Question 1: AVL (14 points)

CONSIDÉREZ l'arbre AVL dans la figure 1 suivante :

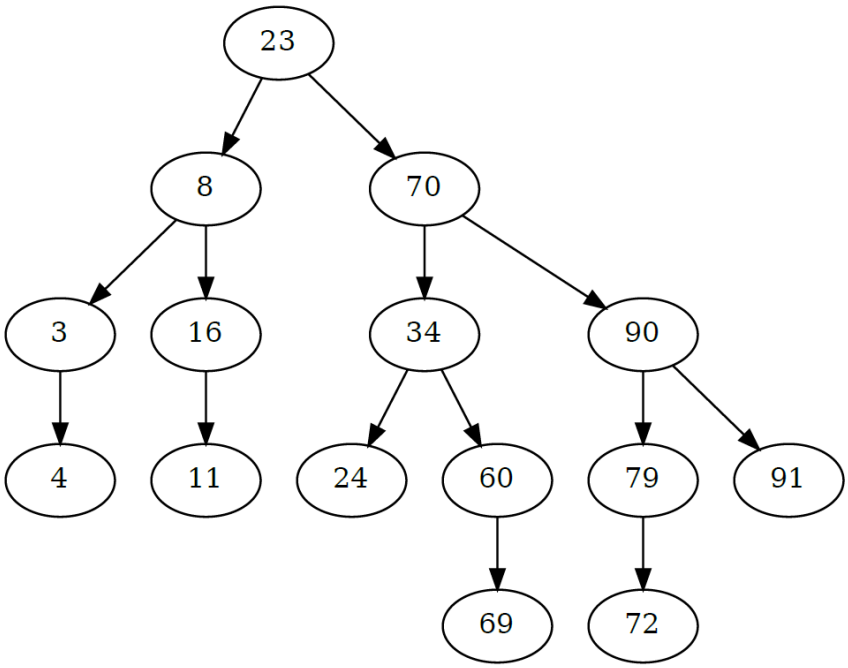


FIGURE 1

1.1 - (3.5 points) En partant de la figure 1, insérez le nœud 28 et remplissez le tableau suivant avec le nouvel arbre. ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

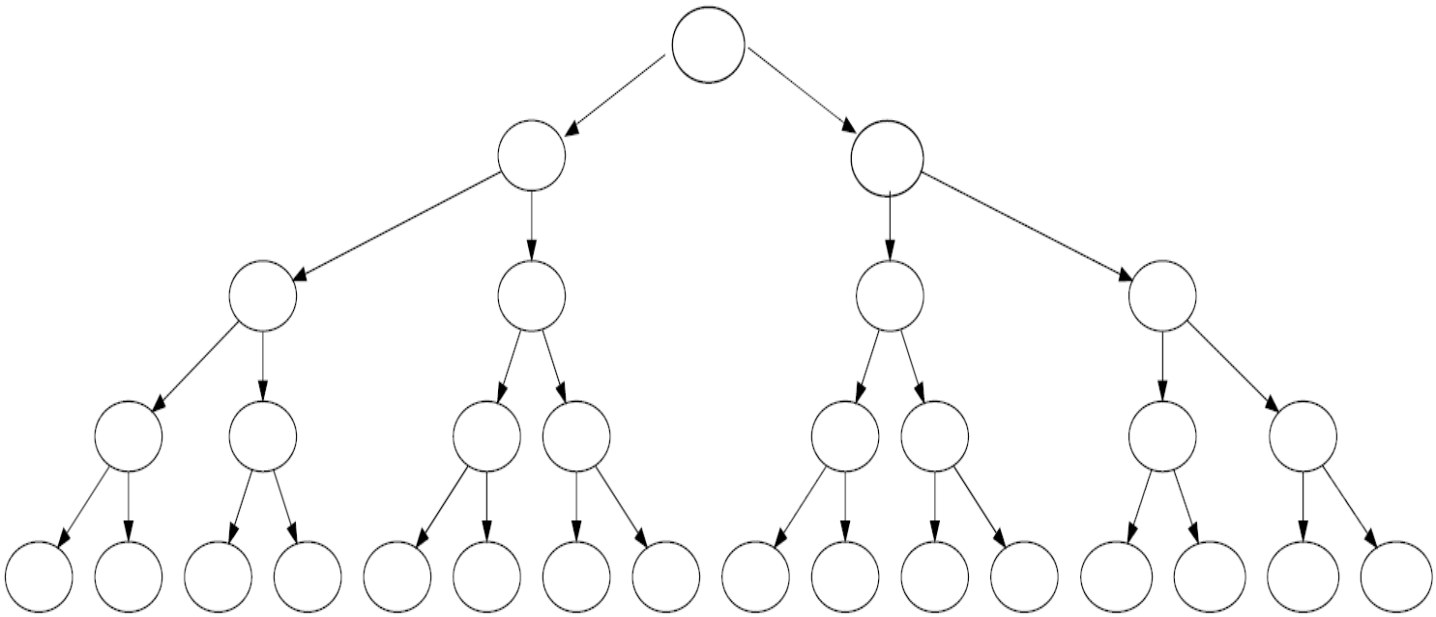


FIGURE 2

SOLUTION : AVL_10.pdf

1.2 - (3.5 points) En partant de l'arbre dans la figure 1, insérez le nœud 23 et remplissez le tableau suivant avec le nouvel arbre. ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

