMin()`

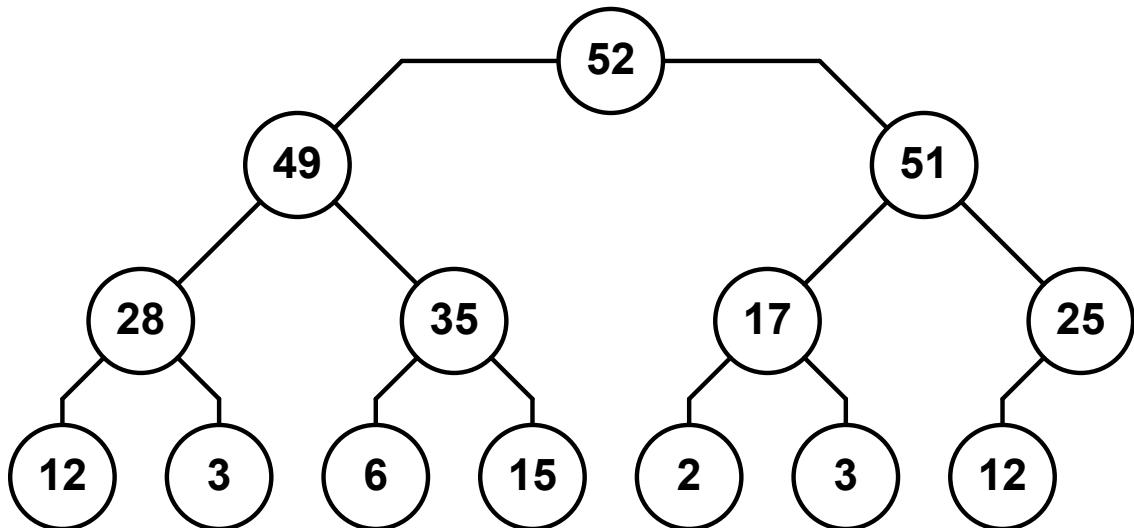


d) Dessiner l'état du dernier monceau de la question 1.b) — monceau MAX que vous avez obtenu — auquel on insère successivement les clés 17 et 52 :

d.1) (2.5 pts) Monceau résultant de insert(17)



d.2) (2.5 pts) Monceau résultant de insert(52)

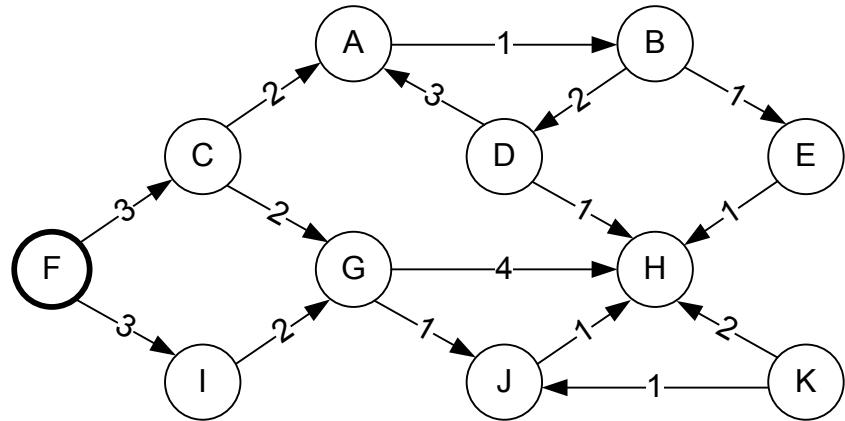


Question 2 : Plus court chemin d'un graphe acyclique

(25 points)

a) Tri topologique

a.1) (10 pts) Essayez de donner un ordre topologique au graphe suivant.



Noeud	1	2	3	4	5	6	7	8	9	10	11
A	2	2	2	1	1	1	1				
B	1	1	1	1	1	1	1				
C	1	0	0	0	0	0	0				
D	1	1	1	1	1	1	1				
E	1	1	1	1	1	1	1				
F	0	0	0	0	0	0	0				
G	2	2	2	1	0	0	0				
H	5	5	4	4	4	3	2				
I	1	0	0	0	0	0	0				
J	2	2	1	1	1	0	0				
K	0	0	0	0	0	0	0				
Entrée	F, K	C, I	-	-	G	J	-				
Sortie	F	K	C	I	G	J	Fin				

a.2) (5 pts) Peut-on dire de ce graphe qu'il est acyclique ? Pourquoi ?

Non. L'exécution de l'algorithme n'a pas abouti. D'ailleurs il apparaît après attentive inspection que A, B, D forment un cycle.

b) Nous voulons trouver les plus courts chemins depuis le nœud F jusqu'à l'ensemble des nœuds en appliquant l'algorithme de Dijkstra.

b.1) (5.5 pts) Continuez l'exécution de l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de F.

Nœud	Connu	Dist min.	Parent
A	✓	$\infty, 5$	C
B	✓	$\infty, 6$	A
C	✓	$\infty, 3$	F
D	✓	$\infty, 8$	B
E	✓	$\infty, 7$	B
F	✓	0	-
G	✓	$\infty, 5$	C
H	✓	$\infty, 9, 7$	G, J
I	✓	$\infty, 3$	F
J	✓	$\infty, 6$	G
K		$\infty,$	

File de priorité

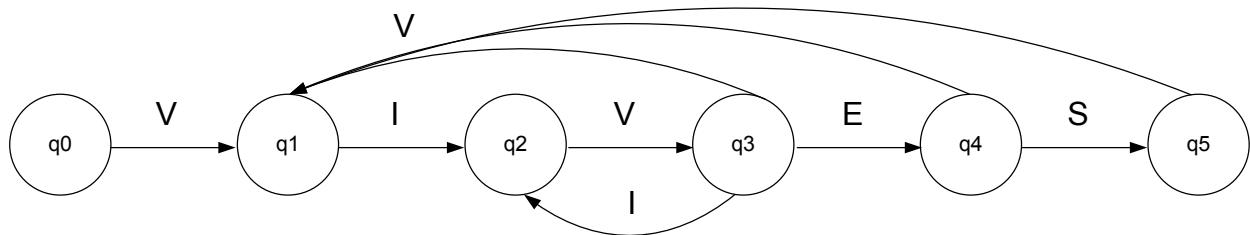
b.2) (4.5 pts) Détaillez chacun des chemins les plus courts trouvés :

Nœud	Le plus court chemin	Distance parcourue
A	F → C → A	5
B	F → C → A → B	6
C	F → C	3
D	F → C → A → B → D	8
E	F → C → A → B → E	7
F	F → F	0
G	F → C → G	5
H	F → C → G → J → H	7
I	F → I	3
J	F → C → G → J	6
K	Aucun chemin	-

Question 3 : Recherche de patrons par automate à états finis (15 points)

On vous demande de retrouver le patron $P[1:5]=\text{VIVES}$ dans un texte.

- a) (5 pnts) Dessiner le diagramme d'états de l'automate à états finis permettant de ce faire :



- b) (5 pnts) Donner la table de transitions de l'automate recherché.

q\a	V	I	E	S	Autre
0	1	0	0	0	0
1	1	2	0	0	0
2	3	0	0	0	0
3	1	2	4	0	0
4	1	0	0	5	0
5	1	0	0	0	0

- c) Dans quel état sera votre automate une fois arrivé à la fin de la phrase suivante :

$T[1 : 34] = \text{« VIVIANE VIT AUJOURD'HUI À VAL-D'OR »}$

État $q0$

Question 5 : Programmation dynamique

(20 points)

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_5 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 2 \times 4 ; A_2 : 4 \times 2; A_3 : 2 \times 3; A_4 : 3 \times 2; A_5 : 2 \times 2$$

Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5
1	0	16	28	36	44
2		0	24	28	36
3			0	12	20
4				0	12
5					0

s	1	2	3	4	5
1		1	2	2	4
2			2	2	2
3				3	4
4					4
5					

Compléter cette table pour répondre aux questions suivantes :

Rappel : $m[i, j] = \min \{m[i, k] + m[k+1, j] + p_{i-1}p_kp_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

a) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_5 . Donnez son coût.

$$(A_1A_2)(A_3A_4))(A_5)$$

Coût : 44

b) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

$$(A_1A_2)(A_3A_4)$$

Coût : 36

c) (4 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_3 . Donnez son coût.

$$(A_1A_2)(A_3)$$

Coût : 25

d) (4 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_5 . Donnez son coût.

$$(A_2)((A_3A_4)(A_5))$$

Coût : 36

e) (4 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_4 . Donnez son coût.

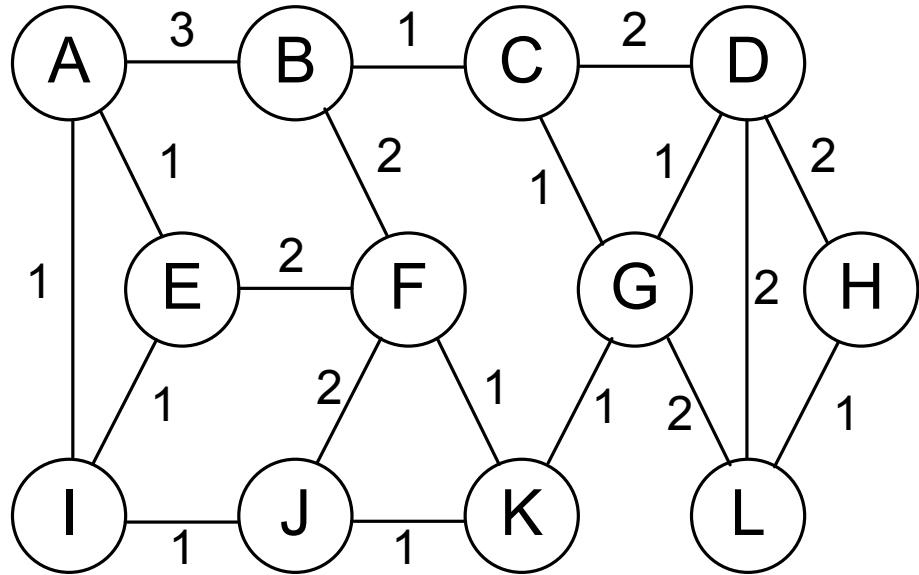
$$(A_2)(A_3A_4)$$

Coût : 28

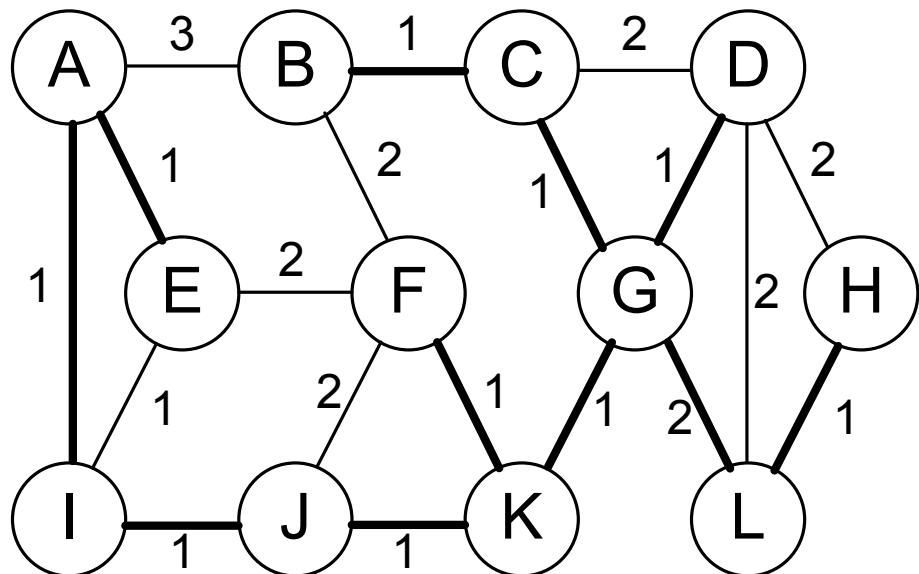
Question 5 : Arbre sous-tendant minimum

(20 points)

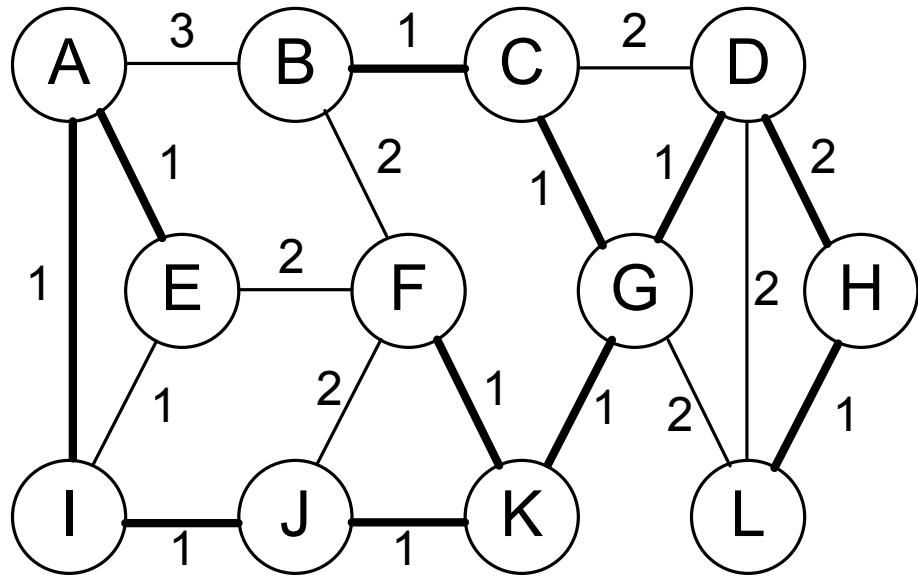
Considérez le graphe suivant :



- a) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe précédent en utilisant l'algorithme de Prim. Partez du nœud A pour ce faire et visitez les voisins par ordre alphabétique.



b) (8 pts) Dessinez l'arbre sous-tendant minimum du graphe précédent en utilisant l'algorithme de Kruskal. En plus de leur poids, traitez les arcs par ordre alphabétique, c'est-à-dire pour un poids identique on traitera (A, E) avant (A, I), (A, E) avant (B, C) et (A, I) avant (B, C).



c) (4 pts) Comparez les solutions 5.a) et 5.b). Est-ce un résultat auquel on pouvait s'attendre? Justifiez brièvement.

Les deux solutions sont différentes. Comme ce graphe a beaucoup d'arcs de même poids, ce résultat est attendu. Les deux algorithmes renvoient cependant des arbres de mêmes poids.

Prim : 12

Kruskal : 12

Ce qui est cohérent.

Corrigé examen final

INF2010

Sigle du cours

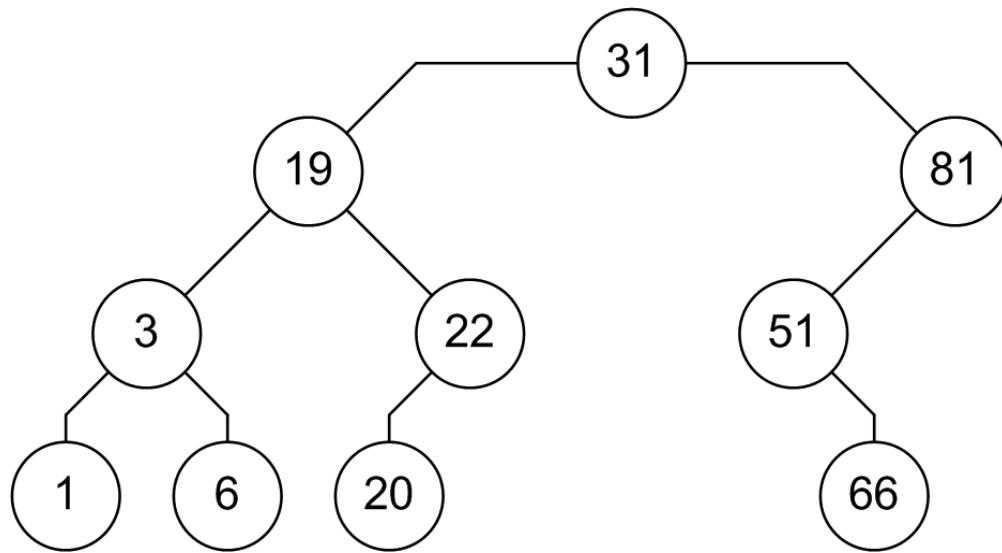
<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20103
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo – responsable / Tarek Ould Bachir - chargé		A-201	
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Jeudi	9 décembre 2010	2h30	13h30-16h00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	
<i>Directives particulière</i> Les calculatrices, agendas électroniques ou téléavertisseurs sont interdits.			
<i>Bonne chance à tous!</i>			
Important	Cet examen contient <input type="checkbox"/> questions sur un total de XX pages (excluant cette page)		
	La pondération de cet examen est de 40 %		
	Vous devez répondre sur : <input type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input checked="" type="checkbox"/> les deux		
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non		

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Arbre Splay**(14 points)**

a) Considérez l'arbre Splay suivant et répondez aux énoncés par vrai ou par faux.



a.1) (2 pts) Ceci est un arbre qui a la structure d'un arbre complet :

Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

a.2) (2 pts) Ceci est un arbre qui a la structure d'un arbre de recherche :

Vrai	<input checked="" type="checkbox"/>
Faux	<input type="checkbox"/>

a.3) (2 pts) Ceci est un arbre qui a la structure d'un arbre AVL :

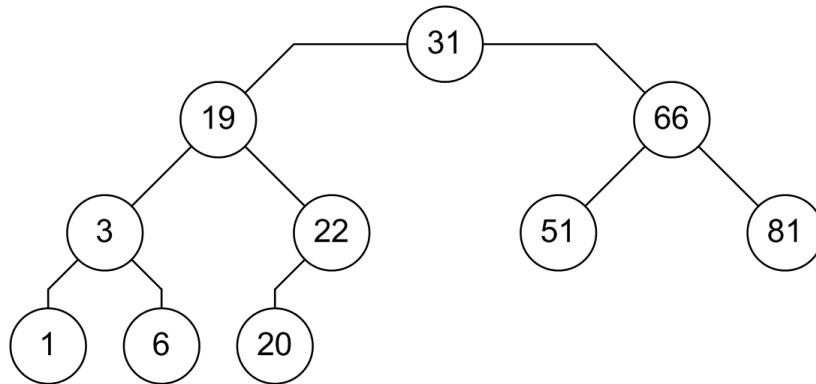
Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

b) Proposez une modification à l'arbre précédent qui le transforme pour rencontrer:

b.1.1) (1 pts) Pour que cet arbre ait la structure d'un arbre complet, il suffit :

- de rien faire, cet arbre a déjà la structure d'un arbre complet
- d'effectuer une rotation simple à droite autour du nœud 51
- d'effectuer une rotation double, à droite autour du nœud 51, à gauche autour de 81
- d'effectuer une rotation simple à gauche autour du nœud 51
- d'effectuer une rotation double à gauche autour du nœud 51, à droite autour de 81**

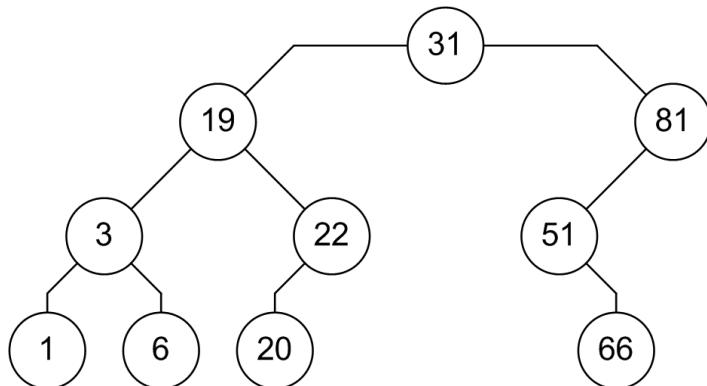
b.1.2) (1 pts) Dessinez l'arbre ainsi obtenu :



b.2.1) (1 pts) Pour que cet arbre ait la structure d'un arbre de recherche, il suffit :

- de rien faire, cet arbre a déjà la structure d'un arbre de recherche**
- d'effectuer une rotation simple à droite autour du nœud 51
- d'effectuer une rotation double, à droite autour du nœud 51, à gauche autour de 81
- d'effectuer une rotation simple à gauche autour du nœud 51
- d'effectuer une rotation double à gauche autour du nœud 51, à droite autour de 81

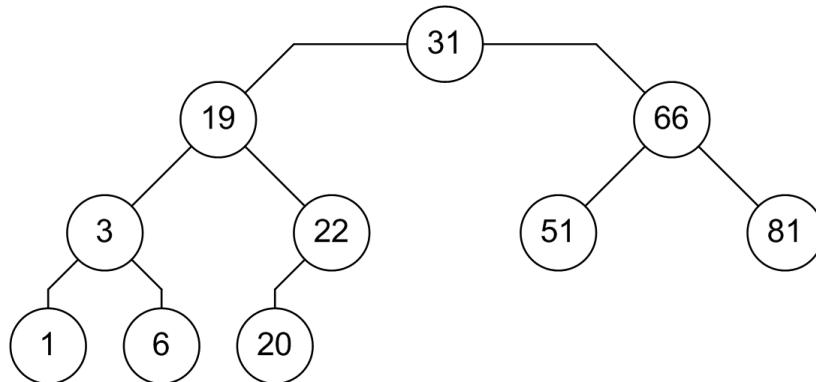
b.2.2) (1 pts) Dessinez l'arbre ainsi obtenu :



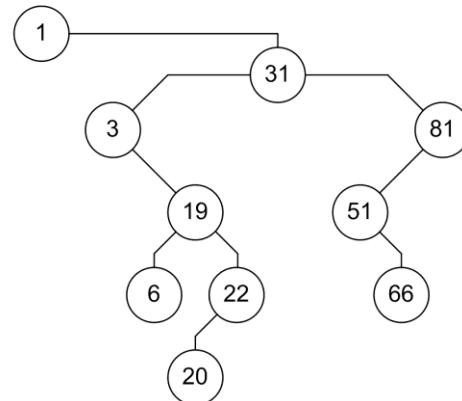
b.3.1) (1 pts) Pour que cet arbre ait la structure d'un arbre AVL, il suffit :

-
-
-
-
- d'effectuer une rotation double à gauche autour du nœud 51, à droite autour de 81**

b.3.2) (1 pts) Dessinez l'arbre ainsi obtenu :



c) On effectue un get(1) sur l'arbre Splay du départ et on obtient ce qui suit.



c.1) (1 pt) Expliquez l'intérêt des arbres Splay s'ils produisent des arbres aussi mal débalancés.

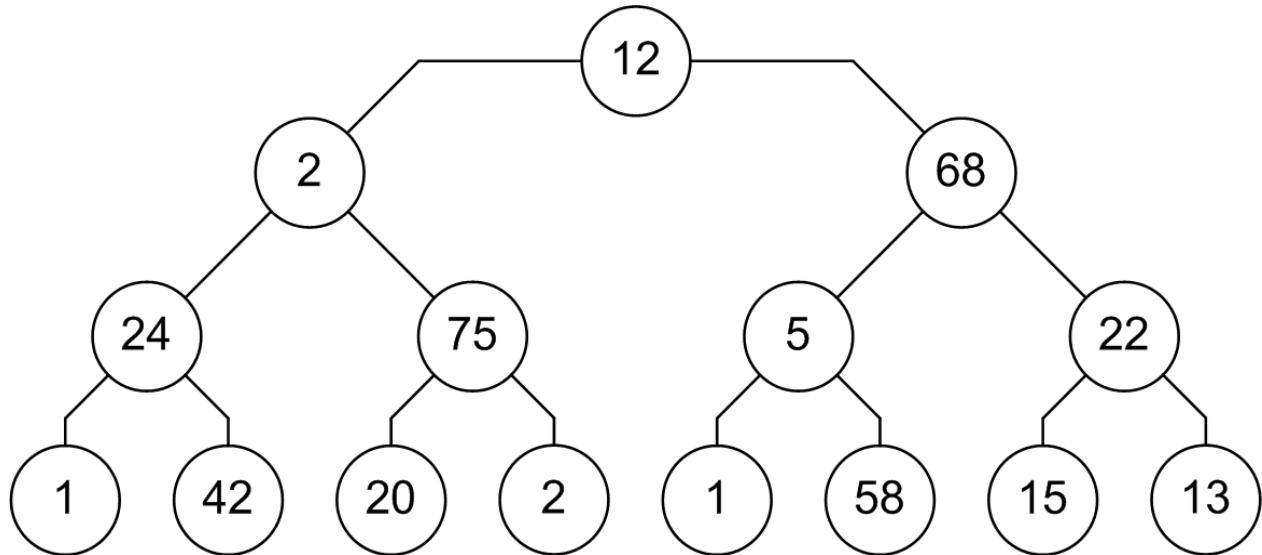
Les arbres Splay partent de l'hypothèse qu'un appel sur un nœud sera suivi de plusieurs appels sur ce même nœud. Ainsi, après un get, les autres nœuds peuvent se retrouver deux niveaux plus bas dans l'arbre et l'arbre devenir débalancé. L'intérêt est que les appels suivant sur le même nœud se feront en O(1), pour une complexité amortie de O(log(n)).

c.2) (1 pt) Pensez-vous que ce résultat (l'arbre obtenu) soit correct ? Discutez.

