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*	
☒ Aucune	☐ Aucune	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**
☐ Toute	☐ Toutes	
☒ Voir directives particulières	☒ 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.

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 : ☒ 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.

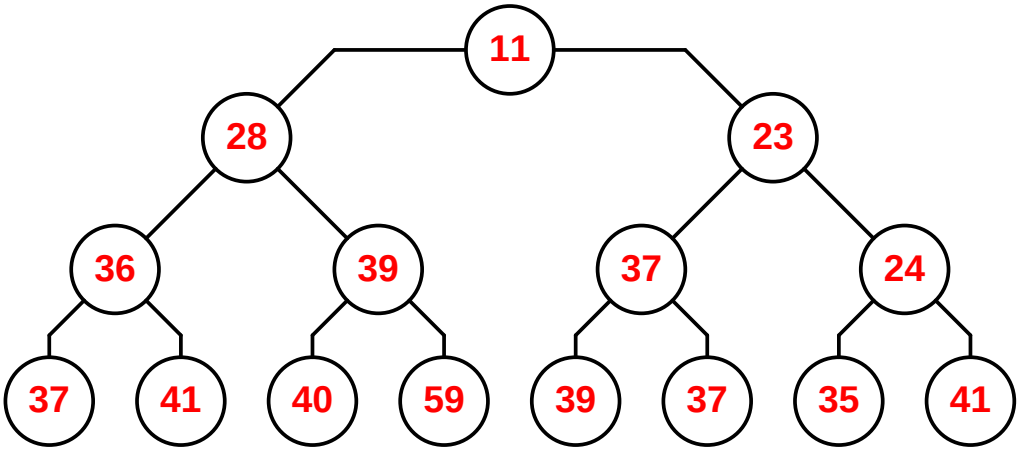
Question 1 : Monceaux (20 points)

Pour cette question, vous pouvez vous référez 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

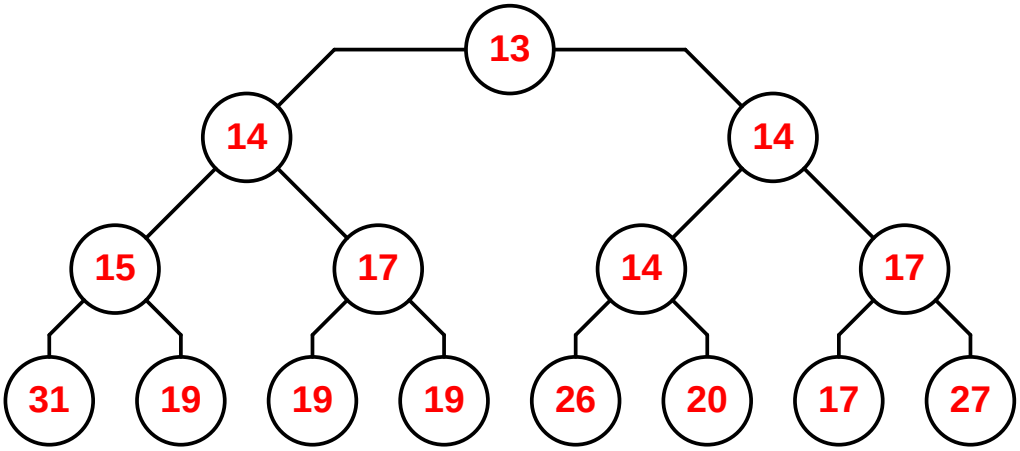
Votre réponse :



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 :

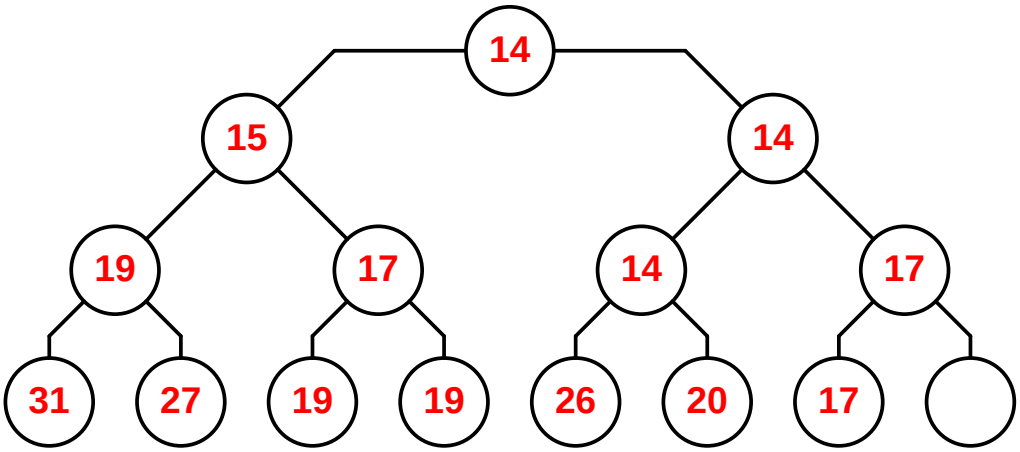


On se souviendra que l'on appelle ici le constructeur :