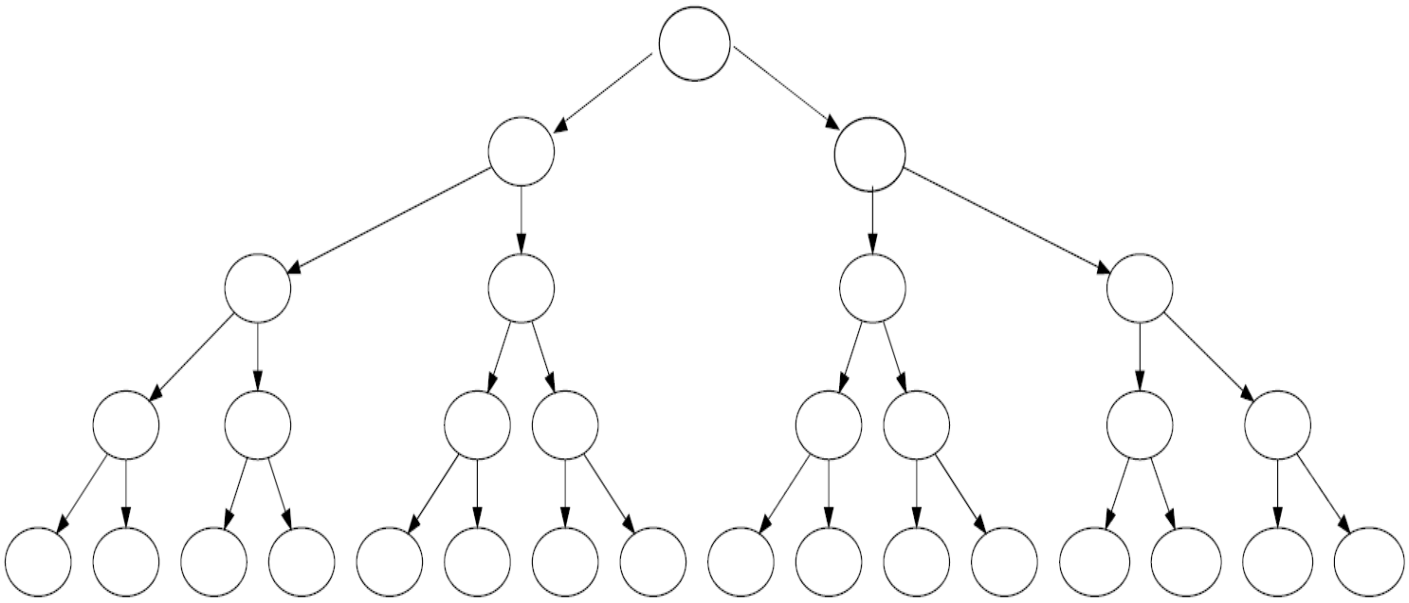


FIGURE 3

SOLUTION: pas de changements

1.3 - (3.5 points) En partant de l'arbre dans la figure 1, insérez le nœud 85 et remplissez le tableau suivant avec le nouvel arbre. ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

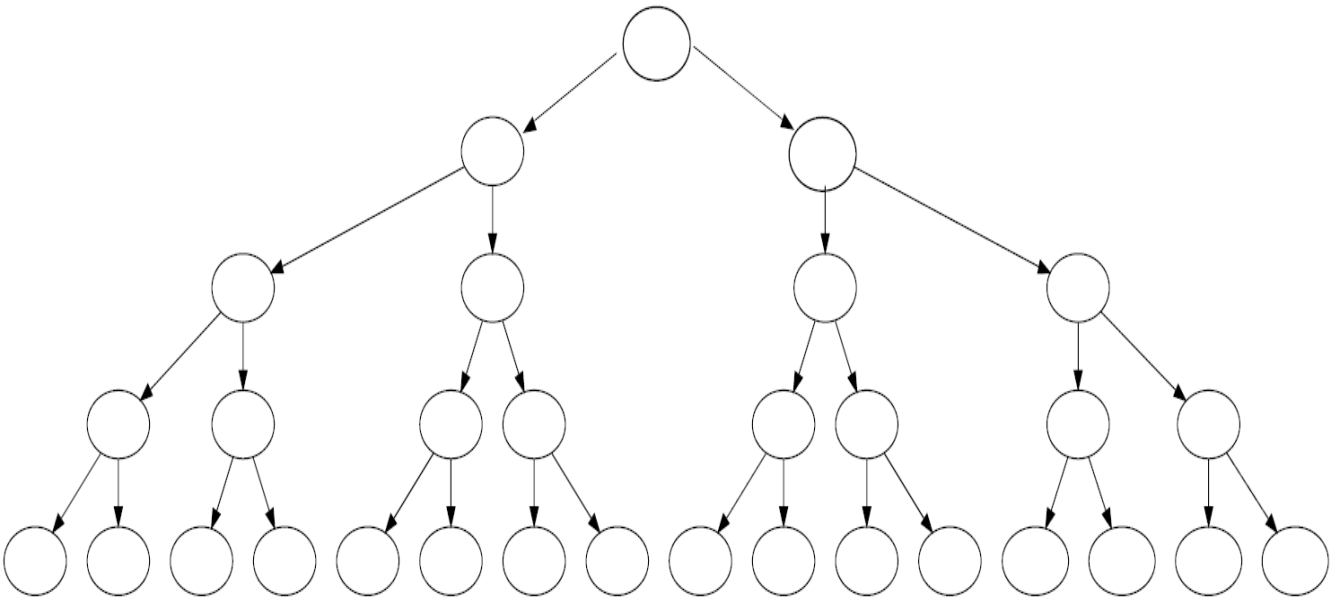


FIGURE 4

SOLUTION : AVL_12.pdf

1.4 - (3.5 points) En partant de l'arbre dans la figure 1, insérez le nœud 70 et remplissez le tableau suivant avec le nouvel arbre. ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