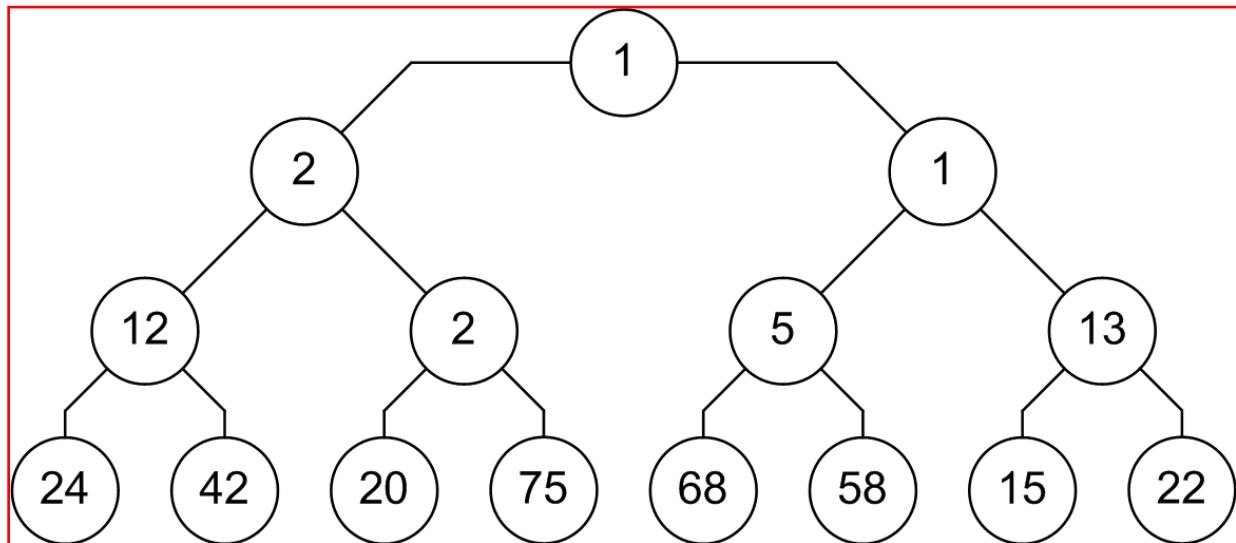
Ce résultat est en effet correct. C'est la réponse exacte en vérité.

Question 2 : Monceaux**(16 points)**

a) (4 pts) Construire, selon la technique vue dans le cours, un monceau à partir de l'arbre binaire suivant :

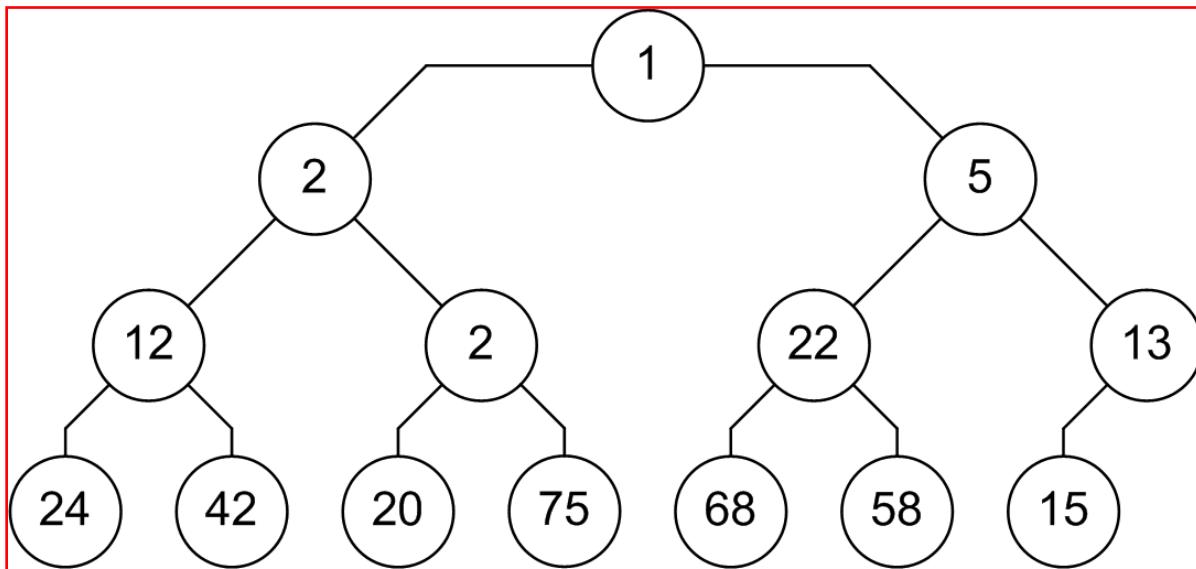


Monceau résultant :

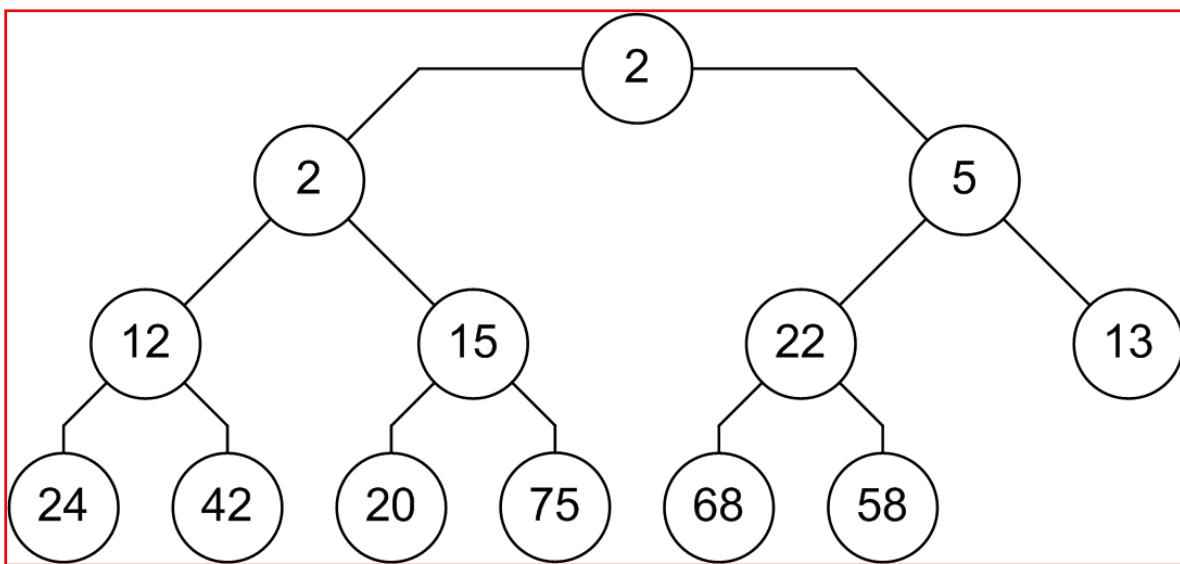


b) Dessiner l'état de ce monceau après deux appels consécutifs à `deleteMin()` :

b.1) (3 pts) Monceau résultant du premier `deleteMin()`

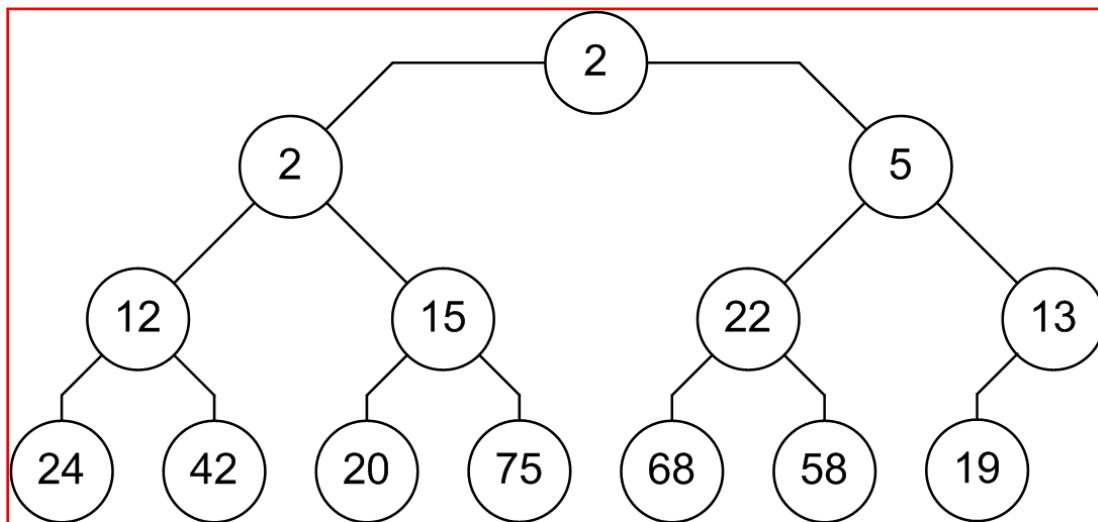


b.2) (3 pts) Monceau résultant du second `deleteMin()`



c) Dessiner l'état du dernier monceau obtenu auquel on insère la clé 19

c.1) (3 pts) Monceau résultant de insert(19)



c.2) (3 pts) Dessiner l'état du tableau contenant le monceau résultant de l'insertion de 19 :

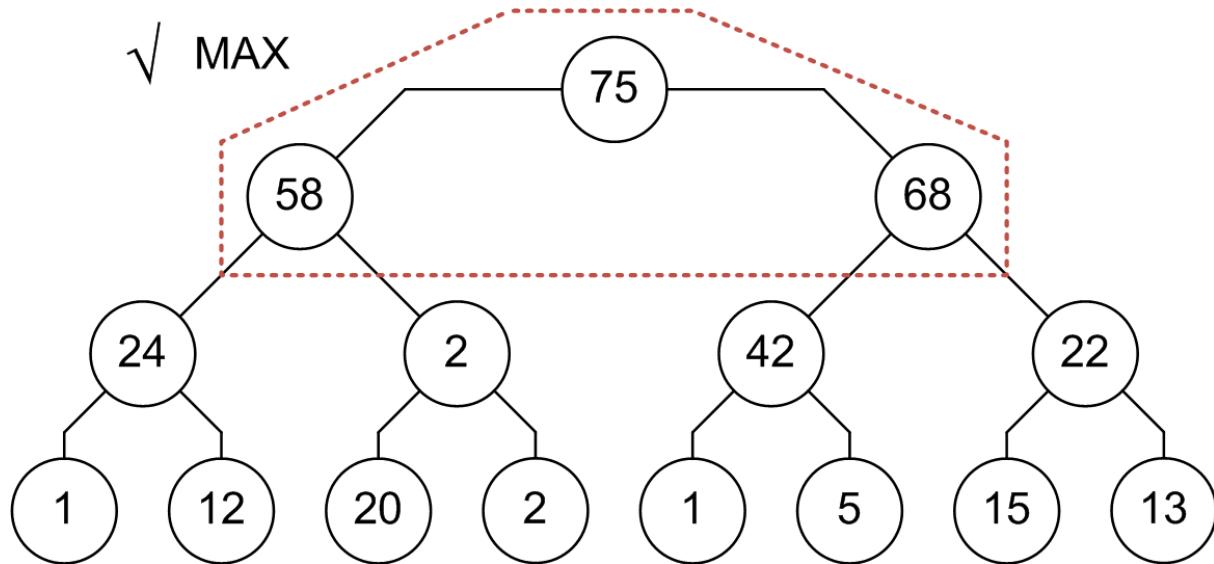
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2	2	5	12	15	22	13	24	42	20	75	68	58	19	

Question 3 : Tri par monceau**(10 points)**

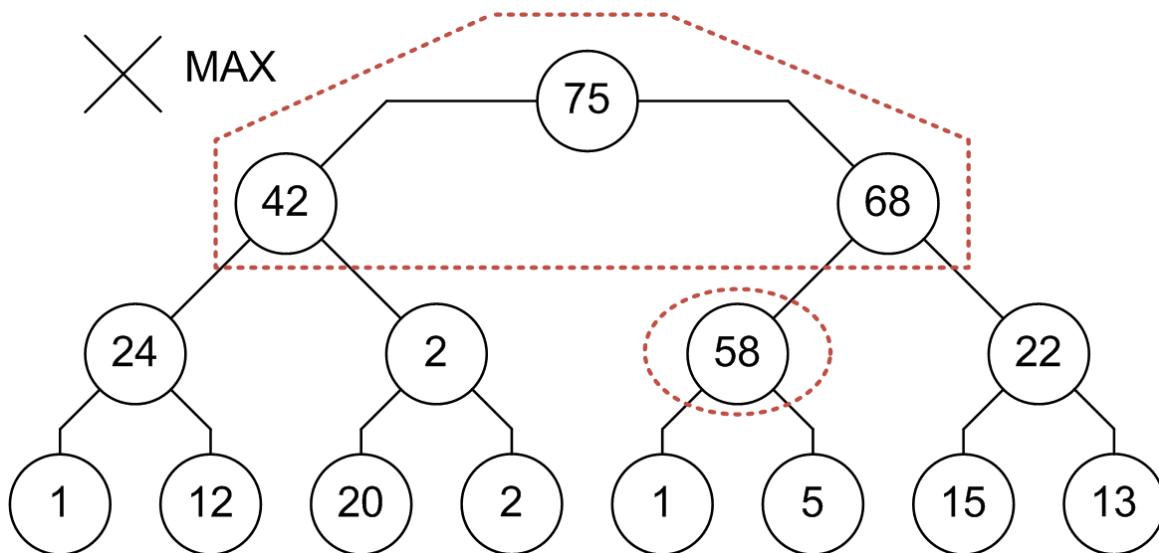
On désire trouver les trois plus grands éléments d'un vecteur d'entiers. On vous suggère, pour ce faire, d'utiliser un monceau MAX et de trouver les trois plus grands éléments au niveau de la racine et de ses deux enfants.

a) Illustriez la pertinence de cette stratégie en construisant un monceau MAX à partir de chacun des vecteurs suivants :

a.1) (3 pts) 12, 2, 68, 24, 75, 5, 22, 1, 58, 20, 2, 1, 42, 15, 13



a.2) (3 pts) 12, 2, 68, 24, 75, 5, 22, 1, 42, 20, 2, 1, 58, 15, 13



b) (2 pts) Commentez cette stratégie. Est-elle justifiée ou non? Expliquez clairement votre réponse.

Il est évident avec les deux cas étudiés que cette stratégie ne marche pas toujours. Les trois nombres plus grands se retrouvent entre les niveaux 0 et 2, et ne sont donc pas forcément parmi la racine et ses enfants.

c) (2 pts) Proposez une stratégie que vous jugez plus efficace. Justifiez votre réponse.

Il serait possible de construire un monceau MAX, d'effectuer un deleteMax() trois fois. Les nœuds retirés seraient alors les trois plus grands. C'est le tri par monceau partiel.

Question 4 : DP-Matching (10 points)

a) (4 pts) En utilisant le tableau suivant, retrouver la plus longue sous-séquence commune aux chaînes d'entrée $X = \text{« ORANAISES »}$ et $Y = \text{« CHAGRINEUSES »}$:

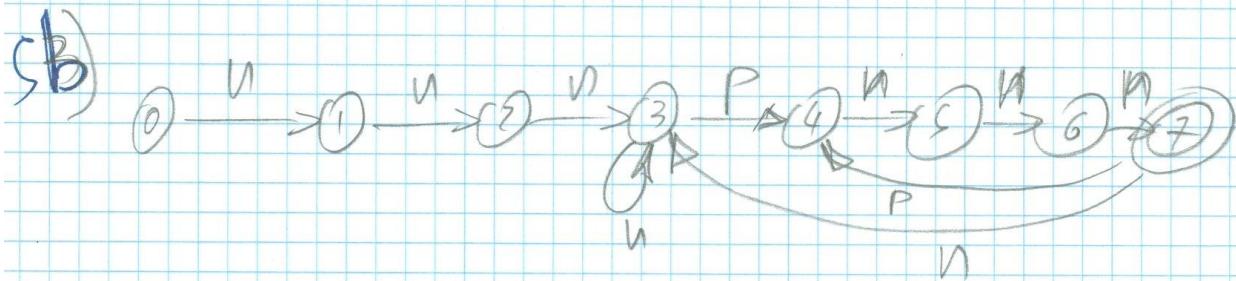
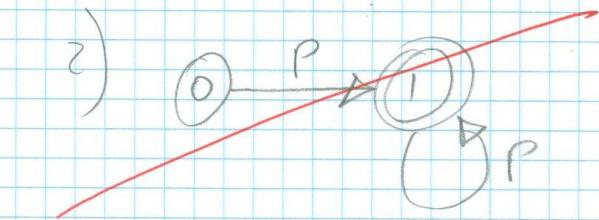
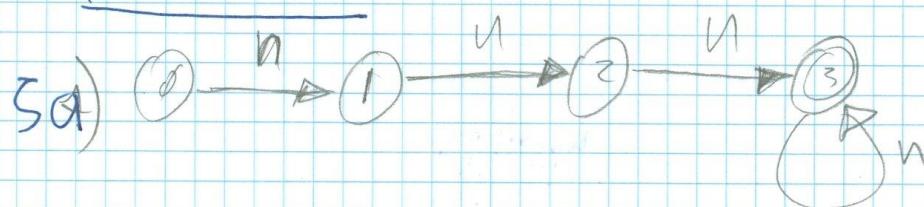
	Y	C	H	A	G	R	I	N	E	U	S	E	S
X	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
R	0	↑	↑	↑	↑	↖	↖	↑	↑	↑	↑	↑	↑
A	0	0	0	1	1	1	1	1	1	1	1	1	1
N	0	↑	↑	↑	↑	↑	↑	2	2	2	2	2	2
A	0	0	0	1	1	1	1	2	2	2	2	2	2
I	0	↑	↑	↑	↑	↑	2	2	2	2	2	2	2
S	0	↑	↑	↑	↑	↑	1	2	2	2	3	3	3
E	0	↑	↑	↑	↑	↑	1	2	3	3	3	4	4
S	0	↑	↑	↑	↑	↑	1	2	3	3	4	4	5

Plus longue sous-séquence commune et sa longueur:

RNSES, longueur 5

b) (6 pnts) Proposez trois autres PLSC des entrées $X = \text{« ORANAISES »}$ et $Y = \text{« CHAGRINEUSES »}$:

**ANSES
RISES
AISES**

Question 5

	δ	l'arbre	l'arbre initial
5b)	0 u 1		T
	a		T
	p		T
1	u 2		
	a		
	p		
2	u 3		
	a		
	p		
3	u 3	3	
	a		
	p		
4	u s		
	a		
	p		
5	u 6		
	a		
	p		
6	u 7		
	a		
	p		
7	u 3	T	
	a	T	
	p	T	

Question 6~~(B1)~~ / 61

$$\begin{aligned} a &= \emptyset \\ b &= 1 \end{aligned}$$

~~RBSV~~

$$1) RK_{s+1} = d(RK_s - h \cdot T[s+1]) + T[m+s+1] \bmod q$$

$$2) h = d^{m-1} \bmod q$$

$$3) d = 256$$

$$4) q = 7$$

~~RK modif = 000~~

$$6a) \quad \text{RBSV} \quad 1) RK_{s+1} = 2 \cdot (RK_s - 2T[s+1]) + T[m+s+1]$$

$$2) h = 2$$

$$3) d = 2$$

$$4) \quad \diagup$$

$$\left\{ \begin{array}{l} ab \\ 01 \end{array} \right\} \rightarrow 1$$

$$RK_{\text{modif}} = 1$$

~~(B2)~~ (même)

$$6b) \quad 1) RK_{s+1} = 2(RK_s - 2T[s+1]) + T[m+s+1]$$

$$2) h = 2$$

$$3) d = 2$$

$$4) \quad \diagup$$

$$\left\{ \begin{array}{l} ba \\ 10 \end{array} \right\} \rightarrow 2$$

$$RK_{\text{modif}} = 2$$

<u>Question 7a)</u>		<u>d</u>	<u>p</u>	<u>k</u>							
<u>v</u>	<u>d v</u>	<u>p v</u>	<u>d v</u>	<u>p v</u>	<u>k v</u>	<u>d v</u>	<u>p v</u>	<u>k v</u>	<u>d</u>	<u>p</u>	<u>k</u>
0	∅	uiP	∅	-	v	∅	-	v	0	-	v
1	g0		2	∅		2	∅	v	2	∅	v
2									1	1	∅
3									∞		
4									∞		
5									∞		
6									1	1	∅
7	1	0							∞		

<u>v</u>	<u>d p k</u>									
0	0 - v	0 - v	0 - v	0 - v	0 - v	0 - v	0 - v	0 - v	0 - v	0 - v
1	2 ∞ v	2 ∞ v	2 0 v	2 0 v	2 0 v	2 0 v	2 0 v	2 0 v	2 0 v	2 0 v
2	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v
3	∞ ∞ -	∞ -	∞ -	4 5	4 5	4 5	4 5	4 5 v	4 5 v	4 5 v
4	∞ -	∞ -	∞ -	∞ -	5 7	5 7	5 7	5 7	2 3	2 3
5	2 2 2	2 6 2	2 2 2	2 2 v	2 2 v	2 2 v	2 2 v	2 2 v	2 2 v	2 2 v
6	1 1 1	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v	1 1 v
7	∞ -	4 6	4 6	3 5	3 5	3 5 v	3 5 v	3 5 v		

	d	p	k	
0	0	-	v	
1	2	0	v	
2	1	1	v	
3	4	5	v	
4	2	3	v	
5	4	4	v	
6	1	1	v	
7	3	5	v	

Hablaen 7.1

Question 7.b

$\mathcal{O}[0, s]$

3 [1, 4]

2 [2,3]

1 [6, 7]

$$2 \quad \emptyset$$

1

4 [8, 15]

1

S [a, 147]

S [a, 147]

l *l*

① ② 6

10

⑦ ⑧

2 3

8/5

8 [16, 17]

(7) (4)

1 1

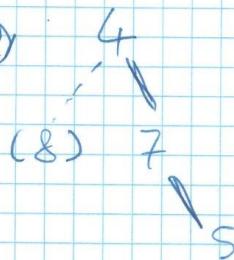
Question 7b

etape	U	J	V	V	V	V	o	v	lebien
0	0	1	2	3	4	5	6	7	8
fin	5	7	3	4	15	14	11	13	17

7.2

CFC : ~~∅~~ ∅

noeuds : 8

CFC : ~~∅~~ 1noeuds : ~~∅~~ 1

CFC : 4

noeuds : 4

(1) 2 3

(1)(3)(5)(6)

CFC : 2

noeuds : 6

5

CFC : 3

noeuds : 1

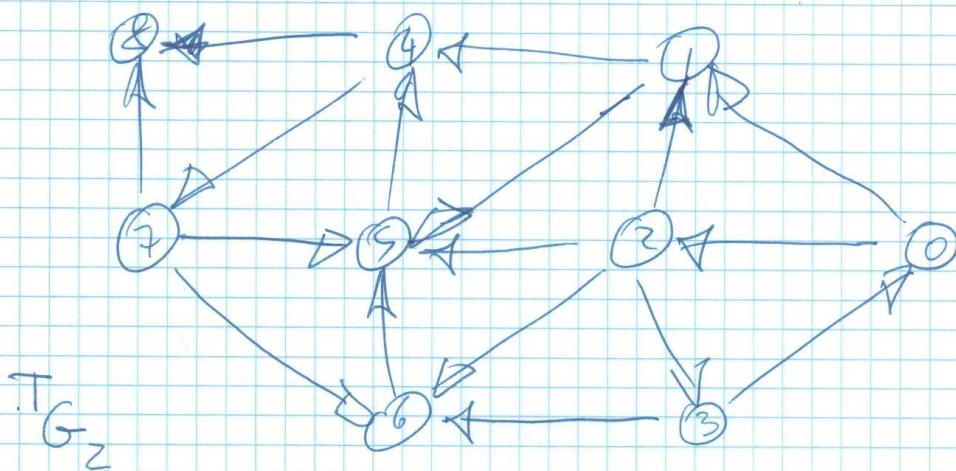
(4) (5)

Q7 b)

$${}^T V_2 = \{0, 1, 2, 3, 4, 5, 6, 7\} \quad \underline{\text{Problème 7.3}}$$

$$\begin{aligned} {}^T E_2 = & \{(3, 0), (2, 1), (0, 1), (0, 2), (2, 3), \\ & (5, 4), (1, 4), (1, 5), (2, 5), (6, 5), (7, 5), \\ & (7, 6), (3, 6), (2, 6), (4, 7), (4, 8), (7, 8)\} \end{aligned}$$

Problème 7.4



a)

Question 7(b)

V	0	/	1)	2	/	3	/	4	/	5	/	6	/	7	/	8
CFQ	4		3		4		4		1		1		2		1		Ø

téléphone 75

**Corrigé
examen final**

INF2010

Sigle du cours

Identification de l'étudiant(e)

Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20101
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Ettore Merlo – responsable / Tarek Ould Bachir - chargé		B-512	
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Jeudi	22 avril 2009	2h30	13h30-16h00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Aucune	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	
<input type="checkbox"/> Toute	<input type="checkbox"/> Toutes		
<input checked="" type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable		

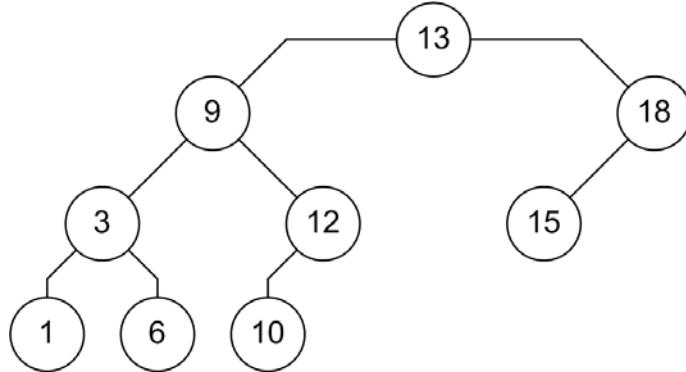
Directives particulières

- Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Nous ne récupérerons pas le cahier supplémentaire à la fin de l'examen. Pas conséquent, écrivez toutes vos réponses dans ce questionnaire.