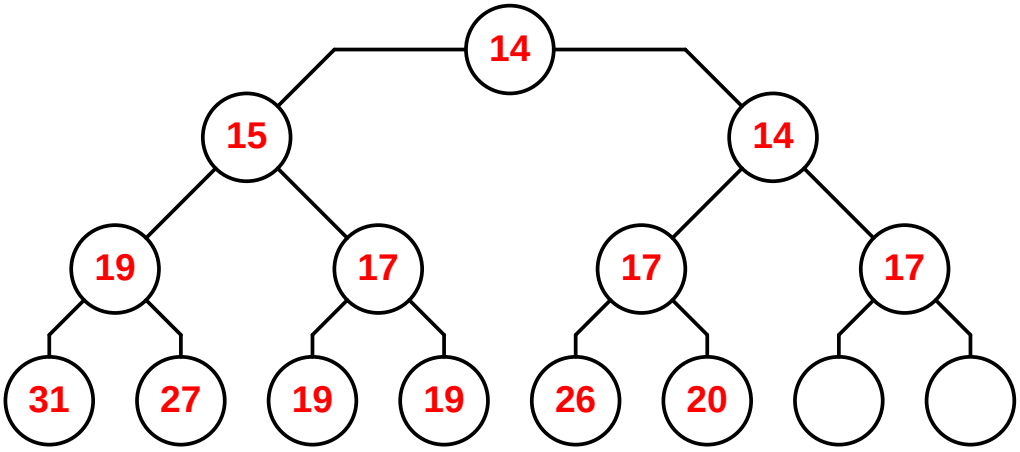
```
public BinaryHeap( AnyType [ ] items ).
```

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()`

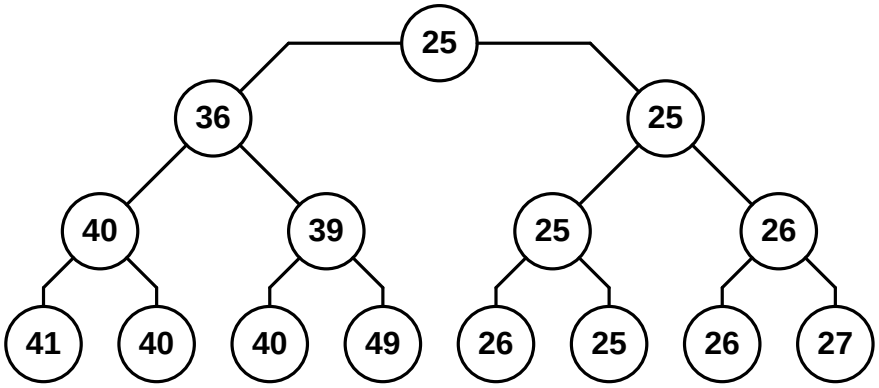


d) Considérez la fonction `changeKey(int location, int newKey)` donnée à l'Annexe 1 permettant de modifier la clé d'une entrée du monceau.

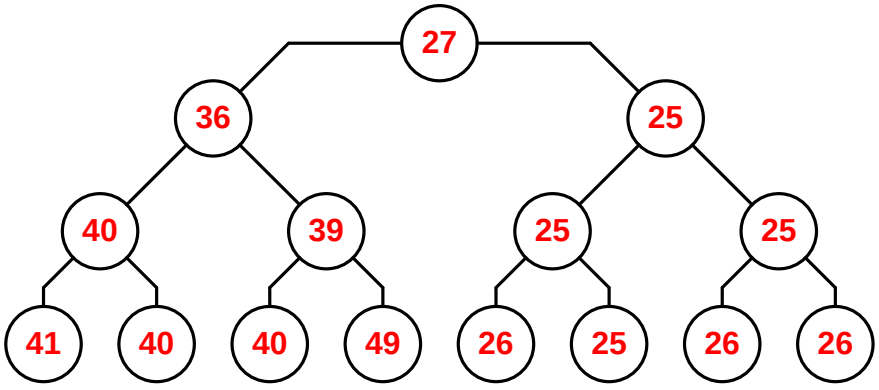
d.1) (3 pts) Quelle est la complexité asymptotique de cette fonction en pire cas ? Justifiez clairement votre réponse. Une réponse non justifiée ne sera pas considérée.

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 **nnnpnnnpnnnpnnn** (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 caractère. 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.