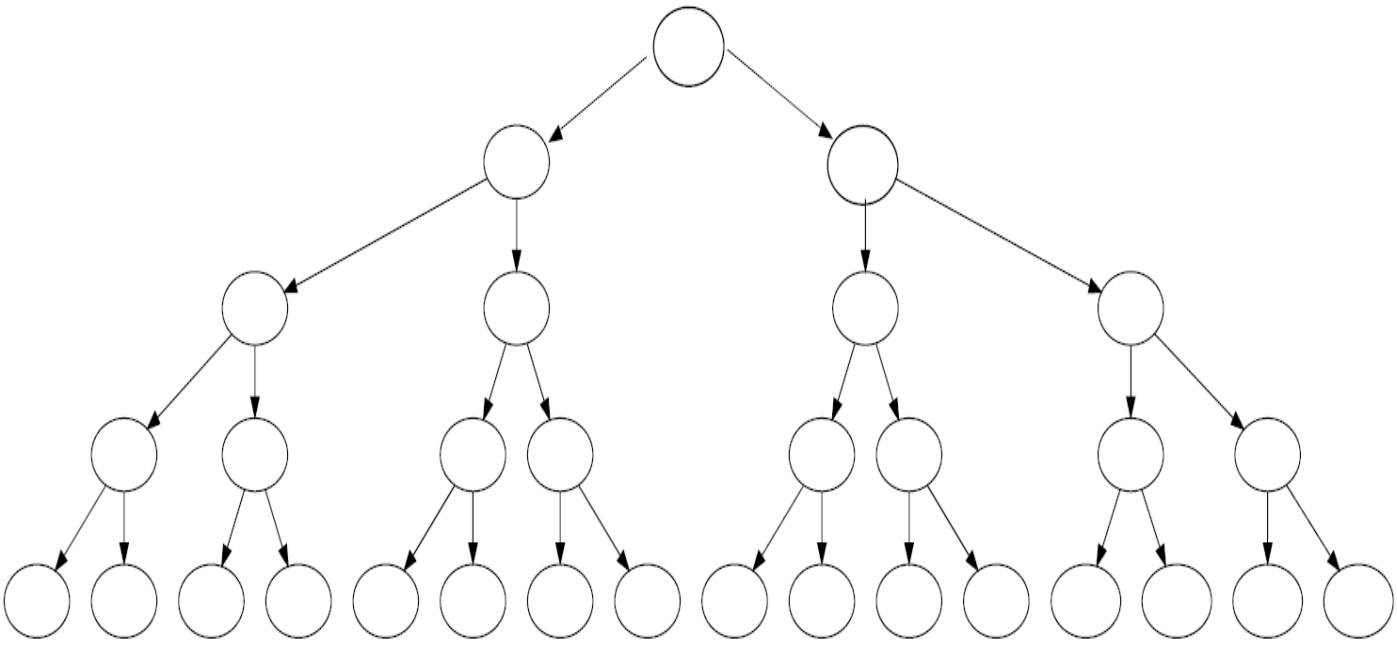


Figure 5

SOLUTION: pas de changements

Question 2 : Monceaux (14 points)

Considérez des monceaux dont la clef d'un nœud est inférieure ou égale à celle de ses descendants.

2.1 - (5 points) Considérez le vecteur suivant dans le tableau A :

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs		0	13	11	16	20	31	15	24	99	54	50	49	34	44	53			
Tableau A																			

Remplissez le tableau B avec le résultat de la construction d'un monceau à partir du vecteur précédant :

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau B																			

POS: 1 VAL: 0
POS: 2 VAL: 13
POS: 3 VAL: 11
POS: 4 VAL: 16
POS: 5 VAL: 20
POS: 6 VAL: 31
POS: 7 VAL: 15
POS: 8 VAL: 24
POS: 9 VAL: 99
POS: 10 VAL: 54
POS: 11 VAL: 50
POS: 12 VAL: 49
POS: 13 VAL: 34
POS: 14 VAL: 44
POS: 15 VAL: 53

0
13
16
24
99
20
54
50
11
31
49
34
15
44
53

2.2 - (3 points) Considérez le monceau suivant dans le tableau C :

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs		4	12	24	39	45	57	25	97	96	63	81	62	70	36	52			
Tableau C																			

Remplissez le tableau D avec le résultat de l'insertion de la valeur 93 au monceau du tableau C :
ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau D																			

POS: 1 VAL: 4
POS: 2 VAL: 12
POS: 3 VAL: 24
POS: 4 VAL: 39
POS: 5 VAL: 45
POS: 6 VAL: 57
POS: 7 VAL: 25
POS: 8 VAL: 93
POS: 9 VAL: 96
POS: 10 VAL: 63
POS: 11 VAL: 81
POS: 12 VAL: 62
POS: 13 VAL: 70
POS: 14 VAL: 36
POS: 15 VAL: 52
POS: 16 VAL: 97

```
4
  12
    39
      93
        97
      96
    45
      63
      81
  24
    57
      62
      70
    25
      36
      52
```

2.3 - (3 points) Considérez le monceau suivant dans le tableau E:

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs		12	18	17	64	34	21	83	89	75	62	76	74	44	90	95			
Tableau E																			

Remplissez le tableau F avec le résultat de l'insertion de la valeur 63 au monceau du tableau E :

ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau F																			

POS: 1 VAL: 12
POS: 2 VAL: 18
POS: 3 VAL: 17
POS: 4 VAL: 63
POS: 5 VAL: 34
POS: 6 VAL: 21
POS: 7 VAL: 83
POS: 8 VAL: 64
POS: 9 VAL: 75
POS: 10 VAL: 62
POS: 11 VAL: 76
POS: 12 VAL: 74
POS: 13 VAL: 44
POS: 14 VAL: 90
POS: 15 VAL: 95
POS: 16 VAL: 89

12
18
63
64
89
75
34
62
76
17
21
74
44
83
90
95

2.4 - (3 points) Considérez le monceau suivant dans le tableau G:

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs		14	18	16	19	31	35	27	36	55	39	42	57	53	67	47			
Tableau G																			

Remplissez le tableau suivant avec l'état du monceau H après l'appel à deleteMin() :
ATTENTION : ce n'est pas nécessaire de copier les nœuds qui n'ont pas changé.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau H																			

POS: 1 VAL: 16
POS: 2 VAL: 18
POS: 3 VAL: 27
POS: 4 VAL: 19
POS: 5 VAL: 31
POS: 6 VAL: 35
POS: 7 VAL: 47
POS: 8 VAL: 36
POS: 9 VAL: 55
POS: 10 VAL: 39
POS: 11 VAL: 42
POS: 12 VAL: 57
POS: 13 VAL: 53
POS: 14 VAL: 67

16
18
19
36
55
31
39
42
27
35
57
53
47
67

Question 3 : Algorithme de Rabin-Karp (recherche de motifs) (10 points)

Considérez l'algorithme de Rabin-Karp avec implantation par formule de Horner récursive et incrémentielle.