Bonne chance à tous!

Important	Cet examen contient 5 questions sur un total de 15 pages (excluant cette page)
	La pondération de cet examen est de 40 %
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Arbre Splay**(20 points)****1.1) (6 pts)** Considérez l'arbre suivant et répondez aux énoncés par vrai ou par faux.**1.1.1) (2 pts)** Ceci est un arbre complet :

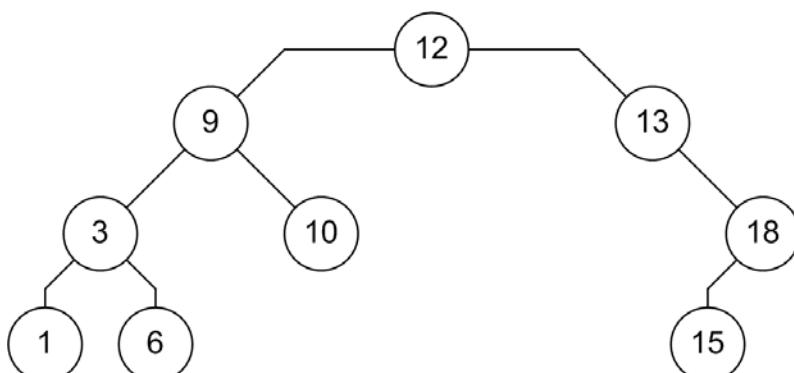
Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

1.1.2) (2 pts) Ceci est un arbre de recherche :

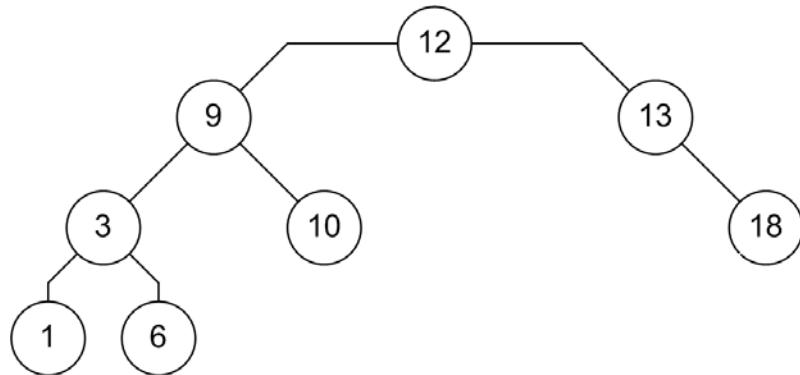
Vrai	<input checked="" type="checkbox"/>
Faux	<input type="checkbox"/>

1.1.3) (2 pts) Ceci est un arbre AVL :

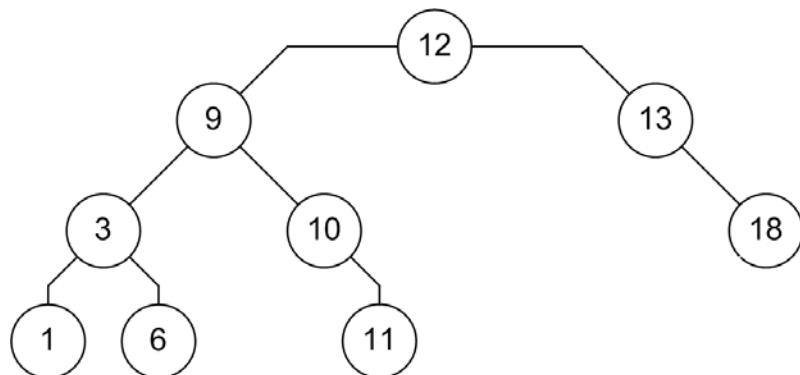
Vrai	<input checked="" type="checkbox"/>
Faux	<input type="checkbox"/>

1.2) (14 pts) L'arbre donné à la question 1.1) est un arbre Splay sur lequel nous voulons exécuter (dans l'ordre) une série d'opérations. Dessinez l'état de l'arbre après chacune des opérations suivantes :**1.2.1) (2 pts)** Get(12) :

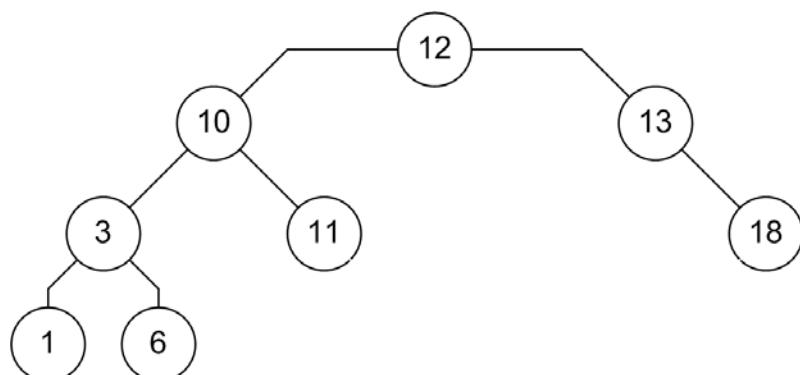
1.2.2) (2 pts) Delete(15) :



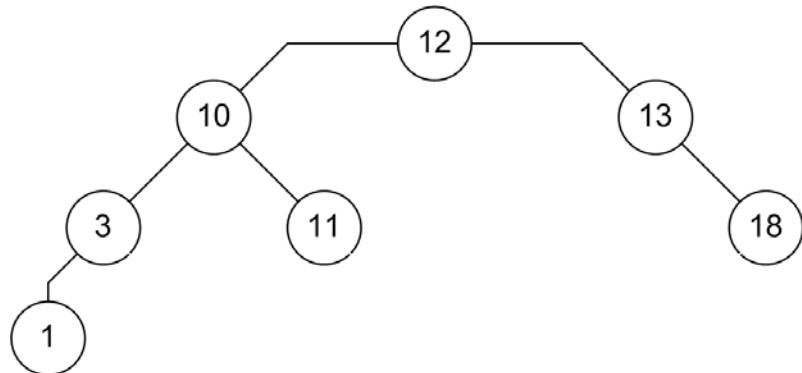
1.2.3) (2 pts) Insert(11) :



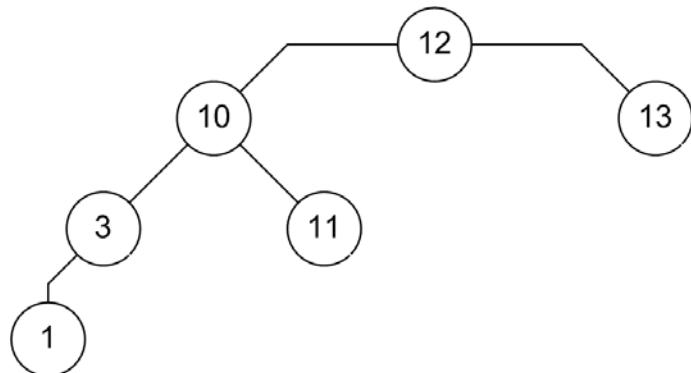
1.2.4) (2 pts) Delete(9) :



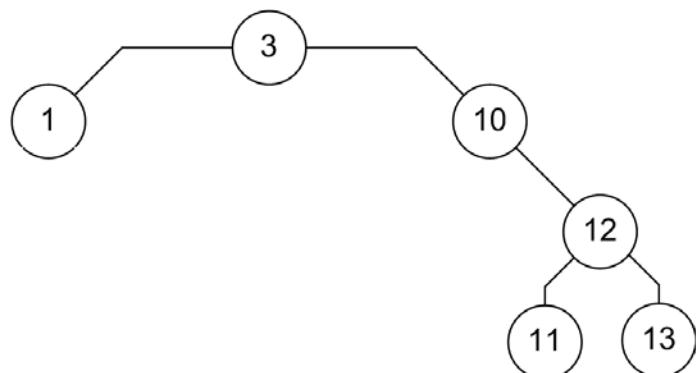
1.2.5) (2 pts) Delete(6) :



1.2.6) (2 pts) Delete(18) :

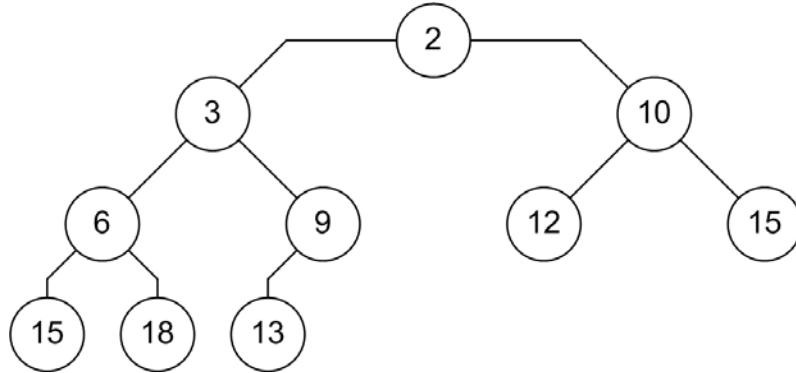


1.2.7) (2 pts) Get(3) :



Question 2 : Monceau**(25 points)**

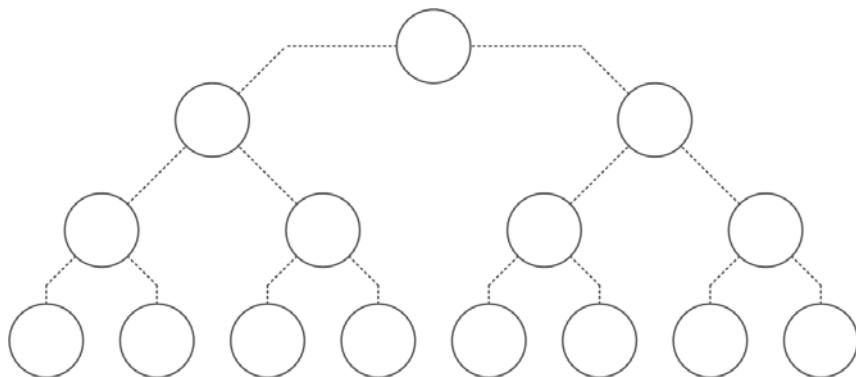
- 2.1) (11 pts)** Considérez le monceau suivant sur lequel nous voulons exécuter (dans l'ordre) une série d'opérations. Dessinez l'état de l'arbre après chacune des opérations qui suivent. Dans chaque cas, dessinez l'état en mémoire du tableau contenant le monceau.



- 2.1.1) (1 pts)** État du tableau de départ :

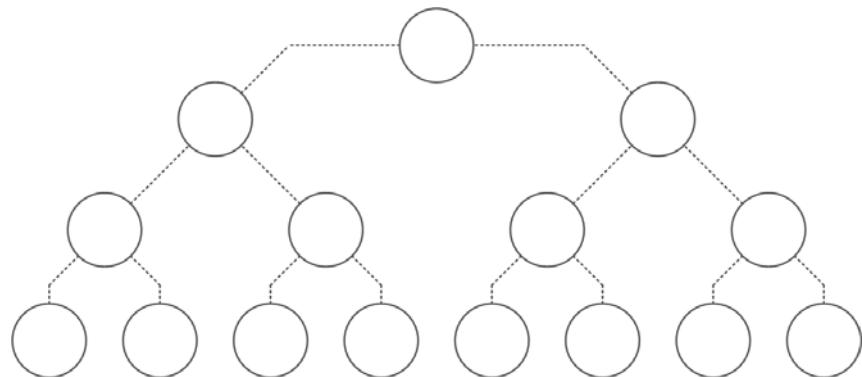
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	2	3	10	6	9	12	15	15	18	13								

- 2.1.2) (2 pts) add(15) :**



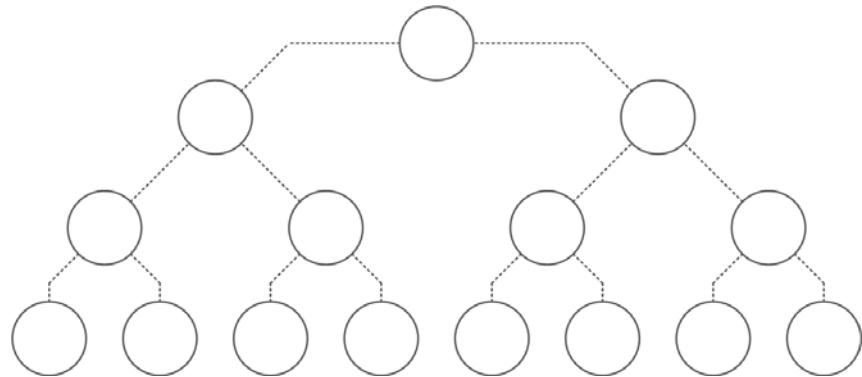
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	2	3	10	6	9	12	15	15	18	13	15							

2.1.3) (2 pts) add(2) :



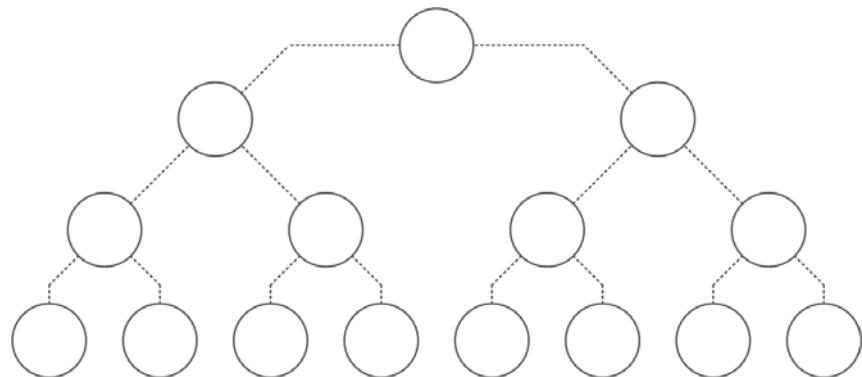
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	2	3	2	6	9	10	15	15	18	13	15	12						

2.1.4) (2 pts) remove() :



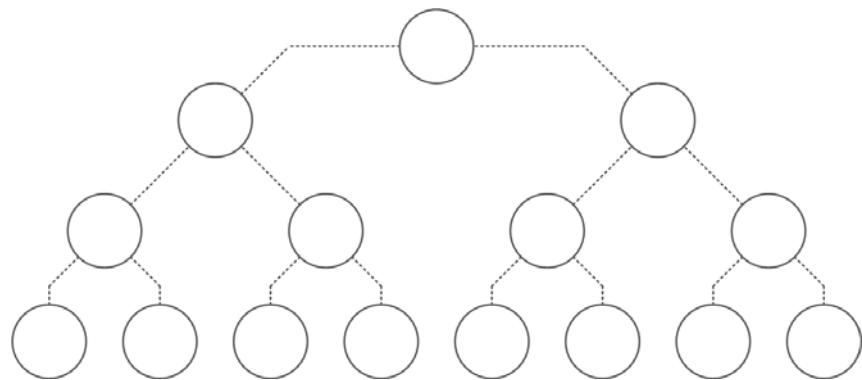
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	2	3	10	6	9	12	15	15	18	13	15							

2.1.5) (2 pts) remove() :



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	3	6	10	15	9	12	15	15	18	13								

2.1.6) (2 pts) remove() :



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valeurs	X	6	9	10	15	13	12	15	15	18									

2.2) L'implémentation du monceau en Java vous est donnée à l'annexe I.

2.2.1) (7 pts) Écrivez le code en java d'une méthode permettant de trouver la valeur maximale du monceau.

```
int findMax( )
{
    int x = array[1];

    for( int i = currentSize/2; i <= currentSize; i++ )
    {
        if (array[i].compareTo (x) >0)
            x = array[i];
    }

    return x ;
}
```

2.2.2) (3 pts) Quelle est la complexité asymptotique de votre algorithme? Justifiez brièvement.

L'algorithme parcourt la moitié des éléments du monceau. La complexité asymptotique est donc $O(n)$.

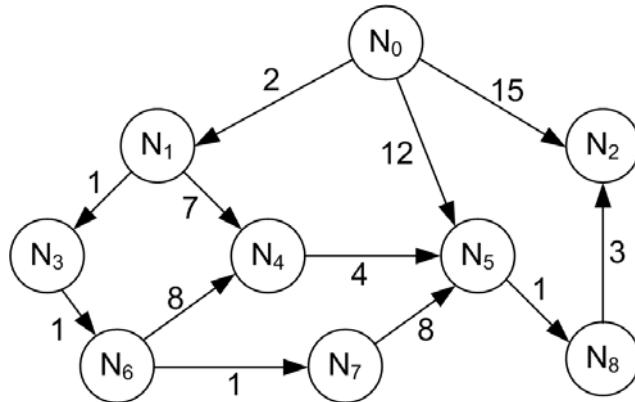
2.2.3) (4 pts) Peut-on améliorer la complexité asymptotique votre algorithme? Justifiez clairement votre réponse.

Indice : Pensez à l'algorithme du k-ième élément.

Le problème que nous avons ici est celui du k -ième élément appliqué sur les $n/2$ derniers éléments du monceau. Le k -ième élément étant résolu avec une complexité $O(n)$, on ne peut donc faire mieux que $O(n)$.

Question 3 : Graphes**(25 points)**

Soit le graphe suivant :

**3.1) (5 pts)** Répondez par vrai ou par faux à ces affirmations :**3.1.1) (1 pts)** Ce graphe n'est pas valué.

Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

3.1.2) (1 pts) Ce graphe n'admet pas d'ordre topologique.

Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

3.1.3) (1 pts) Ce graphe est dirigé.

Vrai	<input checked="" type="checkbox"/>
Faux	<input type="checkbox"/>

3.1.4) (1 pts) Ce graphe est dense.

Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

3.1.5) (1 pts) Ce graphe est cyclique.

Vrai	<input type="checkbox"/>
Faux	<input checked="" type="checkbox"/>

3.2.1) (7 pts) Exécutez l'algorithme de Dijkstra utilisant une file de priorité pour trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de N_0 .

Nœud	Connu	Dist min.	Parent	File de priorité
N_0	✓	0,	-	
N_1	✓	$\infty, 2$	N_0	
N_2	✓	$\infty, 15$	N_0	
N_3	✓	$\infty, 3$	N_1	
N_4	✓	$\infty, 9$	N_1	
N_5	✓	$\infty, 12$	N_0	
N_6	✓	$\infty, 4$	N_3	
N_7	✓	$\infty, 5$	N_6	
N_8	✓	$\infty, 13$	N_5	

3.2.2) (3 pts) Détaillez chacun des chemins trouvés :

Nœud	Le plus court chemin	Distance parcourue
N_0	$N_0 \rightarrow N_0$	0
N_1	$N_0 \rightarrow N_1$	2
N_2	$N_0 \rightarrow N_2$	15
N_3	$N_0 \rightarrow N_1 \rightarrow N_3$	3
N_4	$N_0 \rightarrow N_1 \rightarrow N_4$	9
N_5	$N_0 \rightarrow N_5$	12
N_6	$N_0 \rightarrow N_1 \rightarrow N_3 \rightarrow N_6$	4
N_7	$N_0 \rightarrow N_1 \rightarrow N_3 \rightarrow N_6 \rightarrow N_7$	5
N_8	$N_0 \rightarrow N_5 \rightarrow N_8$	13

3.3) (10 pts) Numérotez de à 1 à 9 les nœuds N_0 à N_8 du graphe de sorte que, s'il existe un chemin du nœud i au nœud j , alors $i \leq j$.

Noeud	1	2	3	4	5	6	7	8	9
N_0	0	0	0	0	0	0	0	0	0
N_1	1	0	0	0	0	0	0	0	0
N_2	2	1	1	1	1	1	1	1	0
N_3	1	1	0	0	0	0	0	0	0
N_4	2	2	1	1	0	0	0	0	0
N_5	3	2	2	2	2	1	0	0	0
N_6	1	1	1	0	0	0	0	0	0
N_7	1	1	1	1	0	0	0	0	0
N_8	1	1	1	1	1	1	1	0	0
Entrée	N_0	N_1	N_3	N_6	N_4, N_7	-	N_5	N_8	N_2
Sortie	N_0	N_1	N_3	N_6	N_4	N_7	N_5	N_8	N_2

Question 4 : Rabin-Karp**(15 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. Supposons que les chaînes de caractères soient constituées des chiffres 0 à 9 auxquels on associe la valeur numérale de ces derniers (le chiffre 9 vaut 9).

- 4.1) (9 pts)** Quel est le nombre de faux-positifs que produit l'exécution de l'algorithme Rabin-Karp sur la chaîne de caractères 32111125881 si l'on recherche le patron 111 en travaillant modulo q=7 ? Donnez le détail de vos calculs.

321 mod 7 = 6, ✓ → faux positif ; 211 mod 7 = 1, ✗;
111 mod 7 = 6, ✓ → vrai positif ; 111 mod 7 = 6, ✓ → vrai positif ;
112 mod 7 = 0, ✗ ; 125 mod 7 = 6, ✓ → faux positif ;
258 mod 7 = 6, ✓ → faux positif ; 588 mod 7 = 0, ✗ ;
881 mod 7 = 6, ✓ → faux positif ;

Il y a donc quatre (4) faux positifs.

- 4.2) (3 pts)** Donnez la valeur des sous-séquences correspondant aux faux positifs que vous avez trouvés dans 4.1).