État	n	p	Autre
0	1	0	0
1	2	0	0
2	3	0	0
3	3	4	0
4	5	0	0
5	6	0	0
6	7	0	0
7	3	8	0
8	9	0	0
9	10	0	0
10	11	0	0
11	3	12	0
12	13	0	0
13	14	0	0
14	15	0	0
15	3	12	0

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	'I'	**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 x 1 ; A_2 : 1 x 3; A_3 : 3 x 1; A_4 : 1 x 4; A_5 : 4 x 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} x 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 équivalent. 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 1x2 et 2x2 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 équivalent. 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 :

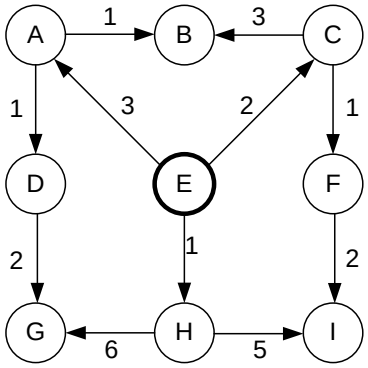
Nœud	A	B	C	D	E	F	G
Ordre:	**2**	**3**	**4**	**1**	**6**	**5**	**7**

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**

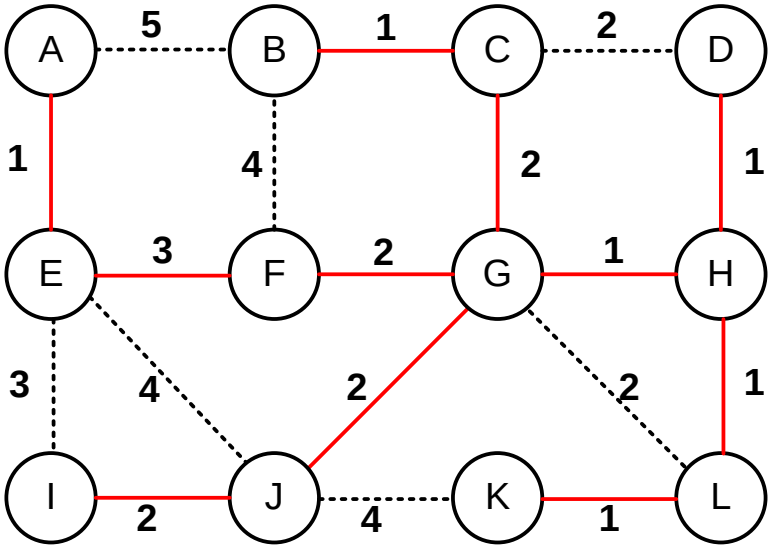
b) (6 pts) Détaillez chacun des chemins les plus courts trouvés parmi ceux demandés :

Destination	Le plus court chemin	Distance parcourue
B	E → **A** → B	**4**
G	E → **A** → **D** → G	**6**
I	E → **C** → **F** → I	**5**

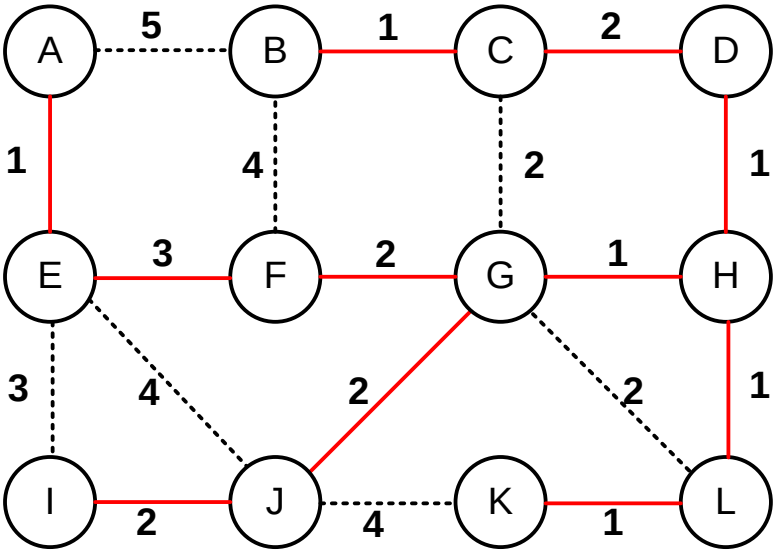
Question 6 : Arbre sous-tendant minimum (12 points)

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	**∞, 5, 4, 1**	**A, F, C**	
C	**∞, 2**	**G**	
D	**∞, 1**	**H**	
E	**∞, 1**	**A**	
F	**∞, 3**	**E**	
G	**∞, 2**	**F**	
H	**∞, 1**	**G**	
I	**∞, 3, 2**	**E, J**	
J	**∞, 4, 2**	**E, G**	
K	**∞, 1**	**L**	
L	**∞, 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							