Considérez le texte de longueur 200 et des positions de 1 à 200 correspondant à la chaine de caractères suivante, contenant des chiffres décimaux:

Positions de 1 à 20:
2 1 3 0 1 3 2 0 1 3 0 2 3 2 0 1 3 2 1 3

Positions de 21 à 40:
0 2 1 0 0 3 2 1 2 2 1 3 0 1 2 1 3 0 1 0

Positions de 41 à 60:
0 2 1 3 0 2 3 2 3 0 1 1 0 3 2 0 1 3 2

Positions de 61 à 80:
1 3 2 1 3 0 2 1 2 0 3 1 1 0 2 1 3 2 0 3

Positions de 81 à 100:
0 0 1 0 2 1 3 0 3 2 1 3 2 1 0 3 1 2 0 2

Positions de 101 à 120:
1 3 0 1 3 0 1 1 2 3 0 2 1 0 3 1 2 3 0 1

Positions de 121 à 140:
0 2 3 0 3 1 1 3 0 1 2 3 1 0 2 1 3 0 1 0

Positions de 141 à 160:
2 1 3 0 1 3 2 0 1 2 3 1 0 3 2 1 0 2 3 3

Positions de 161 à 180:
2 0 2 1 3 0 2 1 1 3 0 1 0 2 3 1 2 3 0 3

Positions de 181 à 200:
2 1 0 0 1 2 3 0 1 2 0 0 2 1 3 3 2 0 1 2

Considérez le motif de longueur 4 et les positions de 1 à 4 correspondant à la chaine de caractères suivante, contenant des chiffres décimaux :

Positions de 1 à 4:
2 1 3 0

3.1 - (2 points) Donnez toutes les positions de CONCORDANCE du motif avec le texte, selon les valeurs de la formule calculée en modulo 100 (exemple: la concordance initiale est à la position 1):

1 ...

2130:

1 29 35 42 63 85 135 141 163

3.2 - (2 points) Donnez toutes les positions des FAUX POSITIFS du motif par rapport au texte, selon les valeurs de la formule calculée en modulo 100. Pour chacun des faux positifs, donnez aussi la valeur de la formule:

30:

3 10 32 37 44 49 65 87 102 105 110 118 123 137 142 165 169 178 187

3.3 - (2 points) Donnez toutes les positions des FAUX POSITIFS du motif par rapport au texte, selon les valeurs de la formule calculée en modulo 1000. Pour chacun des faux positifs, donnez aussi la valeur de la formule:

130:

2 9 31 36 43 64 86 101 103 127 136 142 164 168

3.4 - (2 points) Donnez toutes les positions des FAUX POSITIFS du motif par rapport au texte, selon les valeurs de la formule calculée en modulo 10000. Pour chacun des faux positifs, donnez aussi la valeur de la formule:

2130 :

Aucune

3.5 - (2 points) Donnez la formule de la complexité asymptotique du pire cas pour l'algorithme de Rabin-Karp en fonction d'un texte de longueur n et d'un motif de longueur m :

O(n*(n-m+1))

Question 4 : Le plus court chemin sans poids dans un graphe dirigé (14 points)

Considérez le graphe dirigé suivant en figure 6 :

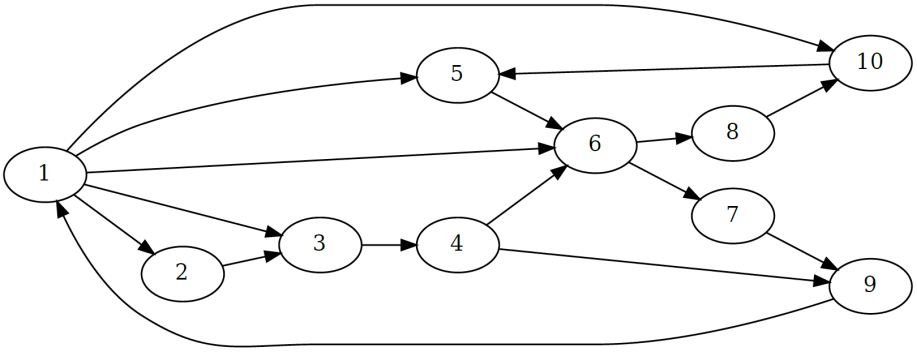


Figure 6

4.1 – (6 points) En appliquant l'algorithme du plus court chemin sans poids selon l'algorithme de Dijkstra, trouvez la longueur du plus court chemin menant à chacun des nœuds du graphe dirigé en partant du nœud 1, calculez le parent de chaque nœud selon ce chemin et remplissez le tableau J suivant avec les résultats:

ATTENTION

- Dans les boucles des algorithmes, traversez les nœuds et les adjacents d'un nœud STRICTEMENT EN ORDRE NUMÉRIQUE ASCENDANT des identificateurs de nœuds

Nœud	Distance minimale	Parent
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Tableau J

```
PATH DIST --->
NODE:  1 DIST: 0
NODE:  2 DIST: 5
NODE:  3 DIST: 6
NODE:  4 DIST: 9
NODE:  5 DIST: 10
NODE:  6 DIST: 11
NODE:  7 DIST: 6
<--- PATH DIST

PATH PRED --->
NODE:  1 PRED: UNDEF
NODE:  2 PRED: 1
NODE:  3 PRED: 1
NODE:  4 PRED: 7
NODE:  5 PRED: 4
NODE:  6 PRED: 7
NODE:  7 PRED: 1
<--- PATH PRED
```

4.2 - (4 points) Écrivez l'algorithme de recherche en largeur (BFS) sur un graphe dirigé.

```
void BFS(G, s) {

   queue q = new queue();

   visited = {};

   q.enqueue(s);

   while (!q.empty()) {

         curNode = q.dequeue();
         print(curNode);

         for adjNode in adjacents(curNode) {
            if (!visited.contains(adjNode)) {
               q.enqueue(adjNode);
            }
             }
   }

   return;
}
```

4.3 - (4 points) Décrivez les modifications requises pour calculer le chemin minimum sans poids sur un graphe dirigé en utilisant l'algorithme de BFS de la réponse 4.2 :

Il suffit d'introduire les vecteurs distance (initialement a zero) et p (predecesseur, initialement non defini) et les mettre a jour dans de la boucle sur les adjacents si pas deja visite :