321, 125, 258, 881

- 4.3) (3 pts)** Rabin-Karp produit parfois des faux-positifs (candidats retenus ne remplissant pas le critère de correspondance). Expliquez pourquoi il ne génère pas de faux-négatifs (candidats rejétés remplissant le critère de correspondance).

Si un candidat remplit le critère de correspondance, la valeur qui lui sera associée est forcément celle recherché et il ne sera pas rejeté. Ainsi, il ne peut pas y avoir de faux négatifs.

Question 5 : Programmation dynamique**(15 points)**

On désire trouver le parenthésage idéal pour multiplier les matrices A_1 à A_6 permettant de minimiser le nombre de multiplications (scalaires) à effectuer. Les matrices sont dimensionnées comme suit :

$$A_1 : 3 \times 6 ; A_2 : 6 \times 2; A_3 : 2 \times 8; A_4 : 8 \times 6; A_5 : 6 \times 1; A_6 : 1 \times 6$$

5.1) (6 pts) Considérez les tables **m** et **s** obtenue par l'exécution de l'algorithme dynamique vu en cours.

m	1	2	3	4	5	6
1	0	36	84	168	94	112
2		0	96	168	76	112
3			0	96	64	76
4				0	48	96
5					0	36
6						0

s	1	2	3	4	5	6
1		1	2	2	1	5
2			2	2	2	5
3				3	3	5
4					4	5
5						5
6						

5.1.1) (2 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_6 . Donnez son coût.

$$A_1(A_2(A_3(A_4A_5)))A_6 \text{ au coût de } 112$$

5.1.2) (2 pts) Donnez le parenthésage optimal pour multiplier A_2 à A_4 . Donnez son coût.

$$A_2(A_3A_4) \text{ au coût de } 168$$

5.1.3) (2 pts) Donnez le parenthésage optimal pour multiplier A_1 à A_4 . Donnez son coût.

$$(A_1A_2)(A_3A_4) \text{ au coût de } 168$$

5.2) (9 pts) Considérez les tables **m** et **s** modifiées du résultat précédent et complétez-les de sorte à trouver le coût optimal de la multiplication des matrices A_1 à A_7 , sachant que A_7 a une taille de 6×12 . Donnez ce parenthésage optimal et son coût.

m	1	2	3	4	5	6	7
1	0	36	84	168	94	112	202
2		0	96	168	76	112	220
3			0	96	64	76	160
4				0	48	96	122
5					0	36	144
6						0	72
7							0

s	1	2	3	4	5	6	7
1		1	2	2	1	5	5
2			2	2	2	5	5
3				3	3	5	5
4					4	5	6
5						5	5
6							6

Rappel : $m[i, j] = \min\{m[i, k] + m[k+1, j] + p_{i-1}p_kp_j\}$ pour $k = i$ à $j-1$, sachant que la matrice A_i a une dimension $p_{i-1} \times p_i$.

Annexe I

```

public class BinaryHeap <AnyType> extends AbstractCollection<AnyType>
{
    /**
     * Crée un BinaryHeap vide
     */
    public BinaryHeap( )
    {
        currentSize = 0;
        cmp = null;
        array = (AnyType[]) new Object[ DEFAULT_CAPACITY + 1 ];
    }

    /**
     * Effectue la comparaison de deux objets de type AnyType
     */
    private int compare( AnyType lhs, AnyType rhs )
    {
        return ((Comparable)lhs).compareTo( rhs );
    }

    /**
     * Insère x
     */
    public boolean add( AnyType x )
    {
        if( currentSize + 1 == array.length )
            doubleArray( );

        // Percolate up
        int hole = ++currentSize;
        array[ 0 ] = x;

        for( ; compare( x, array[ hole / 2 ] ) < 0; hole /= 2 )
            array[ hole ] = array[ hole / 2 ];
        array[ hole ] = x;

        return true;
    }

    /**
     * Donne le nombre d'éléments présents
     */
    public int size( )
    {
        return currentSize;
    }

    /**
     * Vide le monceau
     */
    public void clear( )
    {
        currentSize = 0;
    }

    /**
     * Retourn le plus petit element du monceau
     */
    public AnyType element( )
    {
        if( isEmpty( ) )
            throw new NoSuchElementException( );
        return array[ 1 ];
    }
}

```

```


    /**
     * Retire le plus petit élément
     */
    public AnyType remove( )
    {
        AnyType minItem = element( );
        array[ 1 ] = array[ currentSize-- ];
        percolateDown( 1 );

        return minItem;
    }

    /**
     * Construit un monceau à partir des éléments présents
     */
    private void buildHeap( )
    {
        for( int i = currentSize / 2; i > 0; i-- )
            percolateDown( i );
    }

    private static final int DEFAULT_CAPACITY = 100;
    private int currentSize;
    private AnyType [ ] array;

    /**
     * Méthode privée pour percoler vers le bas
     */
    private void percolateDown( int hole )
    {
        int child;
        AnyType tmp = array[ hole ];

        for( ; hole * 2 <= currentSize; hole = child )
        {
            child = hole * 2;
            if( child != currentSize &&
                compare( array[ child + 1 ], array[ child ] ) < 0 )
                child++;
            if( compare( array[ child ], tmp ) < 0 )
                array[ hole ] = array[ child ];
            else
                break;
        }
        array[ hole ] = tmp;
    }

    /**
     * Méthode privée pour doubler la taille de la mémoire
     */
    private void doubleArray( )
    {
        AnyType [ ] newArray;

        newArray = (AnyType [ ]) new Object[ array.length * 2 ];
        for( int i = 0; i < array.length; i++ )
            newArray[ i ] = array[ i ];
        array = newArray;
    }
}


```

**Corrigé
Examen final**

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>		
INF2010 – Structures de données et algorithmes		Tous	20091		
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>		
Ettore Merlo, responsable – Tarek Ould Bachir, chargé		A-533	5758 - 5193		
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>		
Jeudi	25 avril 2009	2h30	9h30 – 12h00		
<i>Documentation</i>	<i>Calculatrice</i>				
<input type="checkbox"/> Toute <input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Voir directives particulières	<input type="checkbox"/> Aucune <input type="checkbox"/> Programmable <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.			
<i>Directives particulières</i>					

Bonne chance à tous!

Important	Cet examen contient 5 questions sur un total de 18 pages (excluant cette page)
	La pondération de cet examen est de 40 %
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Tri par monceau (20 points)

a) On désire étudier le comportement du tri par monceau.

a.1) (3 pts) Donner la complexité algorithmique du tri par monceau (fonction du nombre d'éléments n) pour chacun des cas énumérés ci-dessous :

i) pire cas : **$O(n \log(n))$**

ii) cas moyen : **$O(n \log(n))$**

iii) meilleur cas : **$O(n \log(n))$**

a.2) (3 pts) Comparer les complexités du tri par monceau données dans a.1) à celles du tri par insertion et celle de QuickSort pour chacun des cas énumérés ci-dessous :

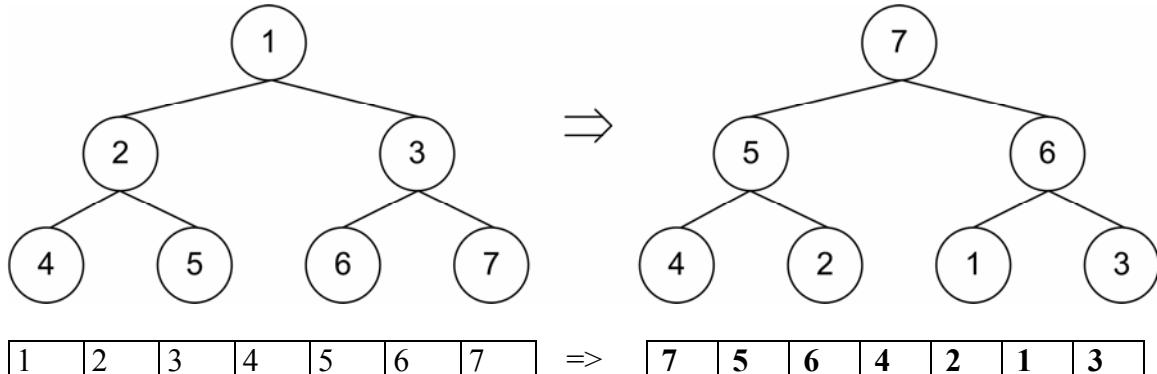
i) pire cas : **QuickSort : $O(n^2)$, Tri par insertion $O(n^2)$: Le tri par monceau est meilleur en pire cas.**

ii) cas moyen : **QuickSort : $O(n \log n)$, Tri par insertion $O(n^2)$: Le tri par monceau est aussi bon que QuickSort et meilleur que le Tri par insertion.**

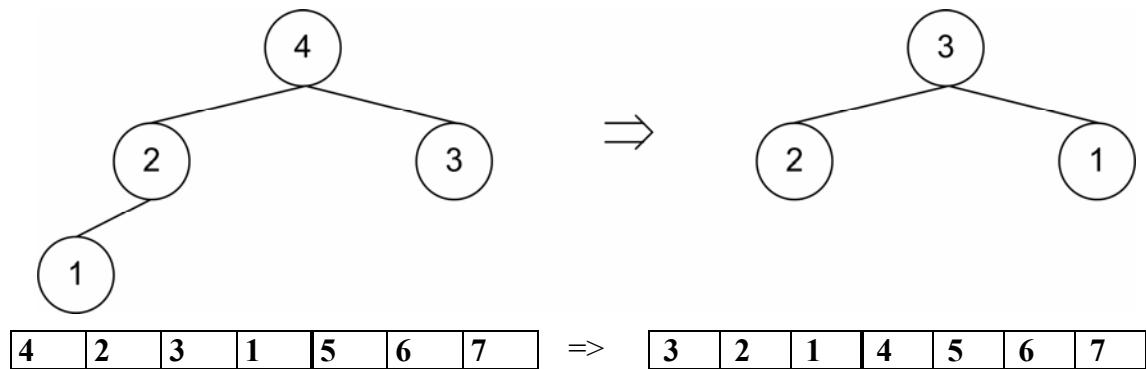
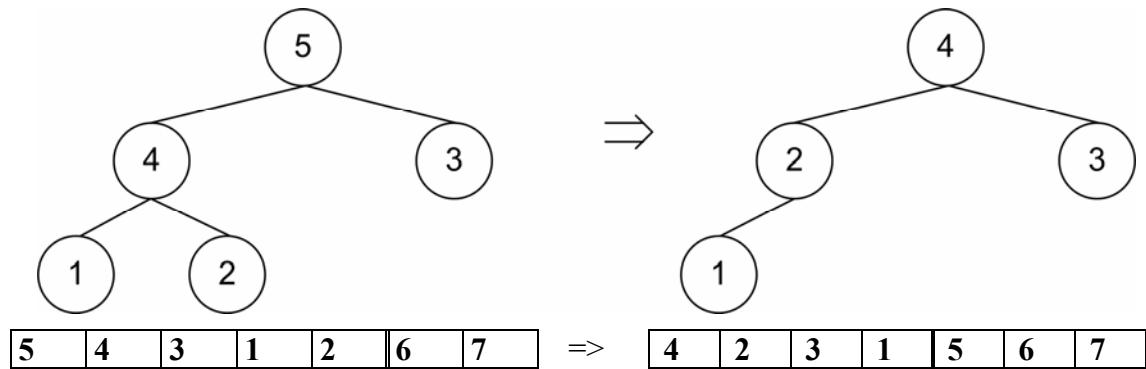
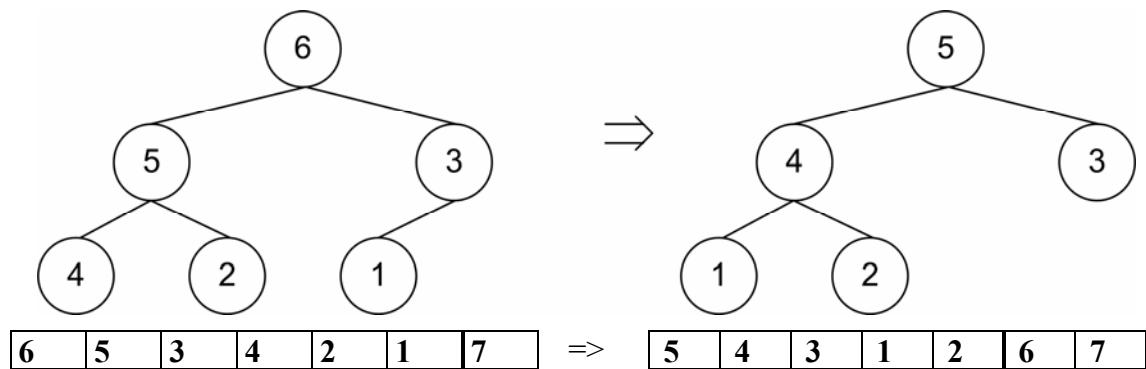
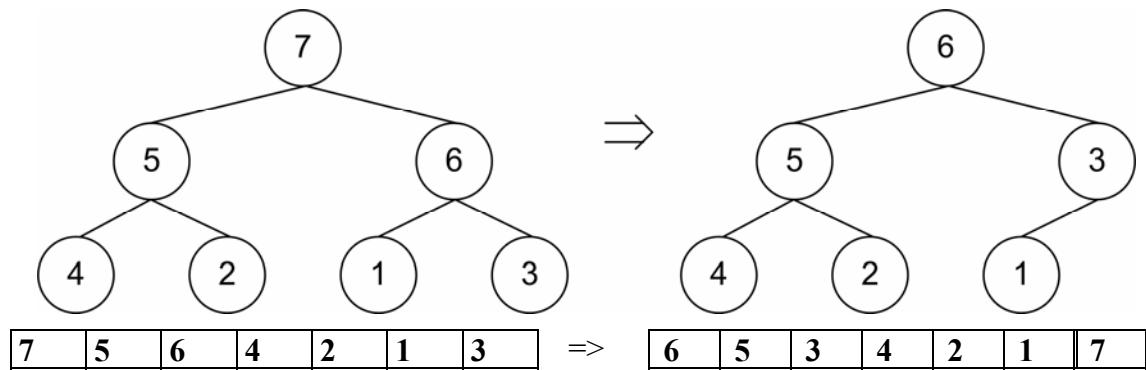
iii) meilleur cas : **QuickSort : $O(n \log n)$, Tri par insertion $O(n)$: Le tri par monceau est aussi bon que QuickSort mais moins bon que le Tri par insertion.**

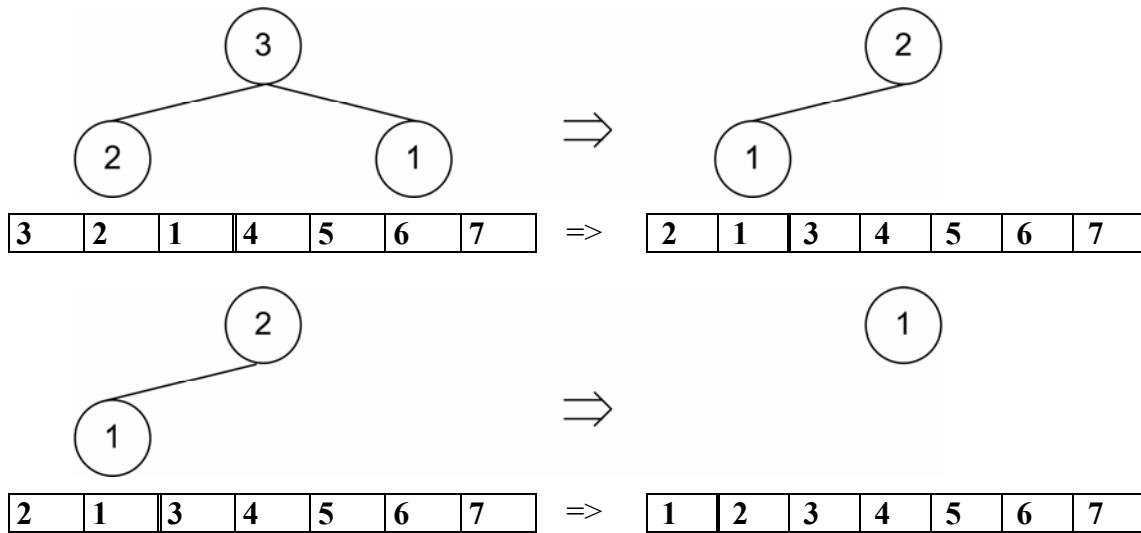
a.3) En partant du vecteur préalablement trié suivant $V = \{1, 2, 3, 4, 5, 6, 7\}$, on vous demande de :

i) (5 pts) Construire un monceau max (indiquer l'état du vecteur après l'opération);



ii) (6 pts) En partant du monceau max obtenu au point i), illustrer ci-dessous chacune des étapes du tri par monceau (indiquer l'étant du vecteur avant et après l'opération);





iii) (3 pts) Commenter le comportement de l'algorithme. Y a-t-il place à amélioration du tri par monceau selon vous ?

Tout comme le Merge Sort, Le tri par monceau ne fait aucune distinction entre les différents cas de figure d'entrée et maintient une complexité constante de $O(n \log(n))$. Ainsi, si l'entrée est déjà triée ou partiellement trié, le tri par monceau aura tendance à réordonner le vecteur pour en faire un monceau sans exploiter les propriétés du vecteur d'entrée.

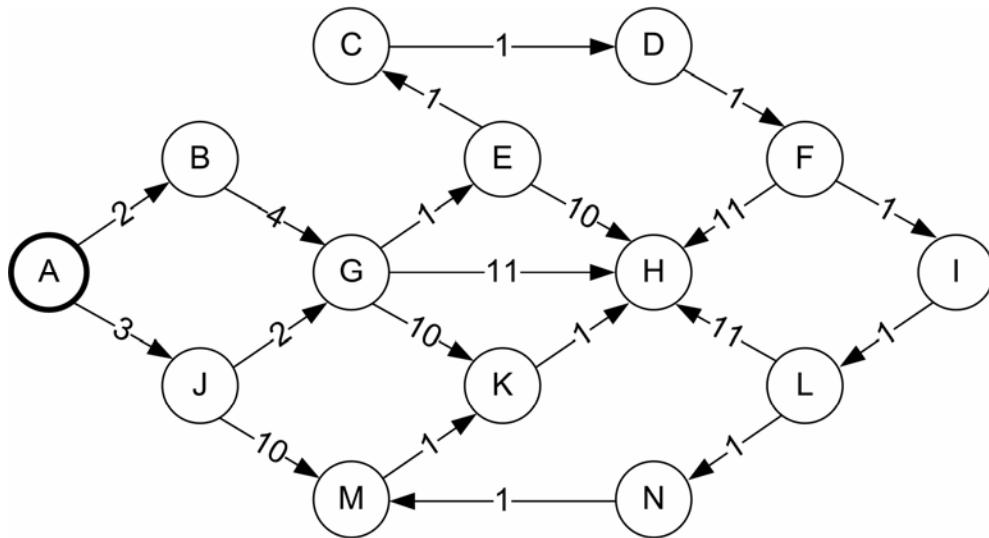
Une amélioration possible serait de vérifier si le vecteur est déjà trié et de le retourner tel quel, de sorte à avoir une complexité en $O(n)$ en meilleure cas (comme le Tri par insertion). Étant donnée la faible probabilité d'un tel cas, cette amélioration serait inutile.

Une autre amélioration serait d'utiliser un monceau Min et de rajouter une contrainte sur sa construction de sorte que le vecteur soit trié (fils de droite plus grand que fils de gauche). Mais cette amélioration serait difficile à implémenter.

Question 2 : Plus court chemin d'un graphe acyclique (20 points)

a) Tri topologique

a.I) (7 pts) Pour le graphe dirigé suivant, numérotez les nœuds de A à N de sorte que, s'il existe un chemin du nœud i au nœud j, alors $i < j$.



Noeud	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	1	0	0	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	0	0	0	0	0	0	0	0
E	1	1	1	1	0	0	0	0	0	0	0	0	0	0
F	1	1	1	1	1	1	0	0	0	0	0	0	0	0
G	2	2	1	0	0	0	0	0	0	0	0	0	0	0
H	5	5	5	5	4	3	3	3	2	2	1	1	1	0
I	1	1	1	1	1	1	1	1	0	0	0	0	0	0
J	1	0	0	0	0	0	0	0	0	0	0	0	0	0
K	2	2	2	2	1	1	1	1	1	1	1	1	0	0
L	1	1	1	1	1	1	1	1	1	0	0	0	0	0
M	2	2	2	2	1	1	1	1	1	1	1	0	0	0
N	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Entrée	A	B,J		G	E	C	D	F	I	L	N	M	K	H
Sortie	A	B	J	G	E	C	D	F	I	L	N	M	K	H

a.2) (3 pts) Peut-on dire de ce graphe qu'il est acyclique? Pourquoi?

Oui puisque la numération (le tri topologique) a été rendue possible.

b) Nous voulons trouver les plus courts chemins depuis le nœud A jusqu'à l'ensemble des nœuds B à N en appliquant l'algorithme de Dijkstra.

b.1) (6 pts) Continuez l'exécution de l'algorithme de Dijkstra utilisant une file de priorité (ici est partiellement réalisé), pour trouvez la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de A.

Nœud	Connu	Dist min.	Parent	File de priorité
A	✓	0		
B	✓	∞, 2	A	
C	✓	∞, 7	E	
D	✓	∞, 8	C	
E	✓	∞, 6	G	
F	✓	∞, 9	D	
G	✓	∞, 6, 5	B	
H	✓	∞, 16, 15	G, K	
I	✓	∞, 10	F	
J	✓	∞, 3	A	
K	✓	∞, 15, 14	G, M	
L	✓	∞, 11	I	
M	✓	∞, 13	J	
N	✓	∞, 12	L	

b.2) (2 pnt) Détaillez chacun des chemins les plus courts trouvés :

Nœud	Le plus court chemin	Distance parcourue
B	A → B	2
C	A → B → G → E → C	7
D	A → B → G → E → C → D	8
E	A → B → G → E	6
F	A → B → G → E → C → D → F	9
G	A → B → G	5
H	A → J → M → K → H	15
I	A → B → G → E → C → D → F → I	10
J	A → J	3
K	A → J → M → K	14
L	A → B → G → E → C → D → F → I → L	11
M	A → J → M	13
N	A → B → G → E → C → D → F → I → L → N	12

b.3) (2 pts) Comparer le chemin A → J → G → E → C → D → F → I → L → N → M avec le chemin allant de A à M que vous avez trouvé pour discuter le comportement de l'algorithme de Dijkstra.

Les chemin le plus court allant de A à M, trouvé grâce à l'algorithme, passe uniquement par J et a une longueur de 13. Le chemin A → J → G → E → C → D → F → I → L → N → M a également une longueur de 13. La raison pour laquelle Dijkstra a choisi ce chemin et non l'autre dépend entièrement de l'ordre d'entrée dans la file de priorité.

Question 3 : Rabin-Karp**(20 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un nombre qu'il faut pré-calculer et on compare toutes les valeurs des sous-séquences du texte à la valeur pré-calculée. Les différentes implémentations (**code java**) discutées dans cette question vous sont données à l'**annexe I**.