- la distance est incrementee de un
- le predecesseur est le nœud de provenance

```
void plusCourtChemin(G, s) {

   queue q = new queue();
__
   visited = {};
__
   for node in G.V {
         distance[node] = 0;
         p[node] = UNDEF;
   q.enqueue(s);

   while (!q.empty()) {

      curNode = q.dequeue();
      print(curNode, distance[node], p[node]);

      for adjNode in adjacents(curNode) {
            if (!visited.contains(adjNode)) {
_____
               distance[adjNode] = distance[curNode] + 1;
               p[adjNode] = curNode;
_____
               q.enqueue(adjNode);
            }
         }
   }

   return;
}
```

Question 5 : DFS généralisé (10 points)

Considérez le graphe dirigé suivant en figure 7:

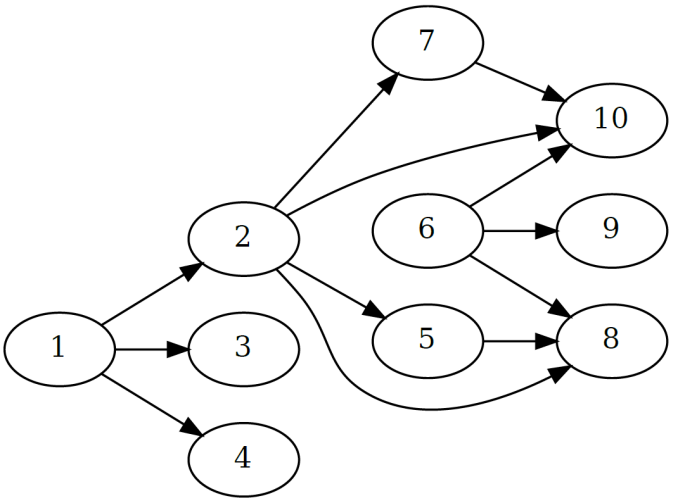


Figure 7

5.1 – (10 points) Calculez les temps de découverte (u.d) et les temps de fermeture (u.f) des nœuds traversés selon l'algorithme DFS présenté en Annexe 1 (extrait du livre "Introduction to Algorithms - T. Cormen, C. Leiserson, R. Rivest, C. Stein, MIT Press, 2009")

ATTENTION

- Dans les boucles aux lignes 5 de DFS et 4 de DFS-visit, traversez les nœuds et les adjacents d'un nœud STRICTEMENT EN ORDRE NUMÉRIQUE ASCENDANT des identificateurs de nœuds.
- Considérez le temps initial = 0 (ligne 4 de DFS)

Remplissez le tableau K suivant avec les résultats :

Nœud u	u.d	u.f
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Tableau K

```
D_TIMES --->
        NODE: 1 D_TIME: 1
        NODE: 2 D_TIME: 2
        NODE: 3 D_TIME: 12
        NODE: 4 D_TIME: 14
        NODE: 5 D_TIME: 3
        NODE: 6 D_TIME: 17
        NODE: 7 D_TIME: 7
        NODE: 8 D_TIME: 4
        NODE: 9 D_TIME: 18
        NODE: 10 D_TIME: 8
<--- D_TIMES
```

```
F_TIMES --->

        NODE: 1 F_TIME: 16

        NODE: 2 F_TIME: 11

        NODE: 3 F_TIME: 13

        NODE: 4 F_TIME: 15

        NODE: 5 F_TIME: 6

        NODE: 6 F_TIME: 20

        NODE: 7 F_TIME: 10

        NODE: 8 F_TIME: 5

        NODE: 9 F_TIME: 19

        NODE: 10 F_TIME: 9

<--- F_TIMES
```

6. – (14 points) Ordre topologique

6.1 – (2 points) Donnez la définition mathématique formelle d'ordre topologique d'un graphe G = (V, E).

1) Pour toutes les paires de nœuds (a, b), si il existe un chemin entre les nœuds a et b, le noeud a doit preceder le nœud b dans l'ordre topologique.

2) Pour toutes les aretes (v, w), le nœud de depart v d'une arete doit preceder le nœud d'arrive w de l'arete dans l'ordre topologique.

Pour toutes a = (v, w) in G.E → ordre(v) ≤ ordre(w)

6.2 – (4 points) Calculez l'ordre topologique du graphe de la figure 8 en utilisant l'algorithme de DFS généralisé en Annexe 1et remplissez le tableau L avec le résultat.

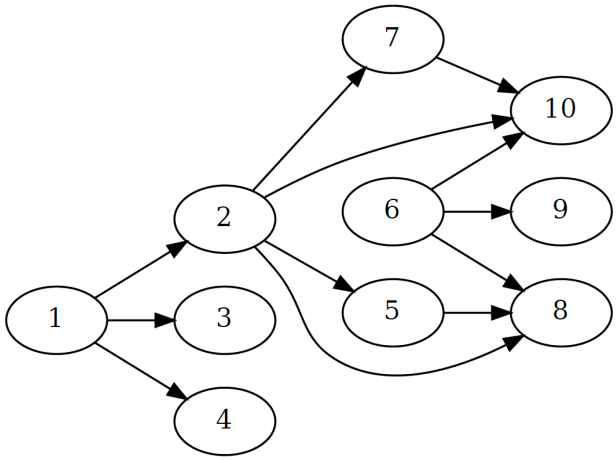


Figure 8

ATTENTION

- Dans les boucles des algorithmes, traversez les nœuds et les adjacents d'un nœud STRICTEMENT EN ORDRE NUMÉRIQUE ASCENDANT des identificateurs de nœuds

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																		
Tableau L																		

FINISH ORDER --->
POS: 1 NODE: 8
POS: 2 NODE: 5
POS: 3 NODE: 10
POS: 4 NODE: 7
POS: 5 NODE: 2
POS: 6 NODE: 3
POS: 7 NODE: 4
POS: 8 NODE: 1
POS: 9 NODE: 9
POS: 10 NODE: 6
<--- FINISH ORDER

6, 9, 1, 4, 3, 2, 7, 10, 5, 8

6.3 – (2 points) Démontrez ou réfutez que la séquence suivante représente un ordre topologique pour le graphe en figure 8 (attention, l'ordre de traversement des noeuds n'est pas nécessaire dans cette question)

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs	6	9	1	3	4	2	5	7	10	8								
Tableau M																		

Il faut verifier que (selon la definition) que le nœud de depart d'une arete precede le nœud d'arrive de l'arete dans l'ordre topologique ou de trouver un contre-example.

6->9: VRAI
6->8: VRAI
6->10: VRAI
1->2: VRAI
1->3: VRAI
1->4: VRAI
2->8: VRAI
2->10: VRAI
2->5: VRAI
2->7: VRAI
5->8: VRAI
7->10: VRAI

6.4 – (2 points) Démontrez ou réfutez que la séquence suivante représente un ordre topologique pour le graphe en figure 8 (attention, l'ordre de traversement des nœuds n'est pas nécessaire dans cette question)

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs	6	9	5	1	2	3	4	7	8	10								
Tableau N																		

l faut verifier que (selon la definition) que le nœud de depart d'une arete precede le nœud d'arrive de l'arete dans l'ordre topologique ou de trouver un contre-example.

6->9: VRAI
6->8: VRAI
6->10: VRAI
1->2: VRAI
1->3: VRAI
1->4: VRAI
2->8: VRAI
2->10: VRAI
2->5: FAUX
... ca suffit!
2->7: VRAI
5->8: VRAI
7->10: VRAI

6.5 – (4 points) Donnez l'algorithme pour obtenir l'ordre topologique en partant des temps calculés par l'algorithme DFS généralisé (référence en Annexe 1)

L'ordre inverse des temps de fermeture calcule avec l'algorithme DFS généralisé represente un ordre topologique.

```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finishing times v.f for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

Question 7 : arbre sous-tendant minimale (9 points)

Considérez le graphe non-dirigé suivant en figure 10 :

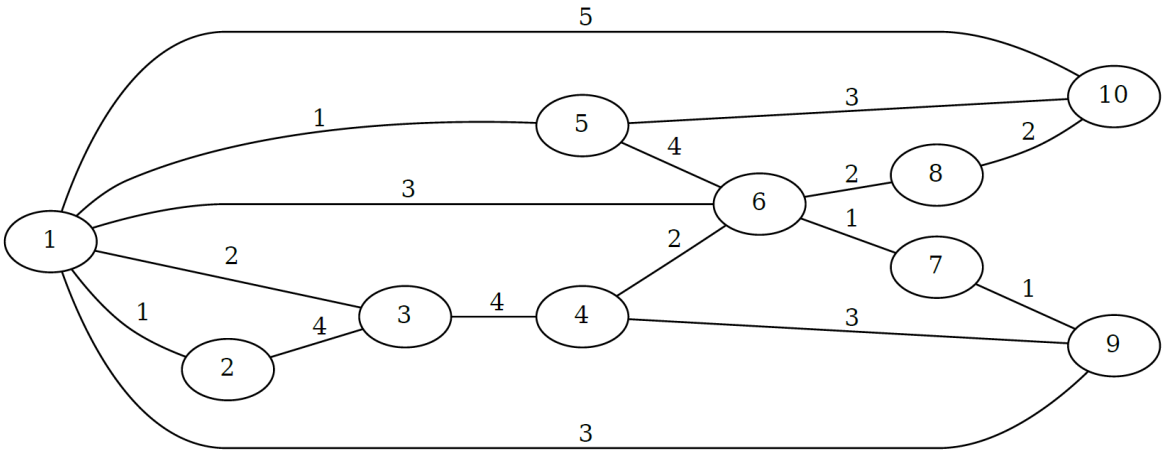


Figure 10

Les poids des arêtes non-dirigées sont aussi indiqués dans le tableau P suivant :

Nœuds		
Départ	Arrivée	Poids
1	2	1
1	3	2
1	5	1
1	6	3
1	10	5
2	3	4
3	4	4
4	6	2
4	9	3
5	6	4
6	7	1
6	8	2
7	9	1
8	10	2
9	1	3
10	5	3
Tableau P		

Calculez l'arbre sous-tendant minimal en utilisant l'algorithme de Prim.

ATTENTION :

- Dans les boucles des algorithmes, traversez les adjacents d'un nœud STRICTEMENT EN ORDRE NUMÉRIQUE ASCENDANT des identificateurs de nœuds.

7.1 – (3 points) Donnez le coût minimal de l'arbre obtenu :

COST: 15

7.2 – (6 points) Identifiez dans la figure 10 précédente les arêtes appartenant à l'arbre en question en les coloriant ou en les marquant en noir gras.

MST --->
EDGE: (2 1) W: 1
EDGE: (3 1) W: 2
EDGE: (4 6) W: 2
EDGE: (5 1) W: 1
EDGE: (6 1) W: 3
EDGE: (7 6) W: 1
EDGE: (8 6) W: 2
EDGE: (9 7) W: 1
EDGE: (10 8) W: 2
<--- MST

Question 8 : Ensembles disjoints (15 points)

Considérez les fonctions « union » et « find » en figure 12 suivante :

```
public void union(int el1, int el2) {
        int root1 = find(el1);
        int root2 = find(el2);
        if(root1 == root2){
                return;
        }

        if(s[root2] < s[root1]) {
                s[root1] = root2;
        }
        else{
                if(s[root1] == s[root2]) {
                        --s[root1];
                }
                s[root2] = root1;
        }
}