Dans ces implémentations de Rabin-Karp, le texte et le patron ne contiennent que les caractères alphabétiques ‘a’-‘z’, ‘A’-‘Z’ et l'espace ‘ ’ dans l'encodage ASCII. Les valeurs respectives de ces caractères sont données ci-dessous :

Caractère	Val. ASCII	Caractère	Val. ASCII
‘A’	65	‘a’	97
‘B’	66	‘b’	98
...		...	
‘Y’	89	‘y’	121
‘Z’	90	‘z’	122
espace	32	-	-

On vous propose la méthode de calcul suivante :

Polynôme modulaire dans la base $d = 10$ modulo 11

Par exemple, $P = \text{‘Anna’}$: Code : 65, 110, 110, 97

$$\begin{aligned}
 p &= (((((65\%11) * 10 + 110) \% 11) * 10 + 110) \% 11) * 10 + 97 \% 11 \\
 &= (((((10) * 10 + 110) \% 11) * 10 + 110) \% 11) * 10 + 97 \% 11 \\
 &= (((((210) \% 11) * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((1 * 10 + 110) \% 11) * 10 + 97) \% 11 \\
 &= (((120) \% 11) * 10 + 97) \% 11 \\
 &= ((10) * 10 + 97) \% 11 \\
 &= (197) \% 11 \\
 &= 10
 \end{aligned}$$

b) **(4 pnt)** Donnez la valeur de p pour le patron

$P = \text{‘merlo’}$: Codes ASCII : 109, 101, 114, 108, 111

$$\begin{aligned}
 p &= (109 * 10\ 000 + 101 * 1\ 000 + 114 * 100 + 108 * 10 + 111) \% 11 \\
 &= 1\ 203\ 591 \% 11 \\
 &= 4
 \end{aligned}$$

c) (16 pnt) Retrouvez tous les décalages donnant la présence de P dans le texte

T= « Un amerloque se divertit »

Codes ASCII :

U	n		a	m	e	r	l	o	q	u	e	s	e	d	i	v	e	r	t	i	t		
85	110	32	97	109	101	114	108	111	113	117	101	32	115	101	32	100	105	118	101	114	116	105	116

Décalage (s)	Sous-chaîne	Valeur du polynôme	Égalité?	Faux positif?	Correspondance?
0	‘Un am’	8			
1	‘n ame’	2			
2	‘ amer’	2			
3	‘amerl’	6			
4	‘merlo’	4	✓		✓
5	‘erloq’	9			
6	‘rloqu’	0			
7	‘loque’	6			
8	‘oque ’	2			
9	‘que s’	4	✓	✓	
10	‘ue se’	1			
11	‘e se ’	5			
12	‘ se d’	9			
13	‘se di’	7			
14	‘e div’	6			
15	‘ dive’	9			
16	‘diver’	5			
17	‘ivert’	2			
18	‘verti’	10			
19	‘ertit’	4	✓	✓	

Faux positifs trouvés :

Aux décalages 9 et 19

Décalages retournés :

Question 4 : PLSC**(20 points)**

Partant de la table de *PLSC* pour les entrées ACRYLONITRILE et ANTICLERICAL suivante, donnant ANTILE pour solution au problème, on propose de :

	A	C	R	Y	L	O	N	I	T	R	I	L	E
0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1
N	0	↑ 1	↑ 1	↑ 1	↑ 1	↑ 1	↑ 2	↑ 2	↑ 2	↑ 2	↑ 2	↑ 2	↑ 2
T	0	↑ 1	↑ 1	↑ 1	↑ 1	↑ 1	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3
I	0	↑ 1	↑ 1	↑ 1	↑ 1	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4
C	0	↑ 1	← 2	← 2	← 2	← 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4
L	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	← 3	↑ 3	↑ 3	↑ 3	↑ 4	← 5	↑ 5
E	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5	↑ 6
R	0	↑ 1	↑ 2	← 3	← 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5	↑ 6
I	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	← 4	↑ 4	↑ 5	↑ 5	↑ 6
C	0	↑ 1	← 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 5	↑ 6
A	0	← 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 5	↑ 6
L	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 4	← 4	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6	↑ 6

a) Évaluer la *PLSC* pour les entrées NITRILE et ANTICLERICAL en tronquant la première table et de récupérer le résultat partiel NTILE :

	N	I	T	R	I	L	E
0	0	0	0	0	0	0	0
A	← 0	← 1	← 1	← 1	← 1	← 1	← 1
N	↑ 0	← 2	← 2	← 2	← 2	← 2	← 2
T	↑ 0	↑ 2	↑ 2	← 3	← 3	← 3	← 3
I	↑ 0	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4
C	← 0	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4
L	← 0	← 3	↑ 3	↑ 3	↑ 3	↑ 4	← 5
E	↑ 0	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 6
R	↑ 0	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5	↑ 6
I	↑ 0	↑ 3	← 4	← 4	↑ 4	↑ 5	↑ 6
C	↑ 0	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
A	↑ 0	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
L	← 0	← 4	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6

a.1) (2 pts) Est-ce que NTILE est une PLSC des entrées NITRILE et ANTICLERICAL ?

Oui car la PLSC des entrées NITRILE et ANTICLERICAL a une longue de 5.

a.2) (2 pts) Discuter brièvement la validité du procédé de troncature dans ce cas de figure.

En aucun cas la troncature à l'avant des mots n'est une technique autorisée. Dans ce cas précis, elle fonctionne par un heureux hasard.

a.3) (6 pts) Proposer une autre PLSC des entrées NITRILE et ANTICLERICAL si elle existe (vous pouvez vous servir de la table donnée à l'annexe II) :

NTRIL est une autre PLSC des entrées NITRILE et ANTICLERICAL.

b) Évaluer la PLSC pour les entrées ACRYLONITRILE et CLERICAL en tronquant la première table et de récupérer le résultat partiel LE :

	A	C	R	Y	L	O	N	I	T	R	I	L	E
0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	↑ 1	↖ 2	← 2	← 2	← 2	← 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4
L	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↖ 5
E	0	↑ 1	↑ 2	↑ 2	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 5
R	0	↑ 1	↑ 2	↖ 3	← 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 6
I	0	↑ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
C	0	↑ 1	↖ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
A	0	↖ 1	↑ 2	↑ 3	↑ 3	↑ 3	↑ 3	↑ 3	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6
L	0	↑ 1	↑ 2	↑ 3	↑ 3	↖ 4	← 4	← 4	↑ 4	↑ 4	↑ 4	↑ 5	↑ 6

b.1) (2 pts) Est-ce que LE est une PLSC des entrées ACRYLONITRILE et CLERICAL ?

Non car il existe une PLSC de longueur 5.

b.2) (2 pts) Discuter brièvement la validité du procédé de troncature dans ce cas de figure.

La technique de troncature à l'avant n'est pas acceptable. Ceci est un bon exemple d'erreur potentielle qu'elle peut engendrer.

b.3) (6 pts) Proposer une autre PLSC des entrées ACRYLONITRILE et CLERICAL si elle existe (vous pouvez vous servir de la table donnée à l'annexe III) :

ALRIL est une PLSC valide.

Question 5 : Ensemble disjoints**(20 points)**

Partant du théorème suivant :

Tout ensemble S_n , défini comme l'ensemble des n premiers carrés entiers, admet une partition parfaite de quatre (4) ensemble disjoints (notés $S_{n,A}$, $S_{n,B}$, $S_{n,C}$, $S_{n,D}$) si et seulement si $n \geq 15$ et que $n \bmod 8 = 0$ ou $n \bmod 8 = 7$; cette partition est telle que chacune des sommes des éléments de ces ensembles disjoints sont égales.

Exemple :

Pour $n=15$, $S_{15}=\{1^2, 2^2, 3^2, 4^2, 5^2, 6^2, 7^2, 8^2, 9^2, 10^2, 11^2, 12^2, 13^2, 14^2, 15^2\}$; nous avons une partition parfaite (elle est aussi unique) :

$$S_{15,A} = \{1^2, 7^2, 8^2, 14^2\}$$

$$S_{15,B} = \{2^2, 9^2, 15^2\}$$

$$S_{15,C} = \{3^2, 6^2, 11^2, 12^2\}$$

$$S_{15,D} = \{4^2, 5^2, 10^2, 13^2\}$$

Où la somme des éléments de chaque ensemble disjoint est 310.

a) (2 pts) Peut-on trouver une partition de 4 ensembles disjoints à somme égale pour les ensembles S_n suivants :

Sn	Partition existe	Partition n'existe pas
S_{15}	✓	
S_{16}	✓	
S_{23}	✓	
S_{27}		✓
S_{33}		✓
S_{48}	✓	

Justifiez vos réponses :

Tous les n proposés sont supérieurs à 15. Aussi, nous avons :

$$23 \bmod 8 = 7 \checkmark$$

$$27 \bmod 8 = 3 X$$

$$33 \bmod 8 = 1 X$$

$$48 \bmod 8 = 0 \checkmark$$

b) Sachant que :

L'ensemble des 32 carrés consécutifs $C_i = S_{i+31} - S_{i-1} = \{i^2, (i+1)^2, \dots, (i+31)^2\}$ ($i \geq 1$) admet une partition parfaite unique :

$$C_{i,A} = \{i^2, (i+7)^2, (i+11)^2, (i+12)^2, (i+18)^2, (i+21)^2, (i+25)^2, (i+30)^2\}$$

$$C_{i,B} = \{(i+1)^2, (i+6)^2, (i+10)^2, (i+13)^2, (i+19)^2, (i+20)^2, (i+24)^2, (i+31)^2\}$$

$$C_{i,C} = \{(i+2)^2, (i+5)^2, (i+9)^2, (i+14)^2, (i+16)^2, (i+23)^2, (i+27)^2, (i+28)^2\}$$

$$C_{i,D} = \{(i+3)^2, (i+4)^2, (i+8)^2, (i+15)^2, (i+17)^2, (i+22)^2, (i+26)^2, (i+29)^2\}$$

Où la somme des éléments de chaque ensemble disjoint est $8i^2 + 248i + 2604$.

b.1) (2 pts) Proposer une partition parfaite pour S_{32} :

	Éléments							
$S_{32,A}$	1^2	8^2	12^2	13^2	19^2	22^2	26^2	31^2
$S_{32,B}$	2^2	7^2	11^2	14^2	20^2	21^2	25^2	32^2
$S_{32,C}$	3^2	6^2	10^2	15^2	17^2	24^2	28^2	29^2
$S_{32,D}$	4^2	5^2	9^2	16^2	18^2	23^2	27^2	30^2

b.2) (3 pts) Combien existe-t-il de partitions parfaites pour S_{47} ? Justifiez par un calcul.

Indice : $47=15+32$

Les partitions parfaites pour S_{47} sont obtenues en combinant les $S_{15,A}, S_{15,B}, S_{15,C}, S_{15,D}$ et $C_{16,A}, C_{16,B}, C_{16,C}, C_{16,D}$. Il existe donc $4 * 3 * 2 * 1$ combinaisons possibles, pour un total de 24.

b.3) (3 pts) En vous servant du raisonnement utilisé pour **b.2)**, proposer une partition parfaite pour S_{47} :

	Éléments											
$S_{47,A}$	1^2	7^2	8^2	14^2	16^2	23^2	27^2	28^2	34^2	37^2	41^2	46^2
$S_{47,B}$	2^2	9^2	15^2	17^2	22^2	26^2	29^2	35^2	36^2	40^2	47^2	
$S_{47,C}$	3^2	6^2	11^2	12^2	18^2	21^2	25^2	30^2	32^2	39^2	43^2	44^2
$S_{47,D}$	4^2	5^2	10^2	13^2	19^2	20^2	24^2	31^2	33^2	38^2	42^2	45^2

c) On vous demande d'exécuter la structure d'ensembles disjoints utilisant la compression du chemin et la métrique des rangs¹ pour générer la partition de S₃₂ (**b.I**).

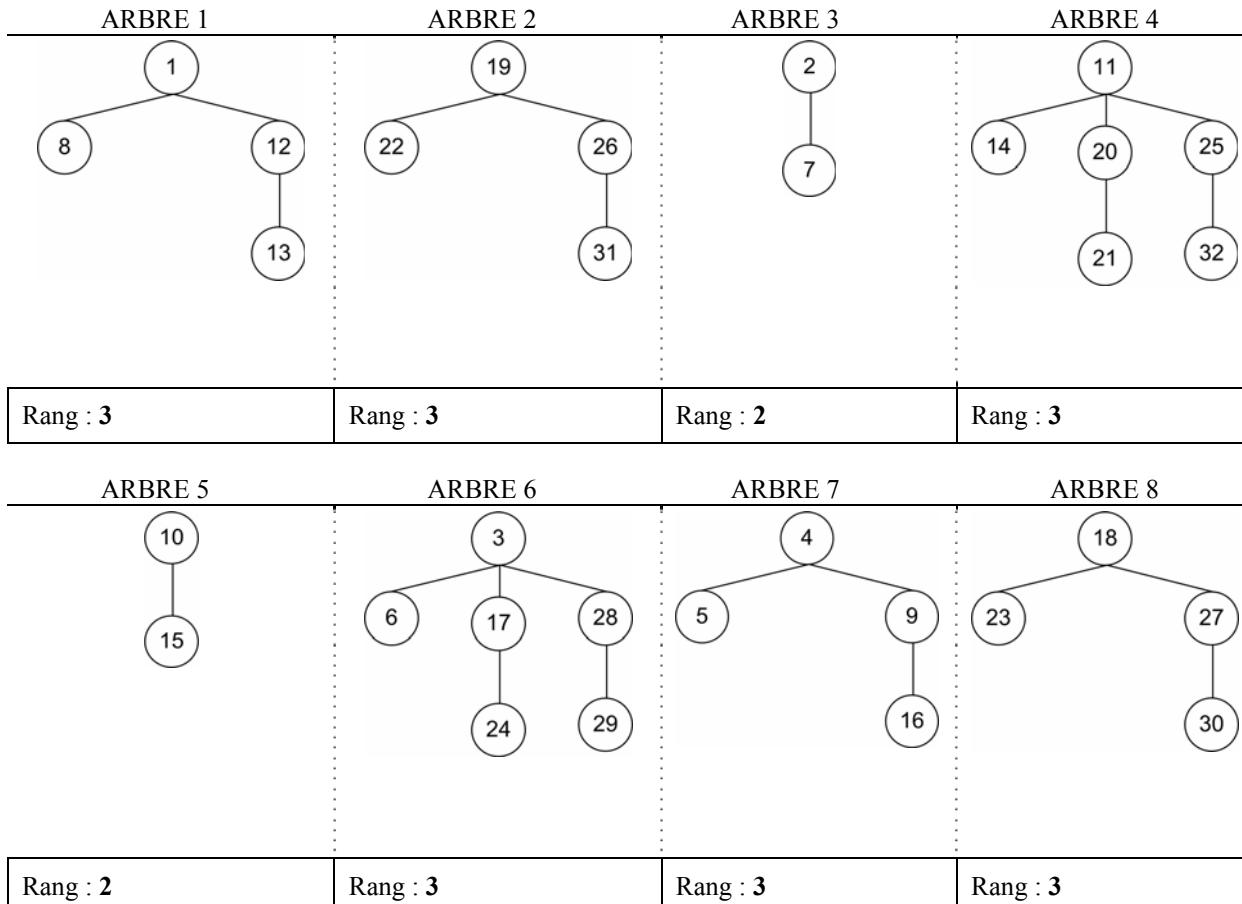
c.I) (4 pts) Dessiner la forêt d'arbres (indiquer le rang de chaque arbre) obtenue à la fin des appels suivants (pseudo-code java) :

```
// A la fin de cet appel, nous avons 32 arbres
for(int i=1; i ≤ 32; i++)
    Create(i2);

// A la fin de ces appels, nous avons 16 arbres
Union(12, 82); Union(122, 132); Union(192, 222); Union(262, 312);
Union(22, 72); Union(112, 142); Union(202, 212); Union(252, 322);
Union(32, 62); Union(102, 152); Union(172, 242); Union(282, 292);
Union(42, 52); Union(92, 162); Union(182, 232); Union(272, 302);

// A la fin de ces appels, nous avons 8 arbres
Union(12, 122); Union(192, 262); Union(112, 202); Union(202, 252);
Union(32, 172); Union(32, 282); Union(42, 92); Union(182, 272);
```

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i



¹ **Rappel :** i) Lors de l'union de deux arbres de rangs différents, l'arbre de rang inférieur devient l'enfant de l'arbre de rang supérieur. Le nouveau rang est le plus grand des deux rangs.

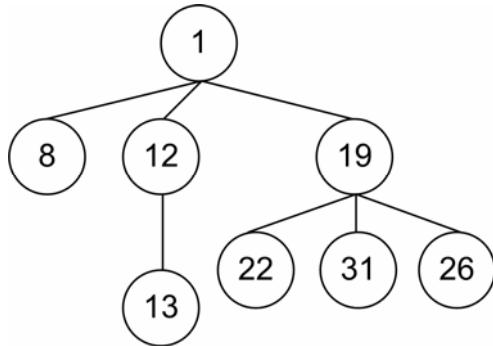
ii) La racine de l'arbre issu de l'union de deux arbres de même rang est la racine ayant la plus petite valeur parmi les deux arbres. Le nouveau rang est celui des deux arbres incrémenté de 1.

c.2) (4 pts) En partant des résultats de la question **c.1)**, dessiner la forêt d'arbres (indiquer le rang de chaque arbre) obtenue à la fin des appels :

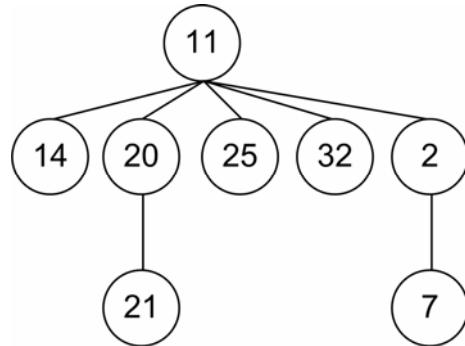
```
// A la fin de ces appels, nous avons 4 arbres
Union(12, 312); Union(72, 322); Union(152, 242); Union(42, 302);
```

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i

ARBRE 1



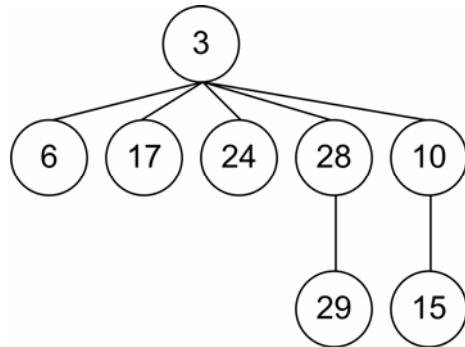
ARBRE 2



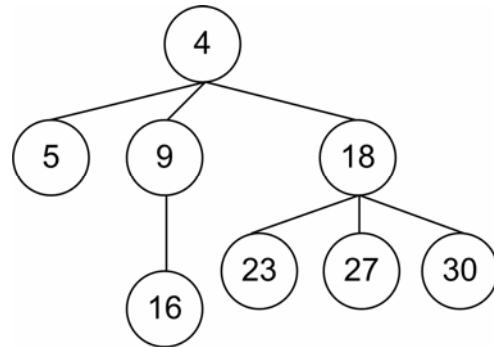
Rang : 4

Rang : 3

ARBRE 3



ARBRE 4



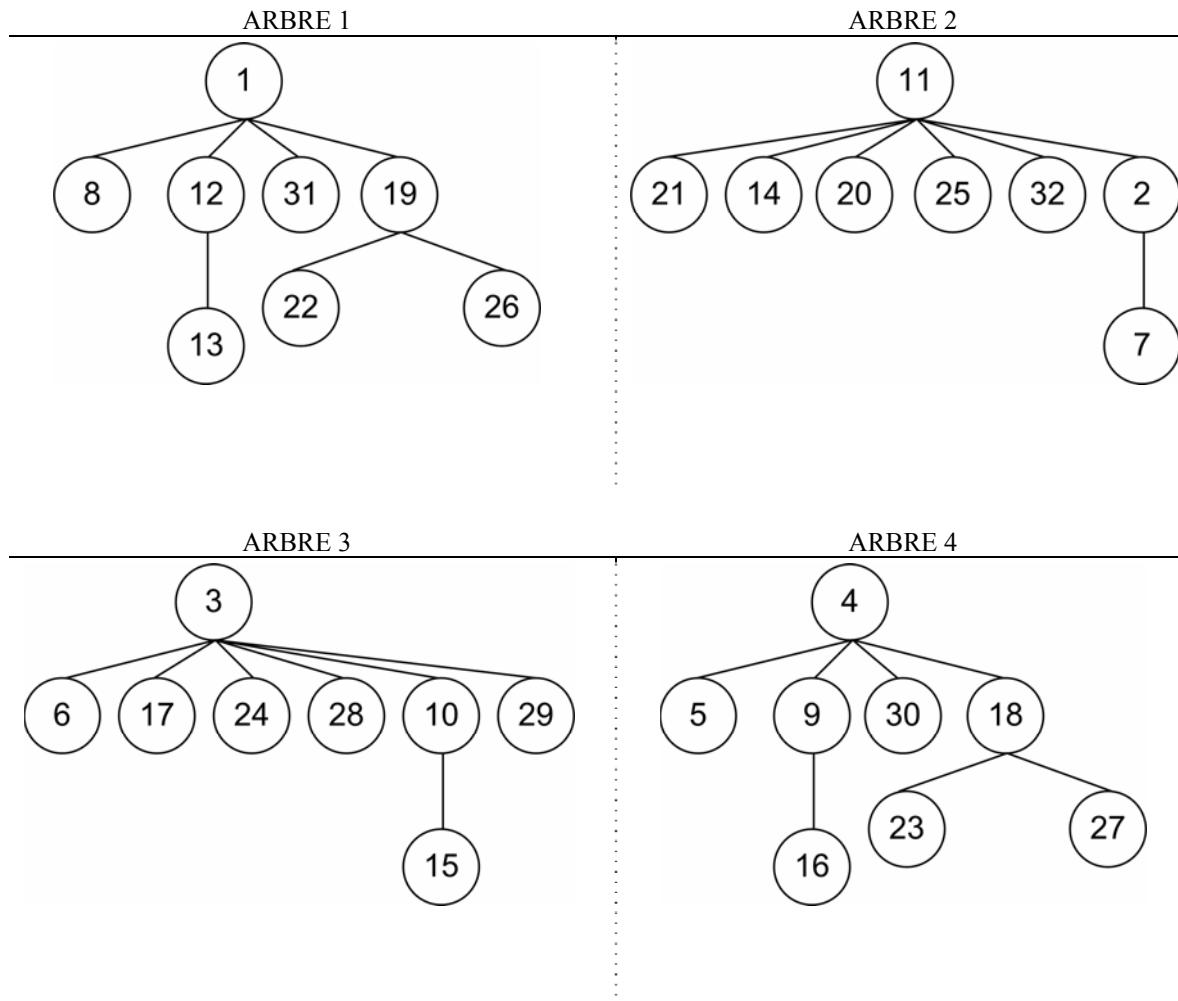
Rang : 3

Rang : 4

c.3) (2 pts) En partant des résultats de la question **c.2)**, dessiner la forêt d'arbres obtenue à la fin des appels :

```
// A la fin de ces appels, nous avons 4 arbres
Find(312) ; Find(212) ; Find(142) ; Find(292) , Find(302) ;
```

N.B.: Vous pouvez omettre le carré dans les nœuds et représenter le nœud i^2 par i



Annexe I

```

import java.util.ArrayList;

public class RabinKarp {

    public static ArrayList<Integer>
    RabinKarpFind(String Text, String Pattern)
    {
        ArrayList<Integer> decalages = new ArrayList<Integer>();

        if( Text.length() < Pattern.length() )
            return decalages;

        int p = ComputePatternValue( Pattern );

        for(int i=0; i <= Text.length() - Pattern.length(); i++)
        {
            int t = ComputePatternValue(Text.substring(i, i+Pattern.length()));

            if( t == p )
            {
                int j;
                for(j=0; j< Pattern.length(); j++)
                {
                    if( Pattern.charAt( j ) != Text.charAt( i + j ) )
                        break;
                }

                if(j == Pattern.length())
                {
                    decalages.add( i );
                    System.out.println("Correspondance à " + i);
                }
                else
                    System.out.println("Faux positif à " + i);
            }
        }

        return decalages;
    }

    public static int ComputePatternValue(String Pattern)
    {
        int p = 0;

        for(int i=0; i<Pattern.length(); i++)
        {
            p *=10;
            p += (int) Pattern.charAt( i );
            p %= 11;
        }

        return p;
    }

    public static void main(String[] args)
    {
        String Pattern = "merlo";
        int p = ComputePatternValue(Pattern);
        System.out.println( p );

        String Text = "Un amerloque se divertit";
        System.out.println( Text.length() );
        ArrayList<Integer> decalages = RabinKarpFind(Text, Pattern);

        for(int s : decalages )
            System.out.print( s + "\t" );
    }
}

```

Annexe II

a) PLSC de NITRILE et ANTICLERICAL

	N	I	T	R	I	L	E
N	0	0	0	0	0	0	0
A	0						
T							
I	0						
C	0						
L	0						
E	0						
R	0						
I	0						
C	0						
A	0						
L	0						

Annexe III

b) PLSC de ACRYLONITRILE et CLERICAL :

	A	C	R	Y	L	O	N	I	T	R	I	L	E
0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0												
L	0												
E	0												
R	0												
I	0												
C	0												
A	0												
L	0												

**Corrigé
examen final**

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 – Structures de données et algorithmes		Tous	20093
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
T. Lavoie et T. Ould Bachir (chargés de cours) Pr. Ettore Merlo (coordonateur)		B-415 (gr. 1) B-418 (gr. 2)	
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Lundi	07 décembre 2009	2h30	13h30-16h00
<i>Documentation</i>		<i>Calculatrice</i>	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input checked="" type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Toutes <input checked="" type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.
<i>Directives particulières</i>			
<ul style="list-style-type: none"> ☒ Un cahier supplémentaire vous sera remis. Servez-vous de ce cahier comme brouillon. Nous ne récupérerons pas le cahier supplémentaire à la fin de l'examen. 			
<i>Bonne chance à tous!</i>			
Important	Cet examen contient 8 questions sur un total de 13 pages (inclus cette page)		
	La pondération de cet examen est de 40 %		
Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux			
Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non			

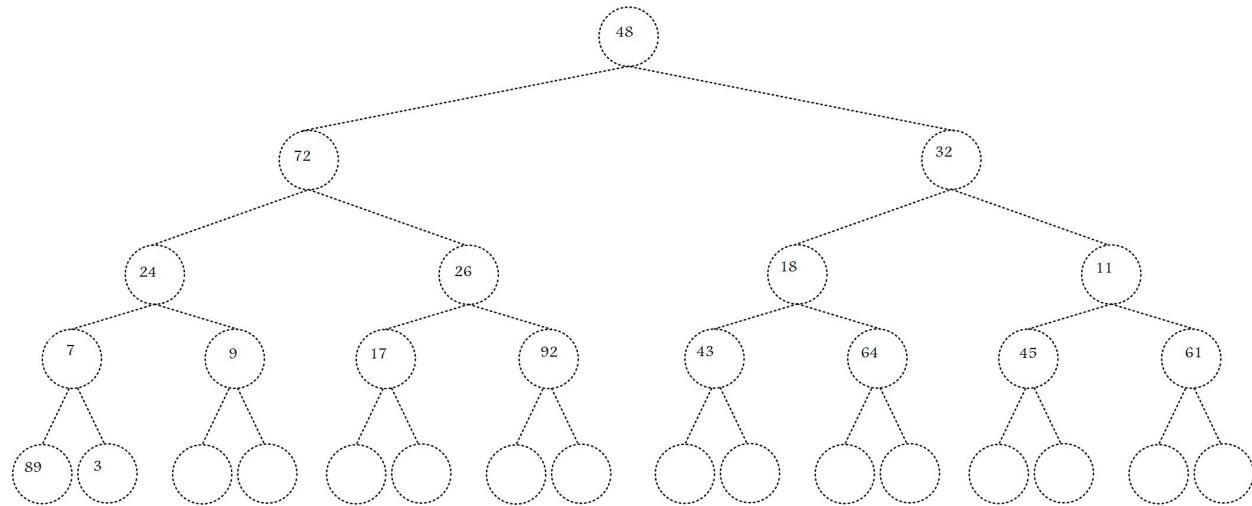
L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Monceaux**(10 points)**

1.1) (2 points) Considérez le tableau suivant qui représente un arbre binaire:

	48	72	32	24	26	18	11	7	9	17	92	43	64	45	61	89	3							
--	----	----	----	----	----	----	----	---	---	----	----	----	----	----	----	----	---	--	--	--	--	--	--	--

Indiquez l'emplacement de chacun des éléments dans l'arbre ci-dessous



1.2) (2 points) L'arbre de la question 1.1) est-il un monceau?

Non. On le voit par exemple au fait que 3 est l'enfant de 7.

1.3) (2 points) Donnez la formule pour calculer l'index des enfants droit et gauche du nœud à l'index i.

Enfant droit:

$$2i+1$$

Enfant gauche:

$$2i$$

1.4) (2 points) Si N est le nombre de nœuds dans un monceau, combien de nœuds faut-il vérifier au maximum pour trouver le nœud maximum dans un monceau MIN ?

Le maximum est forcément une feuille. Il est impossible de savoir quelle feuille contient le maximum. Il faut donc vérifier toutes les feuilles. Il y a au plus $(N+1)/2$ (division entière) feuilles. La réponse est donc $(N+1)/2$ (division entière).

1.5) (2 points) Quelle est la complexité asymptotique de la construction d'un monceau?

La construction d'un monceau s'effectue en $O(N)$.

Question 2 : Tri par monceau (15 points)

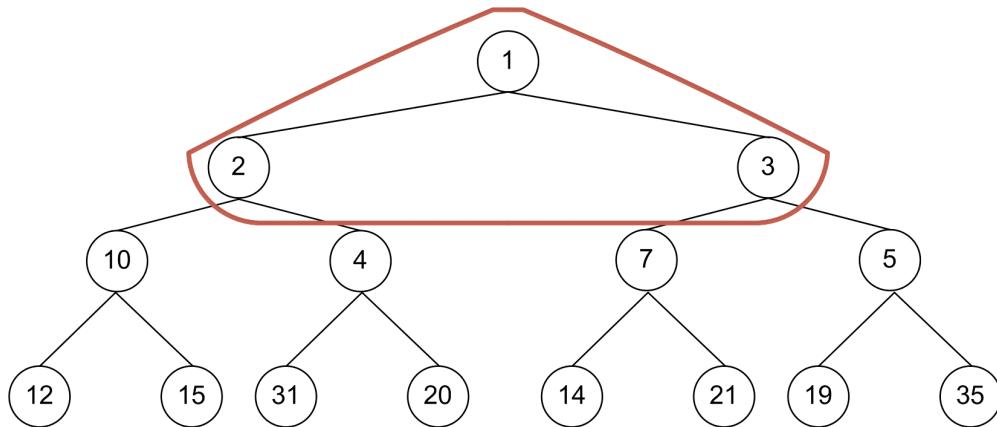
On désire trouver les trois plus petits éléments d'un vecteur d'entiers. On vous suggère, pour ce faire, d'utiliser un monceau MIN et de trouver les trois plus petits éléments au niveau de la racine et de ses deux enfants.

2.1) (3 points) Commentez cette stratégie. Est-elle justifiée ou non? Expliquez clairement votre réponse.

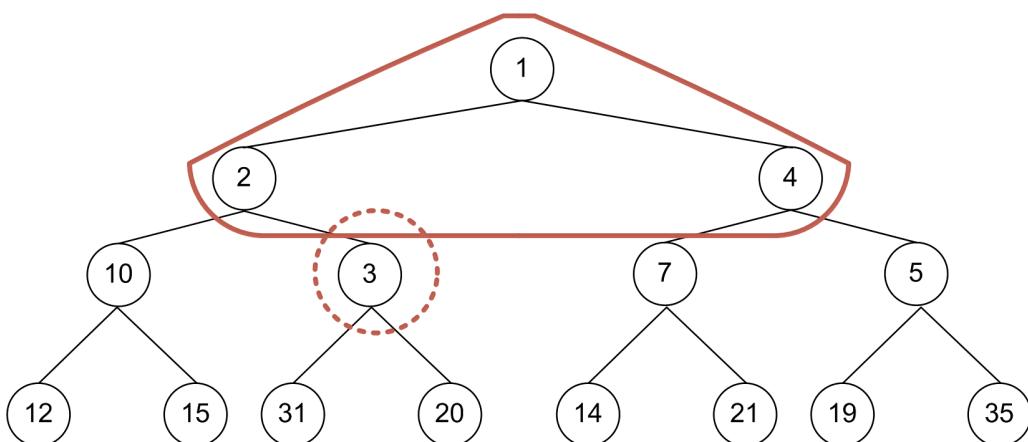
C'est une mauvaise stratégie car elle ne réussit pas à tous les coups.

2.2) Illustriez votre réponse en construisant un monceau min de chacun des vecteurs suivants :

2.2.1) (1.5 points) 15, 31, 21, 10, 20, 14, 5, 12, 2, 4, 1, 3, 7, 19, 35



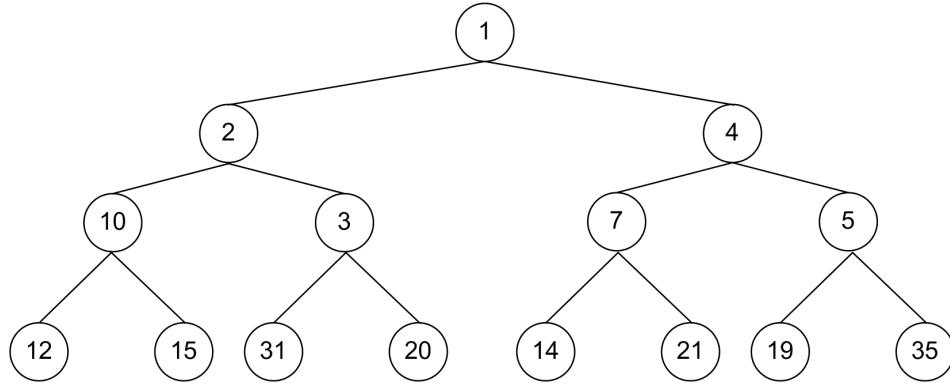
2.2.2) (1.5 points) 15, 31, 21, 10, 20, 14, 5, 12, 2, 3, 1, 4, 7, 19, 35



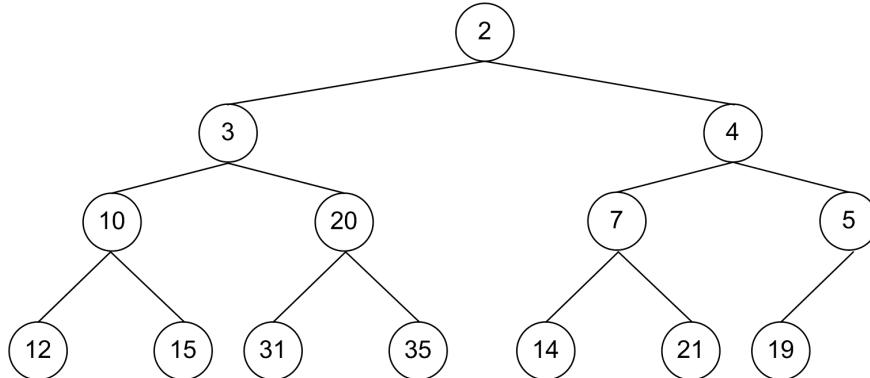
On voit donc bien que cette stratégie ne réussit pas toujours.

2.3) Exécutez l'algorithme de tri partiel vu en cours pour trouver les trois plus petites valeurs parmi les entiers 15, 31, 21, 10, 20, 14, 5, 12, 2, 3, 1, 4, 7, 19, 35 (ceux de 2.2.2). Pour mémoire, le principe du tri partiel consiste à construire un monceau MIN et à retirer la racine pour chaque nouvelle valeur.

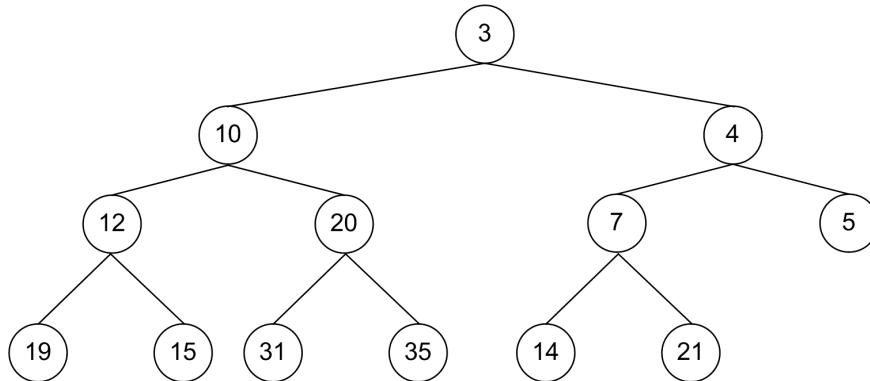
2.3.1) (3 points) Premier monceau du tri partiel, avant retrait de la racine :



2.3.2) (3 points) Second monceau du tri partiel, avant retrait de la racine :



2.3.3) (3 points) Troisième monceau du tri partiel, avant retrait de la racine :



Question 3 : Modélisation par graphes**(15 points)**

3.1) (5 points) Soit un labyrinthe composés d'un nombre N de salles. Chaque salle est de forme carrée et peut avoir jusqu'à 4 portes conduisant à une salle adjacente, soit au nord, au sud, à l'est ou à l'ouest. Proposez une façon de modéliser ce problème avec un graphe. Indiquez clairement ce que représente les nœuds et les arrêtes.

Chaque sommet du graphe représente une salle du labyrinthe. Les arrêtes représentent les portes et il y en a une entre chaque salles adjacentes non-séparées par un mur. Le graphe n'est pas orienté.

3.2) (5 points) Supposons que l'on vous donne un graphe $G(V, E)$ qui respecte votre modélisation de la question 3.1). Donnez, en notation asymptotique, le temps d'exécution le plus rapide qu'il est possible d'obtenir pour le problème consistant à trouver la sortie du labyrinthe à partir du point de départ.

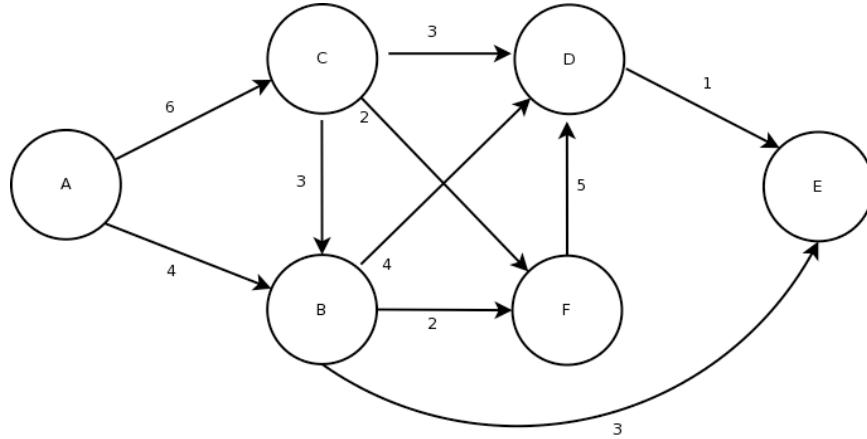
Ce problème est équivalent à trouver le plus court chemin entre deux sommets d'un graphe dans un graphe non-valué. Comme le graphe est épars ($|E| < 4 \times |V|$), l'algorithme pour résoudre ce problème a une complexité de $O(|E| + |V|)$.

3.3) (5 points) Supposons maintenant que l'on permette de détruire des murs pour trouver un chemin, mais la destruction entraîne une pénalité de $P > 0$ cases. Quelles modifications faut-il apporter à celle proposée en 3.1) ? Comment peut-on trouver le chemin le moins pénalisant dans cette version du problème du labyrinthe ?

Il suffit de transformer le graphe en un graphe valué où le coût entre deux nœuds est égal à la pénalité entre deux cases adjacentes (0 si il n'y a pas de murs). Le problème devient alors équivalent à trouver le plus court chemin dans un graphe valué; ce problème résoluble par l'algorithme de Dijkstra.

Question 4 : Algorithme de Dijkstra**(15 points)**

4.1) (10 points) Soit le graphe suivant:



Calculez à l'aide de l'algorithme de Dijkstra (n'utilisant pas de monceau) la distance minimale entre le point A et tous les autres points dans le graphe. Donnez les étapes de votre exécution de l'algorithme dans la table qui suit.

Nœud	Distance A	Distance B	Distance C	Distance D	Distance E	Distance F
-	0	inf	inf	inf	inf	inf
A	0	4	6	inf	inf	inf
B	0	4	6	8	7	6
C	0	4	6	8	7	6
F	0	4	6	8	7	6
E	0	4	6	8	7	6
D	0	4	6	8	7	6

4.2) (5 points) Quelle est la complexité en pire cas de l'algorithme de Dijkstra que vous venez d'exécuter ? Sachant que l'algorithme peut être amélioré, dites comment et donnez la nouvelle complexité ?

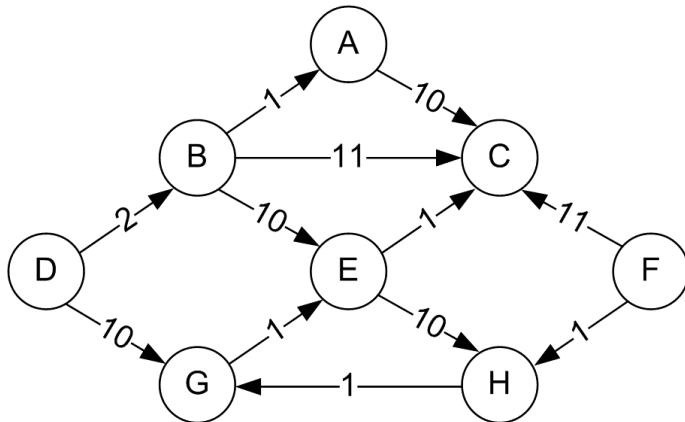
Pire cas: $O(|V|^2)$

L'algorithme peut être amélioré en utilisant un monceau. La complexité devient alors $O(|E|\log(V))$.

Question 5 : Ordre topologique**(10 points)**

Pour chacun des graphes dirigés aux arêtes pondérées suivants, dîtes si le graphe admet un ordre topologique et si oui proposez en un. Vous pouvez vous aider des grilles données à l'annexe I. Identifiez clairement cette annexe à votre nom et matricule et assurez-vous de la remettre avec votre copie (même si elle demeure vierge).

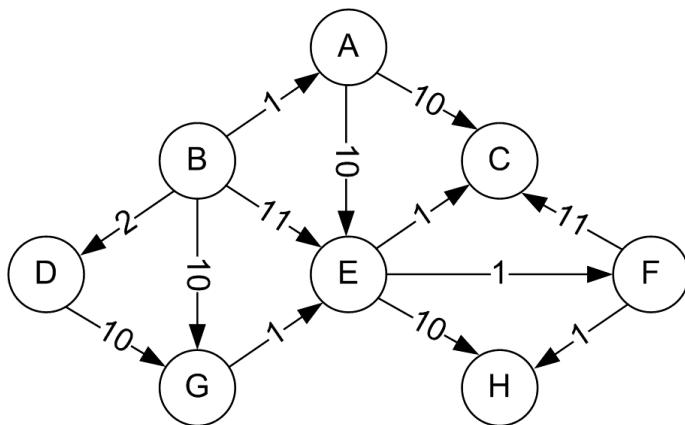
5.1) (5 points)



Ce graphe est cyclique et n'admet pas d'ordre topologique

A	B	C	D	E	F	G	H
✗	✗	✗	✗	✗	✗	✗	✗

5.2) (5 points)



A	B	C	D	E	F	G	H
2	1	7	3	5	6	4	8

Question 6 : Rabin-Karp**(15 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. Supposons que les chaînes de caractères soient constituées des chiffres 0 à 9 auxquels on associe la valeur numérale de ces derniers (le chiffre 9 vaut 9).

6.1) (2 points) Quel est le nombre de faux-positifs que produit l'exécution de l'algorithme Rabin-Karp sur la chaîne de caractères 96325874124896323495 si l'on recherche le patron 96 en travaillant modulo $q=11$?

Note : Vous n'êtes pas obligé de donner vos calculs intermédiaires. On vous conseille cependant de les garder pour les questions suivantes.

Il y a 4 faux-positifs au total.

6.2) (3 points) Donnez la valeur des sous-séquences correspondants aux faux positifs que vous avez trouvés dans 6.1.

63, 74, 41, 63

6.3) (2 points) Rabin-Karp produit parfois des faux-positifs (candidats retenus ne remplissant pas le critère de correspondance). Expliquez pourquoi il ne génère pas de faux-négatifs (candidats rejetés remplissant le critère de correspondance).

Si un candidat est valide, la valeur de hash qu'il produira est forcément la bonne et il est donc impossible de rejeter un candidat valide.

6.4) On veut rechercher deux patrons P_1 et P_2 de longueur m dans un texte T de longueur n , et ce en exécutant une variante de l'algorithme Rabin-Karp (appliquée à deux patrons de même longueur et non plus un).

6.4.1) (2 points) Quelle est la complexité (en pire cas) de cet algorithme ? Justifiez brièvement.

Le pire cas survient si chaque décalage signale un candidat potentiel et que les deux patrons P_1 et P_2 ont la même valeur de hash car il faudrait alors effectuer deux comparaisons sur m caractères chaque fois qu'un candidat est trouvé. Il en résulte alors que la complexité en pire cas est $O(2m(n-m+1)) = O(m(n-m+1))$.

6.4.2) (2 points) Combien de faux-positifs produit cet algorithme sur la chaîne de caractères de la question 6.1 (96325874124896323495) en recherchant les patrons 23 et 87 ?

Nous aurons 3 faux-positifs pour 23 et 2 faux-positifs pour 87, donc un total de 5 faux-positifs.

6.4.3) (2 points) Combien de faux-négatifs produit cet algorithme en recherchant les mêmes patrons ?

Aucun. Rabin-Karp ne produit pas de faux-négatifs.

6.4.4) (2 points) Quels sont les décalages retournés pour chacun de ces patrons ?

Pour 23, nous aurons le décalage 15.

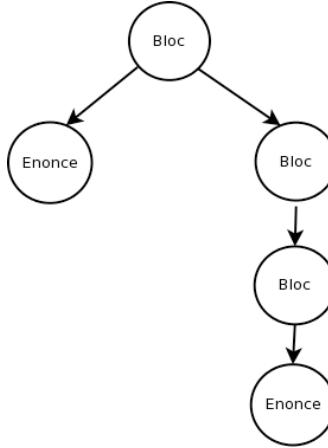
Pour 87, nous aurons le décalage 5.

Question 7 : Métriques dans les arbres de syntaxe abstraits (10 points)

Un AST est une représentation abstraite de la syntaxe du code d'une application. Pour simplifier le problème, on ne représente ici que les blocs et les énoncés dans le code. Les blocs peuvent contenir d'autres blocs ou des énoncés, mais les énoncés ne peuvent avoir aucun enfant. Par exemple, le code:

```
public int uneMethode() {
    int i = 0;
    while(i < 12) {
        if(i > 4) {
            System.out.println(i);
        }
    }
}
```

Serait représenté par l'AST suivant:



On désire connaître le nombre d'énoncés et de blocs contenus à l'intérieur de chacun des blocs de l'AST. Par exemple, le bloc racine (qui correspond au bloc `uneMethode`) contient 2 blocs et 2 énoncés. Proposez un algorithme en temps linéaire qui permet de calculer les métriques de chacun des blocs de l'AST. *Indice: Il faut parcourir tous les nœuds du graphe pour faire ce calcul.*

Il suffit de faire une fouille en profondeur et d'accumuler les métriques en les faisant remonter progressivement vers leur parent.