public int find(int x) {
        if(s[x] < 0){
                return x;
        }
        else{
                s[x] = find(s[x]);
                return s[x];
        }
}
```

Figure 12

Considérez les ensembles disjoints représentés par arbre dans le tableau initial Q suivant :

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs	-4	0	0	2	-2	4	0	6	6	8	-1	-3	11	11	13	-1	-2	16	
Tableau Q																			

<u>8.1 – (3 points)</u> A partir des mêmes ensembles initiaux du tableau Q, remplissez le tableau R avec le résultat de l'opération « find(3) » :

ATTENTION : ce n'est pas nécessaire de remplir les cases qui n'ont pas changé

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau R																			

SET --->
POS: 0 VAL: -4
POS: 1 VAL: 0
POS: 2 VAL: 0
POS: 3 VAL: 0
POS: 4 VAL: -2
POS: 5 VAL: 4
POS: 6 VAL: 0
POS: 7 VAL: 6
POS: 8 VAL: 6
POS: 9 VAL: 8
POS: 10 VAL: -1
POS: 11 VAL: -3
POS: 12 VAL: 11
POS: 13 VAL: 11
POS: 14 VAL: 13
POS: 15 VAL: -1
POS: 16 VAL: -2
POS: 17 VAL: 16
<--- SET

TREE SETS --->
SET --->
0
1
2
3
6
7
8
9
<--- SET

SET --->
4
5
<--- SET

SET --->
10
<--- SET

SET --->
11
12
13
14
<--- SET

SET --->
15
<--- SET

SET --->
16
17
<--- SET

8.2 – (3 points) A partir des mêmes ensembles initiaux du tableau Q, remplissez le tableau S avec le résultat de l'opération « find(17) »:

ATTENTION : ce n'est pas nécessaire de remplir les cases qui n'ont pas changé

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau S																			

SET --->
POS: 0 VAL: -4
POS: 1 VAL: 0
POS: 2 VAL: 0
POS: 3 VAL: 2
POS: 4 VAL: -2
POS: 5 VAL: 4
POS: 6 VAL: 0
POS: 7 VAL: 6
POS: 8 VAL: 6
POS: 9 VAL: 8
POS: 10 VAL: -1
POS: 11 VAL: -3
POS: 12 VAL: 11
POS: 13 VAL: 11
POS: 14 VAL: 13
POS: 15 VAL: -1
POS: 16 VAL: -2
POS: 17 VAL: 16
<--- SET

TREE SETS --->
SET --->
0
1
2
3
6
7
8
9
<--- SET

SET --->
4
5
<--- SET

SET --->
10
<--- SET

SET --->
11
12
13
14
<--- SET

SET --->
15
<--- SET

SET --->
16
17
<--- SET

<--- TREE SETS

8.3 – (3 points) A partir des mêmes ensembles initiaux du tableau Q, remplissez le tableau T avec le résultat de l'opération « find(14) » :

ATTENTION : ce n'est pas nécessaire de remplir les cases qui n'ont pas changé

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau T																			

```
SET --->
        POS: 0 VAL: -4
        POS: 1 VAL: 0
        POS: 2 VAL: 0
        POS: 3 VAL: 2
        POS: 4 VAL: -2
        POS: 5 VAL: 4
        POS: 6 VAL: 0
        POS: 7 VAL: 6
        POS: 8 VAL: 6
        POS: 9 VAL: 8
        POS: 10 VAL: -1
        POS: 11 VAL: -3
        POS: 12 VAL: 11
        POS: 13 VAL: 11
        POS: 14 VAL: 11
        POS: 15 VAL: -1
        POS: 16 VAL: -2
        POS: 17 VAL: 16
<--- SET

TREE SETS --->
        SET --->
          0
            1
            2
              3
            6
              7
              8
                9
        <--- SET