Question 8 : Programmation dynamique**(10 points)**

Les coefficients binomiaux ont une grande utilité en mathématique puisqu'on les retrouve par exemple dans le calcul des combinatoires ou le binôme de Newton. La combinatoire k de n ($k \leq n$) est donnée par :

$$\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}$$

Il est également possible de l'exprimer par une formule de récurrence :

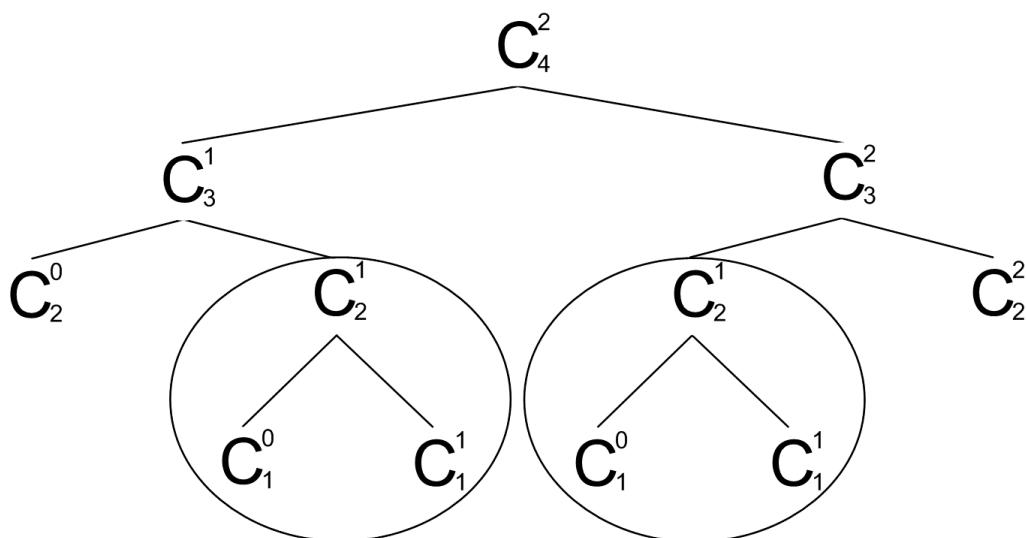
$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$$

où $C_0^0 = C_n^0 = C_n^n = 1$. Cela peut se traduire par l'implémentation suivante :

```
public static int Combinatoire(int n, int k)
{
    if(n <0 || k<0)
        return -1;

    if(n==k || k==0)
        return 1;
    else
        return Combinatoire(n-1, k) + Combinatoire(n-1, k-1);
}
```

8.1) (3 points) En vous appuyant sur la fonction donné ci-dessus, illustrez le fait que l'expression récursive des coefficients binomiaux présente un cas de recouvrement des sous-problèmes en vous basant sur les étapes de calcul de C_4^2 .



8.2) On vous propose l'implémentation d'un algorithme de programmation dynamique suivante pour le calcul des coefficients binomiaux :

```
public static int CombinatoireDynamique(int n, int k)
{
    if(n <0 || k<0)
        return -1;

    int[][] M = new int[ n+1 ][ k+1 ];

    for(int i=0; i<=n; i++)
    {
        for(int j=0; j <= (i<k ? i : k) ; j++)
        {
            if(i==j || j==0)
                M[ i ][ j ] = 1;
            else
                M[ i ][ j ] = M[ i-1 ][ j ] + M[ i-1 ][ j-1 ];
        }
    }

    return M[ n ][ k ];
}
```

8.2.1) (3 points) Détaillez les étapes d'exécution de cet algorithme pour le calcul de C_4^2

1				
1	1			
1	2	1		
1	3	3		
1	4	6		

8.2.2) (2 points) Peut-on réduire l'espace mémoire nécessaire pour le tableau $M[n+1][k+1]$? Justifiez votre réponse.

Oui, puisque la ligne i ne dépend que de la ligne $i-1$, les calculs des lignes antérieures ne sont plus nécessaires pour la suite des opérations.

8.2.3) (2 points) Quelle est selon vous la plus petite taille (en fonction des paramètres n et k) de la table $M[][]$ qui permette l'implémentation d'un algorithme de programmation dynamique pour le calcul des coefficients binomiaux ? Justifiez.

Il est possible de réduire l'espace mémoire à $1 \times k$. Il faut cependant changer le parcours des cases (de droite à gauche et non plus de gauche à droite). La réponse $2 \times k$ est acceptée en cela qu'elle illustre clairement la compréhension de la question 8.2.2.

Annexe I

Nom et prénom : _____

Matricule : _____

	Arcs incidents							
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								

	Arcs incidents							
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								

Solutionnaire examen final

INF2010

Le génie
sans frontières

Sigle du cours

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 - Structures de données et algorithmes		Tous	A2008
Professeur		Local	Téléphone
Tarek Ould Bachir (gr. 1) Ettore Merlo (gr 2)		A-416	5758
Jour	Date	Durée	Heures
Jeudi	18 décembre 2008	2h30	9h30

Documentation		Calculatrice
<input type="checkbox"/> Toute	<input type="checkbox"/> Aucune	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.
<input checked="" type="checkbox"/> Aucune	<input type="checkbox"/> Programmable	
<input type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Non programmable	

Directives particulières

Bonne chance à tous!

Important	Cet examen contient 6 questions sur un total de 12 pages (excluant cette page)
	La pondération de cet examen est de 40 %
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non

Question 1 : Graphes acycliques**(15 points)**

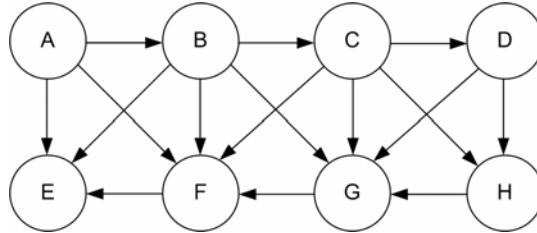
On vous présente un ensemble de graphes dirigés. Pour chacun des graphes qui suivent :

Indiquer si le graphe est cyclique

1.) Si c'est le cas, indiquer au moins un cycle

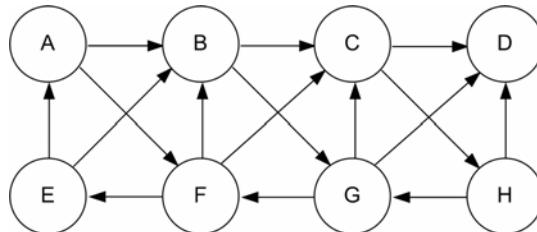
2.) Sinon, numérotter les nœuds de **A** à **H** de sorte que, si il existe un chemin du noeud **i** au nœud **j**, alors **i < j**.

a) Cyclique : OUI NON



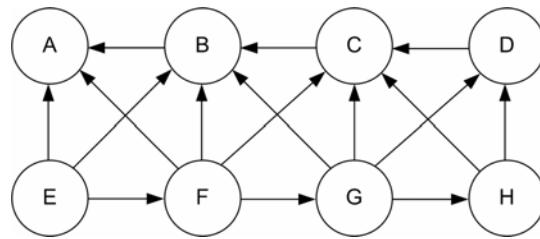
Noeud	Indegree avant la sortie de file							
	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0
B	1	0	0	0	0	0	0	0
C	1	1	0	0	0	0	0	0
D	1	1	1	0	0	0	0	0
E	3	2	1	1	1	1	1	0
F	4	3	2	1	1	1	0	0
G	4	4	3	2	1	0	0	0
H	2	2	2	1	0	0	0	0
Entrée	A	B	C	D	H	G	F	E
Sortie	A	B	C	D	H	G	F	E

b) Cyclique : OUI NON A -> F -> E -> A



Noeud	Indegree avant la sortie de file							
	1	2	3	4	5	6	7	8
A								
B								
C								
D								
E								
F								
G								
H								
Entrée								
Sortie								

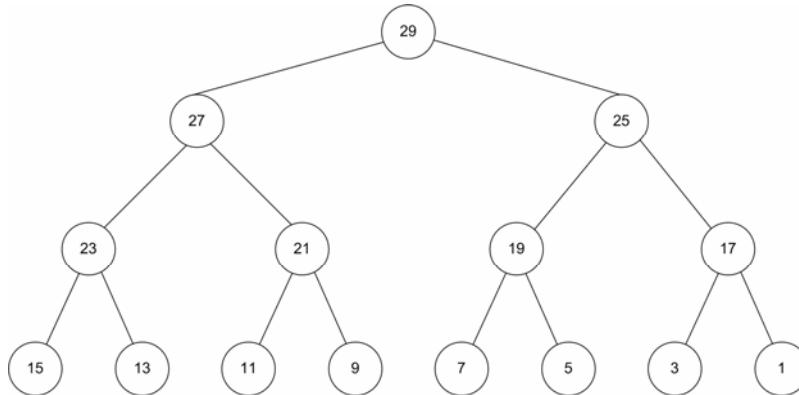
c) Cyclique : OUI NON



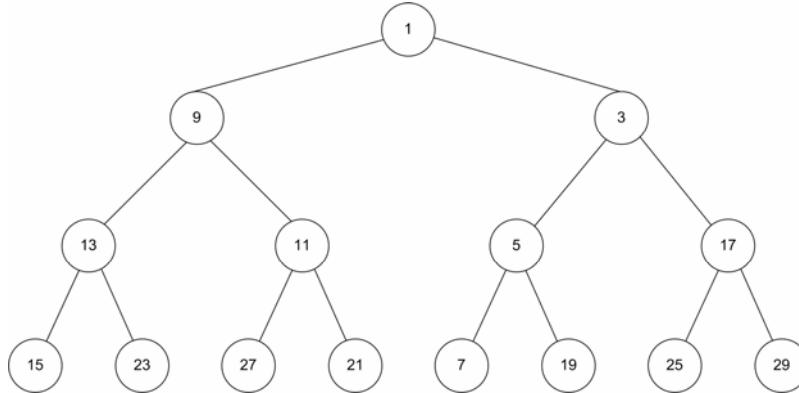
Noeud	Indegree avant la sortie de file							
	1	2	3	4	5	6	7	8
A	3	2	1	1	1	1	1	0
B	4	3	2	1	1	1	0	0
C	3	3	2	1	1	0	0	0
D	2	2	2	1	0	0	0	0
E	0	0	0	0	0	0	0	0
F	1	0	0	0	0	0	0	0
G	1	1	0	0	0	0	0	0
H	1	1	1	0	0	0	0	0
Entrée	E	F	G	H	D	C	B	A
Sortie	E	F	G	H	D	C	B	A

Question 2 : Monceaux (20 points)

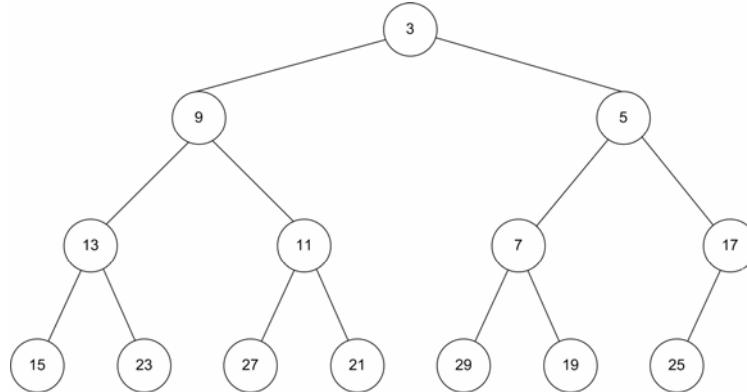
a) Construire, selon la technique vue dans le cours, un monceau à partir de l'arbre binaire suivant :



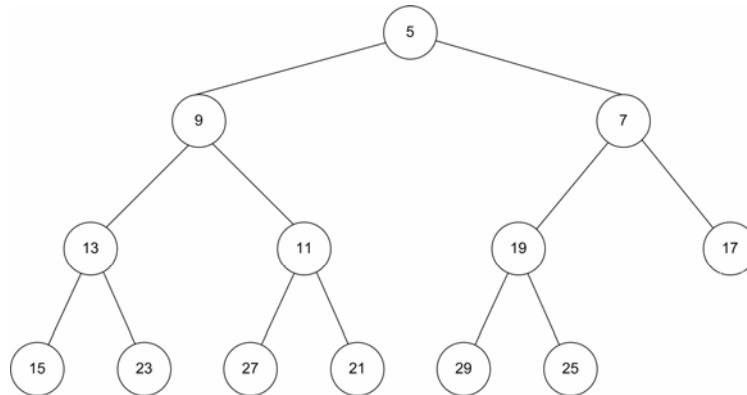
Monceau :



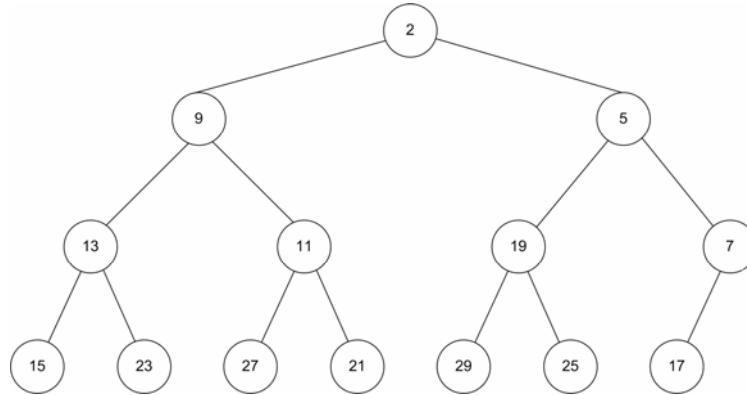
b) Dessiner l'état du monceau ainsi obtenu après l'appel à `deleteMin()` :



c) Dessiner l'état du monceau ainsi obtenu après l'appel à `deleteMin()` :



d) Ajouter au monceau ainsi obtenu un nœud dont la clé est 2 :

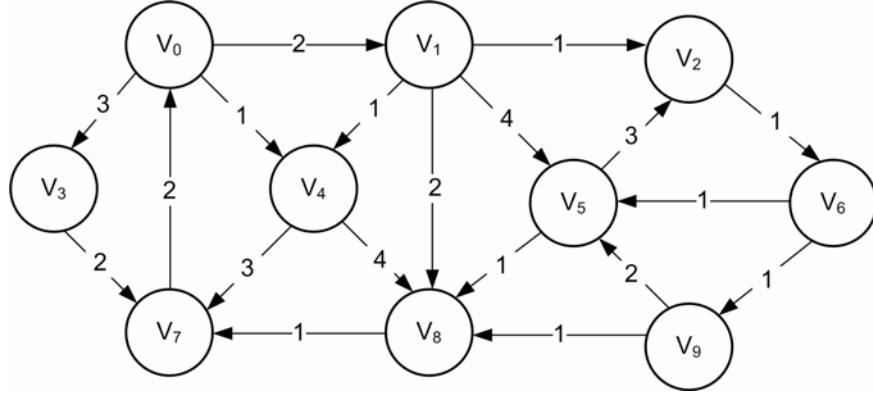


e) Dessiner l'état du tableau contenant le monceau à la fin de ces opérations :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	2	9	5	13	11	19	7	15	23	27	21	29	25	17	

Question 3 : Le plus court chemin dans un graphe (15 points)

En appliquant l'algorithme de Dijkstra utilisant une file de priorité (ici est partiellement réalisé), trouver la longueur du plus court chemin menant à chacun des nœuds du graphe en partant de V_0 .



a) Continuer l'exécution de l'algorithme en vous référant à l'état de la file de priorité

Nœud	Connu	Dist min.	Parent
V_0	✓	0	
V_1	✓	$\infty, 2$	V_0
V_2	✓	$\infty, 3$	$V_1,$
V_3	✓	$\infty, 3$	V_0
V_4	✓	$\infty, 1$	V_0
V_5	✓	$\infty, 6, 5$	$V_1, V_6,$
V_6	✓	$\infty, 4$	$V_2,$
V_7	✓	$\infty, 4$	V_4
V_8	✓	$\infty, 5, 4$	V_4, V_1
V_9	✓	$\infty, 5$	$V_6,$

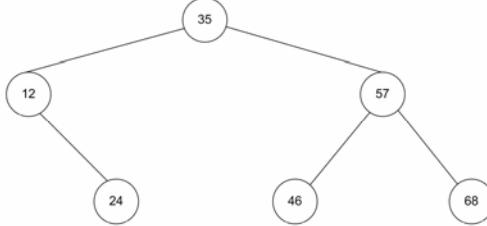
File de priorité
($V_3, 3$)
($V_2, 3$)
($V_7, 4$)
($V_8, 4$)
($V_6, 4$)
($V_5, 5$)
($V_9, 5$)

b) Détailler chacun des chemins les plus courts trouvés :

Nœud	Le plus court chemin	Distance parcourue
V_1	V_0, V_1	2
V_2	V_0, V_1, V_2	3
V_3	V_0, V_3	3
V_4	V_0, V_4	2
V_5	V_0, V_1, V_2, V_6, V_5	5
V_6	V_0, V_1, V_2, V_6	4
V_7	V_0, V_4, V_7	4
V_8	V_0, V_1, V_8	4
V_9	V_0, V_1, V_2, V_6, V_9	5

Question 4 : AVL**(15 points)**

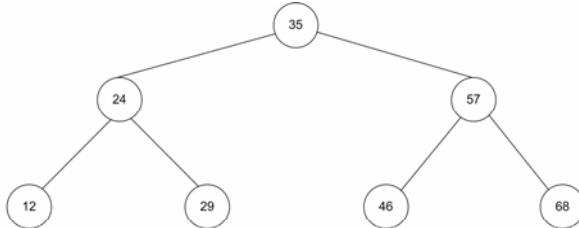
En considérant l'arbre AVL suivant :



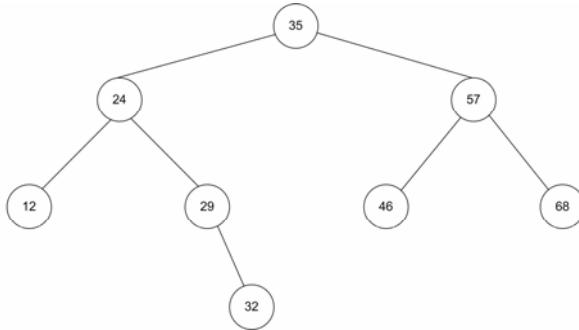
Effectuer l'ensemble des opérations suivantes dans l'ordre en vous servant des arbres ci-bas :

- a) Insérer 29
- b) Insérer 32
- c) Insérer 51
- d) Insérer 34
- e) Insérer 33
- f) Insérer 49

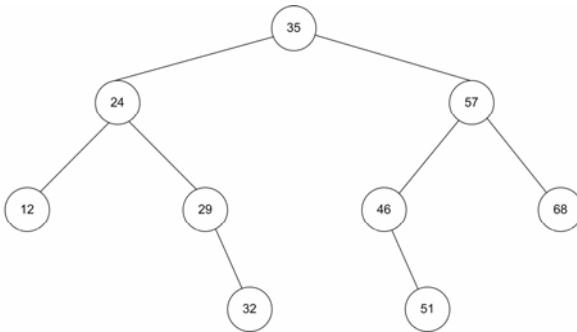
- a) Insérer 29



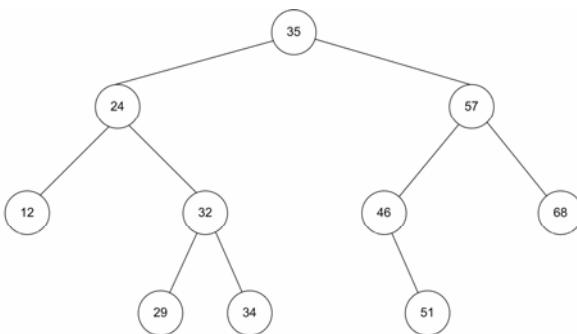
- b) Insérer 32



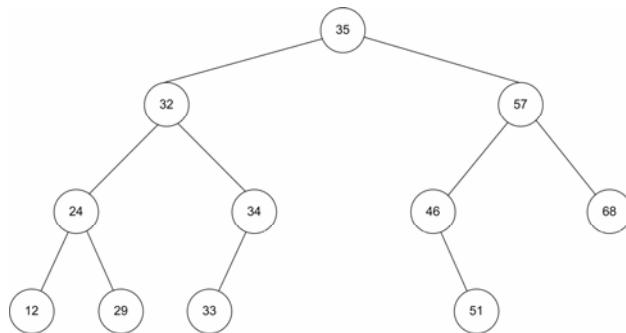
c) Insérer 51



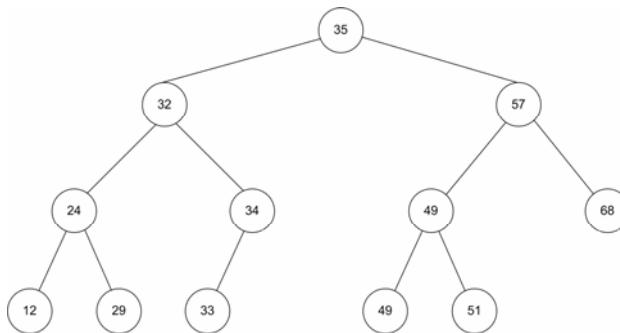
d) Insérer 34



e) Insérer 33



f) Insérer 49



Question 5 : Rabin-Karp**(20 points)**

L'algorithme Rabin Karp est un algorithme permettant de retrouver une chaîne de caractères dans un texte. La chaîne de caractères recherchée est alors remplacée par un nombre qu'il faut pré-calculer et on compare toute les valeurs des sous-séquences du texte à la valeur pré-calculée. Les différentes implémentations (**code java**) discutées dans cette question vous sont données à l'**annexe I**.

Dans ces implémentations de Rabin-Karp, le texte et le patron ne contiennent que les caractères numériques ‘0’-‘9’ dans l'encodage ASCII. Les valeurs respectives de ces caractères sont données ci-dessous :

Caractère	Val. ASCII	Caractère	Val. ASCII
‘0’	48	‘5’	53
‘1’	49	‘6’	54
‘2’	50	‘7’	55
‘3’	51	‘8’	56
‘4’	52	‘9’	57

On vous propose trois méthodes de calcul de la valeur numérique associée aux sous-chaînes :

Méthode	Description
#1	Polynôme dans la base $d = 58$; ex : $P = '123' p= (49*58+50)*58 + 51 = 167787$
#2	Polynôme dans la base $d = 10$; ex : $P = '123' p = (49*10+50)*58 + 51 = 5451$
#3	Polynôme modulaire dans la base $d = 58$ modulo 10; ex : $P = '123' p = 167787 \% 10 = 7$

a) Donner les avantages (au moins un) et les désavantages (au moins un) que vous voyez à utiliser chacune de ces méthodes :

Méthode	Avantages	Désavantages
#1	Pas de faux positifs	Résultat trop gros pouvant provoquer un débordement
#2	Résultat de valeur modérée	Beaucoup de faux positifs
#3	Résultat modéré, facile à calculer	Peut causer quelques faux-positifs

On vous propose la méthode de calcul suivante (option par défaut dans le code) :

Polynôme modulaire dans la base $d = 10$ modulo 11

Par exemple, $P = '123'$

$$\begin{aligned}
 p &= (((49\%11)*10 + 50) \% 11) * 10 + 51 \% 11 \\
 &= (((5)*10 + 50) \% 11) * 10 + 51 \% 11 \\
 &= (((100) \% 11) * 10 + 51) \% 11 \\
 &= ((1) * 10 + 51) \% 11 \\
 &= (61) \% 11 \\
 &= 6
 \end{aligned}$$

b) Donner la valeur de p pour le patron $P = '32123'$

7

c) Retrouver tous les décalages donnant la présence de P dans le texte $T = '3212323212321'$

Décalage (s)	Sous-chaîne	Valeur du polynôme	Égalité?	Faux positif?	Correspondance?
0	'32123'	7	✓		✓
1	'21232'	6			
2	'12323'	7	✓	✓	
3	'23232'	4			
4	'32321'	7	✓	✓	
5	'23212'	6			
6	'32123'	7	✓		✓
7	'21232'	6			
8	'12321'	5			

Faux positifs trouvés :

2 et 4

Décalages retournés :

0 et 6

Question 6 : DP-Matching**(15 points)**

En utilisant le tableau suivant, retrouver la plus longue sous-séquence commune aux chaînes d'entrée : X = ‘CTGAATGACTAG’ et Y = ‘CATAGTCACTAG’

	Y	C	A	T	A	G	T	C	A	C	T	A	G
X	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	D, 1	G, 1	G, 1	G, 1	G, 1	G, 1	D, 1	G, 1	D, 1	G, 1	G, 1	G, 1
T	0	H, 1	H, 1	D, 2	G, 2	G, 2	D, 2	G, 2	G, 2	G, 2	D, 2	G, 2	G, 2
G	0	H, 1	H, 1	H, 2	H, 2	D, 3	G, 3	D, 3					
A	0	H, 1	D, 2	H, 2	D, 3	H, 3	H, 3	H, 3	D, 4	G, 4	G, 4	D, 4	G, 4
A	0	H, 1	D, 2	H, 2	D, 3	H, 3	H, 3	H, 3	D, 4	H, 4	H, 4	D, 4	H, 4
T	0	H, 1	H, 2	D, 3	H, 3	H, 3	D, 4	G, 4	H, 4	H, 4	D, 5	G, 5	G, 5
G	0	H, 1	H, 2	H, 3	H, 3	D, 4	H, 4	H, 4	H, 4	H, 4	H, 5	H, 5	D, 6
A	0	H, 1	D, 2	H, 3	D, 4	H, 4	H, 4	H, 4	D, 5	G, 5	H, 5	D, 6	H, 6
C	0	D, 1	H, 2	H, 3	H, 4	H, 4	H, 4	D, 5	H, 5	D, 6	G, 6	H, 6	H, 6
T	0	H, 1	H, 2	D, 3	H, 4	H, 4	D, 5	H, 5	H, 5	H, 6	D, 7	G, 7	G, 7
A	0	H, 1	D, 2	H, 3	D, 4	H, 4	H, 5	H, 5	D, 6	H, 6	H, 7	D, 8	G, 8
G	0	H, 1	H, 2	H, 3	H, 4	D, 5	H, 5	H, 5	H, 6	H, 6	H, 7	H, 8	D, 9