        SET --->
          4
            5
        <--- SET

        SET --->
          10
        <--- SET
```

```
        SET --->
          11
            12
            13
            14
        <--- SET

        SET --->
          15
        <--- SET

        SET --->
          16
            17
        <--- SET

<--- TREE SETS
```

8.4 – (3 points) A partir des mêmes ensembles initiaux du tableau Q, remplissez le tableau U avec le résultat de l'opération « union(14, 5) »

ATTENTION : ce n'est pas nécessaire de remplir les cases qui n'ont pas changé

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau U																			

SET --->
POS: 0 VAL: -4
POS: 1 VAL: 0
POS: 2 VAL: 0
POS: 3 VAL: 2
POS: 4 VAL: 11
POS: 5 VAL: 4
POS: 6 VAL: 0
POS: 7 VAL: 6
POS: 8 VAL: 6
POS: 9 VAL: 8
POS: 10 VAL: -1
POS: 11 VAL: -3
POS: 12 VAL: 11
POS: 13 VAL: 11
POS: 14 VAL: 11
POS: 15 VAL: -1
POS: 16 VAL: -2
POS: 17 VAL: 16
<--- SET

TREE SETS --->
SET --->
0
1
2
3
6
7
8
9
<--- SET

SET --->
10
<--- SET

SET --->
11
4
5
12
13
14
<--- SET

SET --->
15
<--- SET

SET --->
16
17
<--- SET

<--- TREE SETS

8.5 – (3 points) A partir des mêmes ensembles initiaux du tableau Q, remplissez le tableau V avec le résultat de l'opération « union (12, 15) »

ATTENTION : ce n'est pas nécessaire de remplir les cases qui n'ont pas changé

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Valeurs																			
Tableau V																			

SET --->
POS: 0 VAL: -4
POS: 1 VAL: 0
POS: 2 VAL: 0
POS: 3 VAL: 2
POS: 4 VAL: -2
POS: 5 VAL: 4
POS: 6 VAL: 0
POS: 7 VAL: 6
POS: 8 VAL: 6
POS: 9 VAL: 8
POS: 10 VAL: -1
POS: 11 VAL: -3
POS: 12 VAL: 11
POS: 13 VAL: 11
POS: 14 VAL: 13
POS: 15 VAL: 11
POS: 16 VAL: -2
POS: 17 VAL: 16
<--- SET

TREE SETS --->
SET --->
0
1
2
3
6
7
8
9
<--- SET

SET --->
4
5
<--- SET

SET --->
10
<--- SET

SET --->
11
12
13
14
15
<--- SET

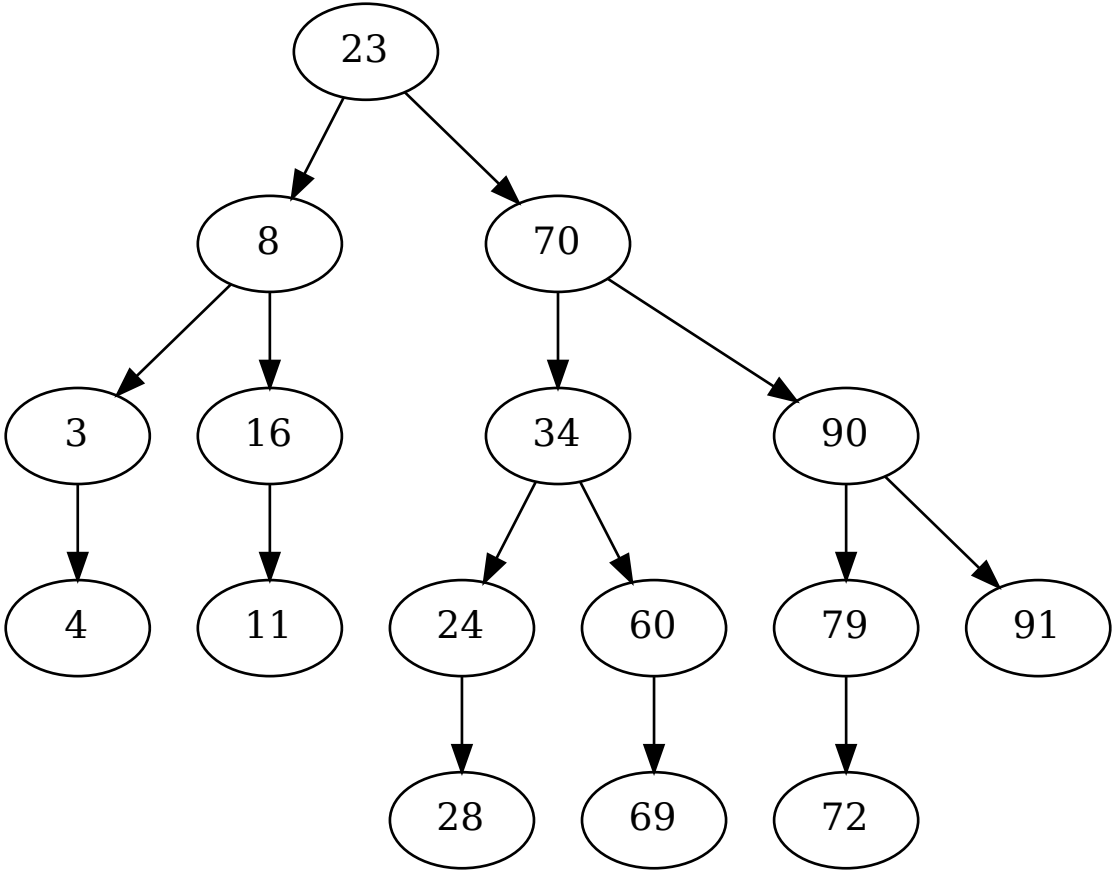
SET --->
16
17
<--- SET

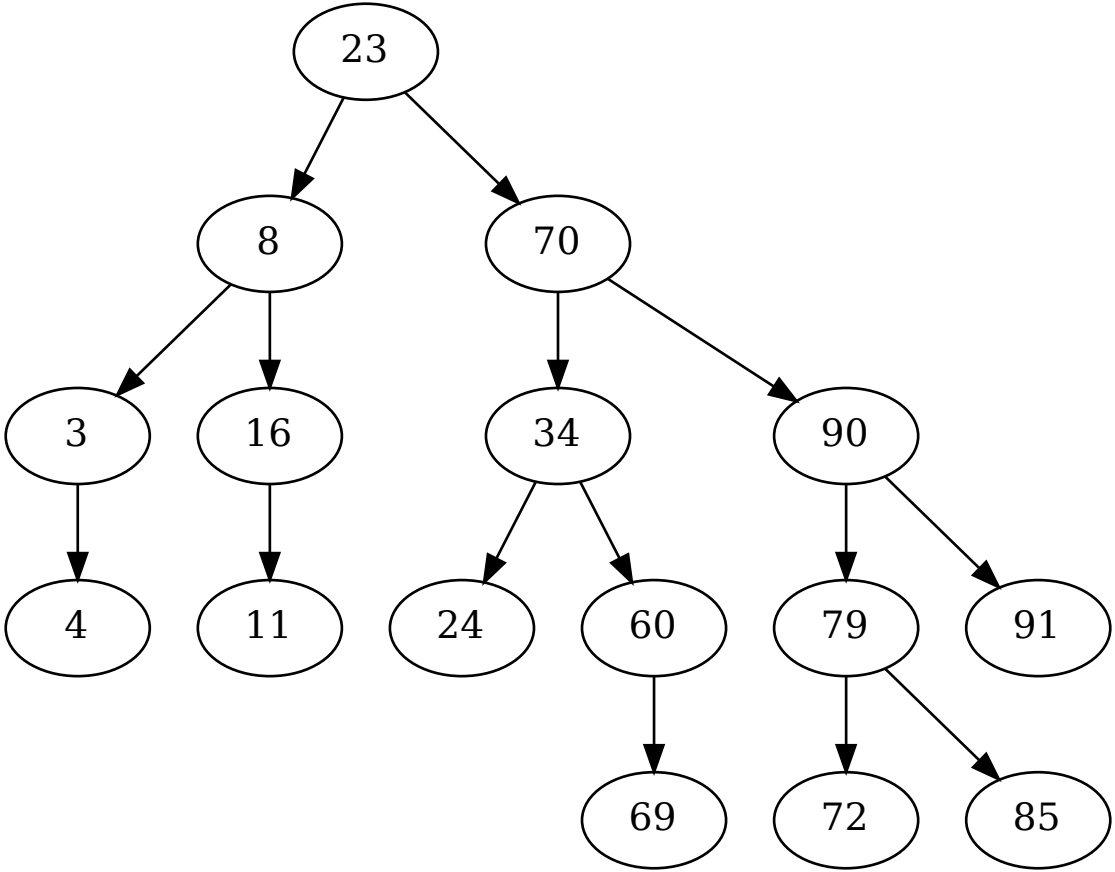
<--- TREE SETS

Annexe 1

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
 1  time = time + 1          // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]    // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK          // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```


23
8
70
3
16
34
90
4
11
24
60
79
91
28
69
72


23
8
70
3
16
34
90
4
11
24
60
79
91
69
72
85