Longueur de la plus longue sous-séquence commune :

9

Plus longue sous-séquence commune :

CTGTACTAG

Annexe I

```

import java.util.ArrayList;

public class RabinKarp {

    public static ArrayList<Integer>
        RabinKarpFind(String Text, String Pattern)
    {
        ArrayList<Integer> decalages = new ArrayList<Integer>();

        if( Text.length() < Pattern.length() )
            return decalages;

        int p = ComputePatternValue( Pattern );

        for(int i=0; i <= Text.length() - Pattern.length(); i++)
        {
            int t = ComputePatternValue(
                Text.substring( i, i+Pattern.length() ) );
            if( t == p )
            {
                int j;
                for(j=0; j< Pattern.length(); j++)
                {
                    if( Pattern.charAt( j ) != Text.charAt( i +j ) )
                    {
                        break;
                    }
                }
                if(j == Pattern.length())
                {
                    decalages.add( i );
                    System.out.println("Correspondance à " + i);
                }
                else
                {
                    System.out.println("Faux positif à " + i);
                }
            }
        }
        return decalages;
    }

    public static int ComputePatternValue(String Pattern)
    {
        return ComputePatternValue(Pattern, 0);
    }
}

```

```
public static int ComputePatternValue(String Pattern, int method)
{
    // TODO Auto-generated method stub
    int p = 0;

    switch( method )
    {
        case 1:
            for(int i=0; i<Pattern.length(); i++)
            {
                p *=58;
                p += (int) Pattern.charAt( i );
            }
            break;

        case 2:
            for(int i=0; i<Pattern.length(); i++)
            {
                p *=10;
                p += (int) Pattern.charAt( i );
            }
            break;

        case 3:
            for(int i=0; i<Pattern.length(); i++)
            {
                p *=58;
                p += (int) Pattern.charAt( i );
                p %= 10;
            }
            break;

        default:
            for(int i=0; i<Pattern.length(); i++)
            {
                p *=10;
                p += (int) Pattern.charAt( i );
                p %= 11;
            }
    }

    return p;
}

public static void main(String[] args)
{
    // TODO Auto-generated method stub
    String Pattern = "123";
    int p = ComputePatternValue(Pattern, 1);
    System.out.println( p );

    p = ComputePatternValue(Pattern, 2);
    System.out.println( p );

    p = ComputePatternValue(Pattern, 3);
    System.out.println( p );
}
```

```
p = ComputePatternValue(Pattern);
System.out.println( p );

Pattern = "32123";
p = ComputePatternValue(Pattern);
System.out.println( p );

String Text = "3212323212321";
ArrayList<Integer> decalages = RabinKarpFind(Text, Pattern);

for(int s : decalages )
{
    System.out.print( s + "\t" );
}
}
```

Questionnaire examen final

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

<i>Sigle et titre du cours</i>		<i>Groupe</i>	<i>Trimestre</i>
INF2010 - Structures de données et algorithmes		Tous	20073
<i>Professeur</i>		<i>Local</i>	<i>Téléphone</i>
Coordonnateur : Ettore Merlo, Chargé : Chamseddine Talhi		M-4021	5758 / 5193
<i>Jour</i>	<i>Date</i>	<i>Durée</i>	<i>Heures</i>
Lundi	17 décembre 2007	2h30	9h30
<i>Documentation</i>	<i>Calculatrice</i>		
<input type="checkbox"/> Toute <input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Programmable <input type="checkbox"/> Non programmable	Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	

Directives particulières

- ☞ Écrivez votre nom et votre matricule sur les pages réponses soit les pages 1, 2, 4, 5, 6 et 7.

Bonne chance à tous!

Important	Cet examen contient 5 questions sur un total de 14 pages (excluant cette page)
	La pondération de cet examen est de 55 %

Vous devez répondre sur : le questionnaire le cahier les deux

Vous devez remettre le questionnaire : oui non

Le plagiat, la participation au plagiat, la tentative de plagiat entraînent automatiquement l'attribution de la note **F** dans tous les cours suivis par l'étudiant durant le trimestre. L'École est libre d'imposer toute autre sanction jugée opportune, y compris l'exclus

Note : Remplissez les tableaux au besoin et laissez les cases non pertinentes vides.

Question 1. Tables de dispersement

(6 points)

Pour chaque question, donnez le numéro qui correspond à la bonne réponse.

1.1. Le facteur de compression dans une table de dispersement est le :

- (a) nombre d'éléments/la taille de l'espace libre dans la table.
- (b) nombre d'éléments/taille de la table.
- (c) nombre de collisions/le nombre des éléments insérés dans la table.
- (d) nombre de collisions/la taille de la table.
- (e) nombre de collision/la taille de l'espace libre dans la table.

Votre réponse :

(b)

1.2. Dans une table de dispersement, deux enregistrements synonymes sont :

- (a) deux éléments qui ont la même clé.
- (b) deux éléments dont les clés sont en collision.
- (c) deux éléments identiques.

Votre réponse :

(b)

1.3. Dans une table de dispersement de taille N, où les collisions sont gérées en utilisant les listes chaînées, la somme des tailles de toutes les listes utilisées est :

- (a) inférieure à $2N$.
- (b) plus grande que N .
- (c) inférieure à N .
- (d) sans limites.

Votre réponse :

(c)

1.4. Il est TOUJOURS possible d'insérer un élément dans une table de dispersement à débordement progressif par dispersement quadratique si :

- (a) la taille de la table correspond à un nombre pair et au moins le tiers ($\frac{1}{3}$) de l'espace de la table est vide.
- (b) la taille de la table correspond à un nombre impair et au moins le quart ($\frac{1}{4}$) de l'espace de la table est vide.
- (c) la taille de la table correspond à un nombre premier et au moins la moitié ($\frac{1}{2}$) de l'espace de la table est vide.
- (d) la taille de la table correspond à un nombre impair et au moins la moitié ($\frac{1}{2}$) de l'espace de la table est vide.
- (e) la taille de la table correspond à un nombre premier et au moins le tiers ($\frac{1}{3}$) de l'espace de la table est vide.

Votre réponse :

(c)

Nom : _____ Matricule : _____

Question 2. Arbres (12points)

2.1. Pour chaque question, donnez le numéro qui correspond à la bonne réponse (**5 points**).

a. Un arbre binaire complet de degré N contient :

- (a) au moins $2^{(N-1)}$ nœuds.
- (b) au moins $2^{(N+1)}$ nœuds.
- (c) au plus $2^{(N+1)}$ nœuds.
- (d) au plus $2^{(N-1)}$ nœuds.

Votre réponse :

(a)

b. Dans un arbre binaire de N nœuds, il y a :

- (a) au moins $N + 1$ fils NULL.
- (b) exactement $N + 1$ fils NULL.
- (c) exactement $N - 1$ fils NULL.
- (d) au plus N fils NULL.

Votre réponse :

(b)

c. En manipulant un arbre binaire implanté par des pointeurs, où chaque nœud pointe vers ses deux fils :

- (a) il est facile de parcourir l'arbre par niveau.
- (b) il est difficile de parcourir l'arbre en postordre.
- (c) il est difficile de parcourir l'arbre par niveau.

Votre réponse :

(c)

d. Dans un parcours préordre d'un arbre binaire :

- (a) on commence par le fils de gauche, puis le fils de droite et enfin le nœud lui-même.
- (b) on commence par le nœud puis son fils de gauche et enfin son fils de droite.
- (c) on commence par le fils de gauche, puis le nœud et enfin le fils de droite.

Votre réponse :

(b)

e. Après une insertion dans un arbre AVL :

- (a) on doit faire une rotation double si l'insertion a été effectuée dans le sous-arbre de gauche du fils gauche ou dans le sous-arbre de droite du fils de droite.
- (b) on doit faire une rotation simple si l'insertion a été effectuée dans le sous-arbre de droite du fils gauche ou dans le sous-arbre de gauche du fils de droite.
- (c) on doit faire une rotation double si l'insertion a été effectuée dans le sous-arbre de droite du fils gauche ou dans le sous-arbre de gauche du fils de droite.

Votre réponse :

(c)

- 2.2. Complétez le code des fonctions du [tableau 2.1 de la page 5](#). Les deux fonctions sont récursives. Chaque fonction reçoit comme paramètre une référence vers la racine d'un arbre binaire. La première fonction retourne le nombre de noeuds dans l'arbre binaire et la deuxième fonction retourne le nombre de feuilles dans l'arbre binaire. Quelle est la complexité en temps d'exécution de chaque fonction? Le nombre d'éléments dans l'arbre est N. **(7 points)**

Question 3. Monceaux **(12 points)**

Considérez le code en annexe 3.1 aux pages 8 à 11.

- a) On veut ajouter une fonction **void heapsort()** à la classe BinaryHeap. **heapsort()** trie en ordre descendant les éléments du monceau (dans le tableau *array* du code de l'annexe 3.1) sans l'utilisation d'un tableau intermédiaire. Donnez le code de la fonction **void heapsort()** (**Répondez sur le cahier**). **(5 points)**

Réponse :

```
void heapsort()
{
    for( int i = currentSize; i > 0; i-- )
    {
        int x = array[0];
        array[0] = array[i];
        array[i] = x;
        currentSize--;
        percDown( 0 );
    }
}
```

- b) On veut ajouter une fonction **int findMax()** à la classe BinaryHeap. Donnez le code de la fonction **int findMax()** en assurant un temps d'exécution dans le pire cas qui ne doit pas dépasser *currentSize*/2. (**Répondez sur le cahier**) **(5 points)**

Note : afin de pouvoir répondre à la question c), donnez votre algorithme même si son temps d'exécution en pire cas dépasse *currentSize*/2.

Réponse :

```

int findMax( )
{
    int x = 0;
    for( int i = currentSize/2; i <= currentSize; i++ )
    {
        if (array[i] < x) x = array[i];
    }
    return x ;
}

```

- c) Quel est le temps d'exécution de votre fonction **int findMax()** dans le pire cas ? Expliquez.
(Répondez sur le cahier) (2points)

Réponse :

currentSize/2

Le nombre d'itérations de la boucle while ne dépasse pas currentSize/2 !

Question 4. Graphes **(13 points)**

- a) En se basant sur les classes en annexe 4.1 aux pages 12, 13 et 14, et en admettant l'existence d'une classe **Queue** offrant les deux fonctions **enqueue** et **dequeue**, complétez le code de la méthode **unweighted()** ([tableau 4.1 page 6](#)) qui fera partie de la classe **graph**. La méthode **unweighted()** doit calculer pour chaque noeud du graphe, la longueur du plus court chemin menant de la racine du graphe (le premier noeud de nodeArr) vers ce noeud. La distance doit être sauvegardée dans la variable **dist**. En plus, cette méthode doit mettre à jour le lien **pred** de chaque noeud du graphe afin de pointer vers le noeud qui le précède dans le plus court chemin qui mène de la racine du graphe vers ce noeud. L'algorithme doit suivre la démarche améliorée, suggérée par M. A. Weiss que vous avez vue en cours et qui utilise une liste de travail (Worklist). **(7 points)**

Quelle est la complexité de cette méthode ? Expliquez! Considérez N et M comme nombre de noeuds et nombre d'arêtes du graphe respectivement. **(3 points)**

La complexité de cette fonction est $O(N + M)$

Explication :

Le nombre maximal de fils visités est égale au nombre d'arêtes.

Le nombre maximal de noeuds insérés puis retirés de la file est égale au nombre de noeuds dans le graphe

- b) En se basant sur les classes en annexe 4.1 aux pages 12, 13 et 14, écrivez la méthode **PrintPath(graphNode node)** qui imprime pour un nœud du graphe, le plus court chemin menant de la racine du graphe vers ce nœud. On suppose que la fonction **unweighted()** a déjà été exécutée et que la variable **pred** de chaque nœud pointe vers le nœud qui le précède dans le plus court chemin qui mène de la racine du graphe vers ce nœud. Respectez l'ordre des nœuds dans le chemin dans l'impression du chemin. (**Répondez sur le cahier**) **(3 points)**

Réponse :

```
PrintPath(graphNode node) {  
    if (node !=null){  
        PrintPath(node.pred);  
        System.out.println("to");  
    }  
    node.print();  
}
```

Question 5. Chaînes **(12 points)**

Considérez la chaîne $c = "bababb"$ et le texte $t = "abababbababb"$.

- a) Construisez l'automate à états finis qui effectue la recherche de la chaîne c dans un texte arbitraire. **(10 points)**

REmplisseZ le tableau 5.1 de la page 7.

- b) Identifiez le(s) décalage(s) ($shift(s)$) de concordance du texte t avec la chaîne c . **(2 points)**

REmplisseZ le tableau 5.2 de la page 7 avec le(s) décalage(s) identifié(s).

Nom : _____ Matricule : _____

Tableau 2.1**a)**

```
static int countNodes(BinaryNode t){ // (2points)
    if (t == null) return 0 ;
    return countNodes(t.left) + countNodes(t.right) + 1 ;
}
```

La complexité de cette fonction en temps d'exécution est : O(N) (1 point)

b)

```
static int countLeaves(BinaryNode t){ // (3points)
    if (t == null) { return 0 }
    else if (t.left == null && t.right == null)
        {return 1}
    else return countLeaves(t.left) + countLeaves(t.right) ;
}
```

La complexité de cette fonction en temps d'exécution est : O(N) (1 point)

- Le code de la classe `BinaryNode` est le suivant :

```
private static class BinaryNode<AnyType>
{
    BinaryNode( AnyType theElement ) // Constructors
    { this( theElement, null, null ); }

    BinaryNode( AnyType theElement, BinaryNode<AnyType> lt, BinaryNode<AnyType> rt )
    { element = theElement; left = lt; right = rt; }

    AnyType element; // The data in the node
    BinaryNode<AnyType> left; // Left child
    BinaryNode<AnyType> right; // Right child
}
```

Tableau 4.1**(7 points)**

```

void unweighted( ) // Le code demandé est une instruction ou
                  // une partie d'une instruction
{
    Queue< graphNode > q = new Queue<graphNode >();
    for( int i = 0; i < size(); i++ )
        { //Initialisation de dist et pred.
            getNode(i).init();
        }
    // attribuer la valeur 0 à dist de la racine
    getNode(0). UpdateDist(0);                                     ; /* Complétez */

    //insérer le noeud racine dans la file
    q.enqueue( getNode(0) );                                         ; /* Complétez */

    while( !q.isEmpty( ) )
    {
        //retirer un nœud de la file
        graphNode v = q.dequeue( );                                    ; /* Complétez */

        //récupérer les fils (adjacents) du noeud v
        adj kids = v.getKids();                                       ; /* Complétez */

        // lire le premier noeud adjacent
        kids.first();
        graphNode w = kids.getCurrent();
        // vérifier tous les fils
        while (w !=null)
        {
            if( w.getDist() == UNDEF_VAL )
            {
                // mettre à jour la distance de ce fils
                w. UpdateDist(v.getDist + 1);                           ; /* Complétez */

                // mettre à jour le prédécesseur de ce fils
                w. updatePred (v);                                      ; /* Complétez */

                q.enqueue( w );
            }
            // lire le prochain fils
            kids.next();
            w = kids.getCurrent();                                     ; /* Complétez */
        }
    }
}

```

Nom : _____ Matricule : _____

Tableau 5.1 (3 points)

Q	0,1,2,3,4,5,6
Q₀	0
F	6

Fonction de transition δ (7 points)

Symbole de l'alphabet

{ État }

	a	b		
0	0	1		
1	2	1		
2	0	3		
3	4	1		
4	0	5		
5	4	6		
6	2	1		
7				
8				

Tableau 4.2 (2 points)

0	1	2	3	4	5	6	7	8
	X					X		

Décalages identifiées

Annexe 3.1

```
import java.io.*;
import java.util.*;
import java.lang.*;

// BinaryHeap class
//
// CONSTRUCTION: with optional capacity (that defaults to 100)
//               or an array containing initial items
//
// ***** PUBLIC OPERATIONS *****
// void insert( x )      --> Insert x
// Comparable deleteMin( )--> Return and remove smallest item
// Comparable findMin( )  --> Return smallest item
// boolean isEmpty( )    --> Return true if empty; else false
// void makeEmpty( )     --> Remove all items
// ***** ERRORS *****
// Throws UnderflowException as appropriate

/**
 * Implements a binary heap.
 * Note that all "matching" is based on the compareTo method.
 * @author Mark Allen Weiss
 */
public class BinaryHeap<AnyType extends Comparable<? super AnyType>>
{
    /**
     * Construct the binary heap.
     */
    public BinaryHeap( )
    {
        this( DEFAULT_CAPACITY );
    }

    /**
     * Construct the binary heap.
     * @param capacity the capacity of the binary heap.
     */
    public BinaryHeap( int capacity )
    {
        currentSize = 0;
        array = (AnyType[]) new Comparable[ capacity + 1 ];
    }

    /**
     * Construct the binary heap given an array of items.
     */
    public BinaryHeap( AnyType [ ] items )
    {
```

```
{  
    currentSize = items.length;  
    array = (AnyType[]) new Comparable[ ( currentSize + 2 ) * 11 / 10 ];  
  
    int i = 1;  
    for( AnyType item : items )  
        array[ i++ ] = item;  
    buildHeap( );  
}  
  
/**  
 * Insert into the priority queue, maintaining heap order.  
 * Duplicates are allowed.  
 * @param x the item to insert.  
 */  
public void insert( AnyType x )  
{  
    if( currentSize == array.length - 1 )  
        enlargeArray( array.length * 2 + 1 );  
  
    // Percolate up  
    int hole = ++currentSize;  
    for( ; hole > 1 && x.compareTo( array[ hole / 2 ] ) < 0; hole /= 2 )  
        array[ hole ] = array[ hole / 2 ];  
    array[ hole ] = x;  
}  
  
private void enlargeArray( int newSize )  
{  
    AnyType [] old = array;  
    array = (AnyType []) new Comparable[ newSize ];  
    for( int i = 0; i < old.length; i++ )  
        array[ i ] = old[ i ];  
}  
  
/**  
 * Find the smallest item in the priority queue.  
 * @return the smallest item, or throw an UnderflowException if empty.  
 */  
public AnyType findMin( )  
{  
    if( isEmpty( ) )  
        //throw new UnderflowException( );  
        System.exit(1);  
    return array[ 1 ];  
}  
  
/**  
 * Remove the smallest item from the priority queue.  
 * @return the smallest item, or throw an UnderflowException if empty.  
 */  
public AnyType deleteMin( )
```

```
{  
    if( isEmpty( ) )  
        //throw new UnderflowException( );  
        System.exit(1);  
  
    AnyType minItem = findMin( );  
    array[ 1 ] = array[ currentSize-- ];  
    percolateDown( 1 );  
  
    return minItem;  
}  
  
/**  
 * Establish heap order property from an arbitrary  
 * arrangement of items. Runs in linear time.  
 */  
private void buildHeap( )  
{  
    for( int i = currentSize / 2; i > 0; i-- )  
        percolateDown( i );  
}  
  
/**  
 * Test if the priority queue is logically empty.  
 * @return true if empty, false otherwise.  
 */  
public boolean isEmpty( )  
{  
    return currentSize == 0;  
}  
  
/**  
 * Make the priority queue logically empty.  
 */  
public void makeEmpty( )  
{  
    currentSize = 0;  
}  
  
private static final int DEFAULT_CAPACITY = 10;  
  
private int currentSize;      // Number of elements in heap  
private AnyType [ ] array; // The heap array  
  
/**  
 * Internal method to percolate down in the heap.  
 * @param hole the index at which the percolate begins.  
 */  
private void percolateDown( int hole )
```

```
{  
    int child;  
    AnyType tmp = array[ hole ];  
  
    for( ; hole * 2 <= currentSize; hole = child )  
    {  
        child = hole * 2;  
        if( child != currentSize &&  
            array[ child + 1 ].compareTo( array[ child ] ) < 0 )  
            child++;  
        if( array[ child ].compareTo( tmp ) < 0 )  
            array[ hole ] = array[ child ];  
        else  
            break;  
    }  
    array[ hole ] = tmp;  
}  
  
public void print() {  
  
    int i = 0;  
  
    for (i=1; i <= currentSize; i++) {  
        if (array[i] != null)  
            System.out.println("POS: " + i + " VAL: " + array[i]);  
    }  
    System.out.println();  
}  
}
```

Annexe 4.1

```
import java.io.*;
import java.util.*;

class adj {
    static final int UNDEF_VAL = -9999;

    ArrayList<graphNode> adjList = new ArrayList<graphNode>();
    //ArrayIterator<graphNode> listIt = null;
    //graphNode curItem = null;
    int curIndex = UNDEF_VAL;

    void first() {
        curIndex = 0;
    }

    boolean currentIsValid() {
        return((curIndex >= 0) && (curIndex < adjList.size()));
    }

    graphNode getCurrent() {
        if ((curIndex >= 0) && (curIndex < adjList.size()))
            return(adjList.get(curIndex));
        else
            return(null);
    }

    void next() {
        if ((curIndex >= 0) && (curIndex < (adjList.size() - 1)))
            curIndex++;
        else
            curIndex = UNDEF_VAL;
    }

    void add(graphNode node) {
        adjList.add(node);
    }
}
```

```
import java.io.*;
import java.util.*;

class graph {

    static final int UNDEF_VAL = -9999;

    ArrayList<graphNode> nodeArr = new ArrayList<graphNode>();
    int curIndex = UNDEF_VAL;

    void initNodes() {
        int i = UNDEF_VAL;

        for (i = 0; i < nodeArr.size(); i++) {
            nodeArr.get(i).init();
        }
    }

    void first() {
        curIndex = 0;
    }

    boolean currentIsValid() {
        return((curIndex >= 0) && (curIndex < nodeArr.size()));
    }

    graphNode getCurrent() {
        if ((curIndex >= 0) && (curIndex < nodeArr.size()))
            return(nodeArr.get(curIndex));
        else
            return(null);
    }

    graphNode getNode(int pos) { // get(0) retourne la racine du graphe
        if ((pos >= 0) && (pos < nodeArr.size()))
            return(nodeArr.get(pos));
        else
            return(null);
    }

    void next() {
        if (curIndex < (nodeArr.size() - 1))
            curIndex++;
        else
            curIndex = UNDEF_VAL;
    }

    int size() {
        return(nodeArr.size());
    }
}
```

```
class graphNode {  
  
    static final int UNDEF_VAL = -9999;  
  
    int nodeId;  
    int dist; //distance du plus court chemin de la racine vers ce noeud  
    graphNode pred; // le prédécesseur dans le plus cours chemin  
    adj kids = new adj(); //Les noeuds reliés avec des arcs sortants  
    adj parents = new adj(); //Les noeuds reliés avec des arcs entrants  
  
    graphNode() {  
        nodeId = UNDEF_VAL;  
        dist = UNDEF_VAL;  
        pred = null;  
    }  
  
    void init() {  
        dist = UNDEF_VAL;  
        pred = null;  
    }  
  
    int getDist(){  
        return dist;  
    }  
  
    void updateDist(int newDist){  
        dist = newDist;  
    }  
  
    void updatePred(graphNode newPred){  
        pred = newPred;  
    }  
  
    adj getKids(){  
        return kids;  
    }  
  
    adj getParents(){  
        return parents;  
    }  
  
    void addKid (graphNode node) {  
        kids.add(node);  
    }  
  
    void addParent (graphNode node) {  
        parents.add(node);  
    }  
    void print() {  
        System.out.println("NODE: " + nodeId);  
    }  
};
```