UQÀM Université du Québec à Montréal

INF-3105, Structures de données et algorithmes
— 2009

Examen final
— 2009

CONSIGNES

· **Les règlements de l'UQAM concernant le plagiat seront strictement appliqués.**

· Aucune sortie n'est permise durant l'examen.

· Il est important de bien expliquer vos choix s'il y a lieu.

· Aucun document n'est permis.

· La durée de l'examen est de 3 heures.

· Vous pouvez utiliser les versos comme brouillon ou comme espace supplémentaire.

· **Il est interdit de dégrafer le questionnaire.**

· Les téléphones cellulaires, calculatrices, ordinateurs, palm, baladeurs, iPods, etc. sont interdits.

IDENTIFICATION

NOM : _____

PRÉNOM : _____

CODE PERMANENT : _____

SIGNATURE : _____

GROUPE : _____

PROFESSEUR : _____

#1 _____ / 0

#2 _____ / 0

#3 _____ / 0

#4 _____ / 0

#5 _____ / 0

#6 _____ / 0

#7 _____ / 0

TOTAL

_____ / 0

commentaire :

Numéro 1. (0 pts)
Objectif(s) :

- **Synthèse de la matière.**
- **Algorithme de Prim.**

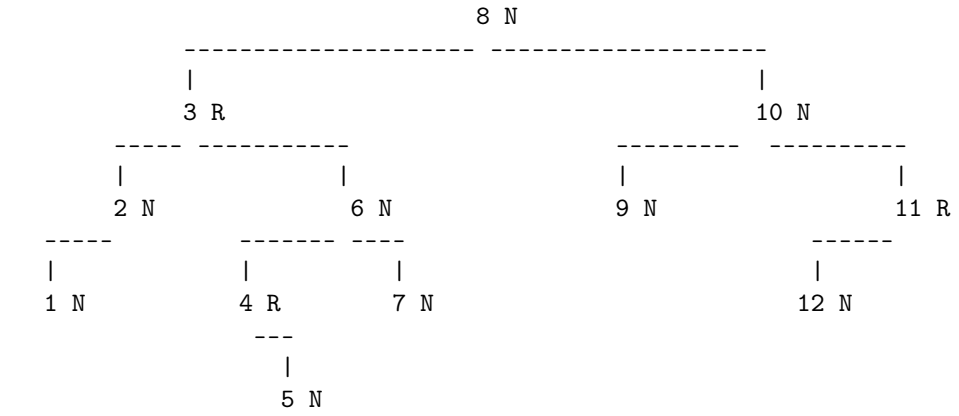
Question : Expliquez le fonctionnement de l'algorithme de Prim.
Réponse :

L'algorithme de Prim construit un arbre en ajoutant des arcs, un par un. Il démarre avec un noeud du graphe comme arbre initial, peu importe lequel. Ensuite, les arcs sont ajoutés dans une boucle. Un arc est choisi de telle sorte a ce qu'il relie un noeud déjà dans l'arbre avec un noeud non présent dans l'arbre. Parmi tous les arcs possibles, c'est l'arc ayant un poids minimum qui est choisi. Cette opération est répétée jusqu'à ce que tous les noeuds soient dans l'arbre.

Numéro 2. (0 pts)
Objectif(s) :

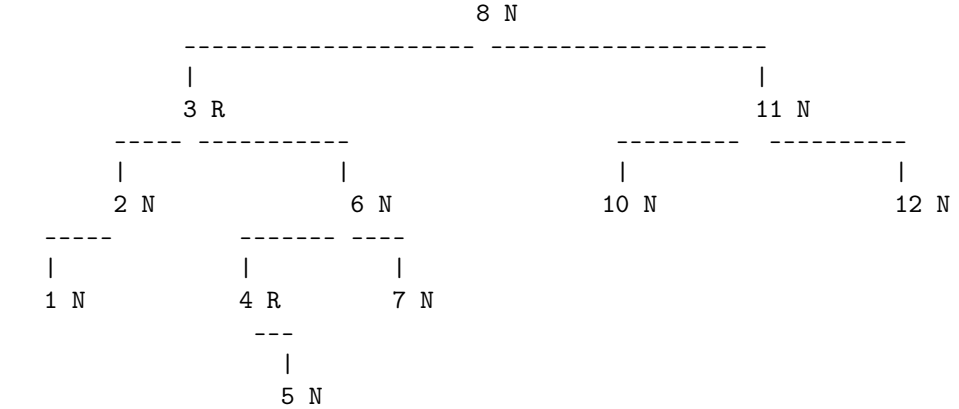
- **Application des connaissances.**
- **Arbre RN.**

Question : Soit l'arbre AVL suivant :



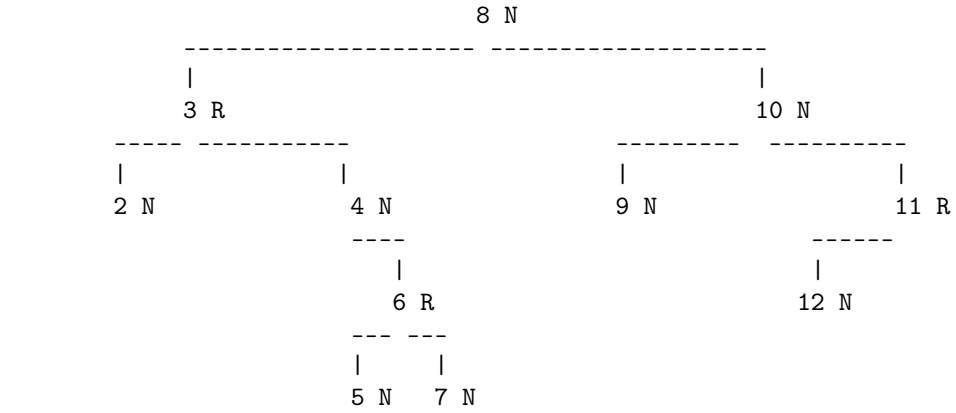
a) (0 pts) Supprimez l'élément 9 et donnez l'arbre résultant.

cas 1

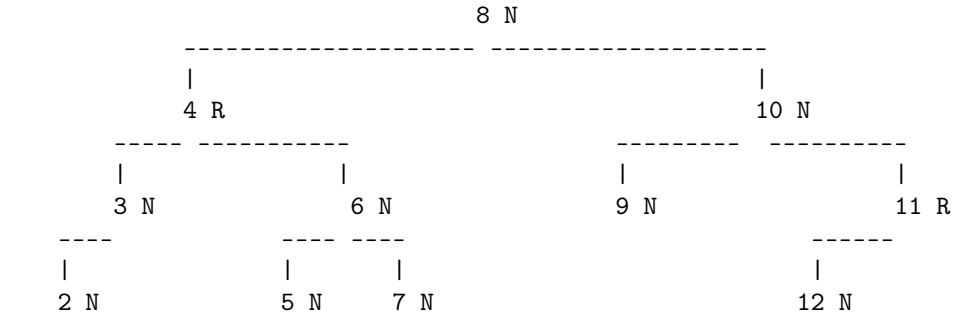


b) (0 pts) Dans l'arbre de base, avant la suppression de 9, supprimez l'élément 1.

cas 3



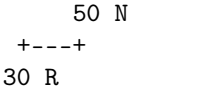
cas 4

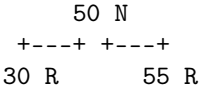


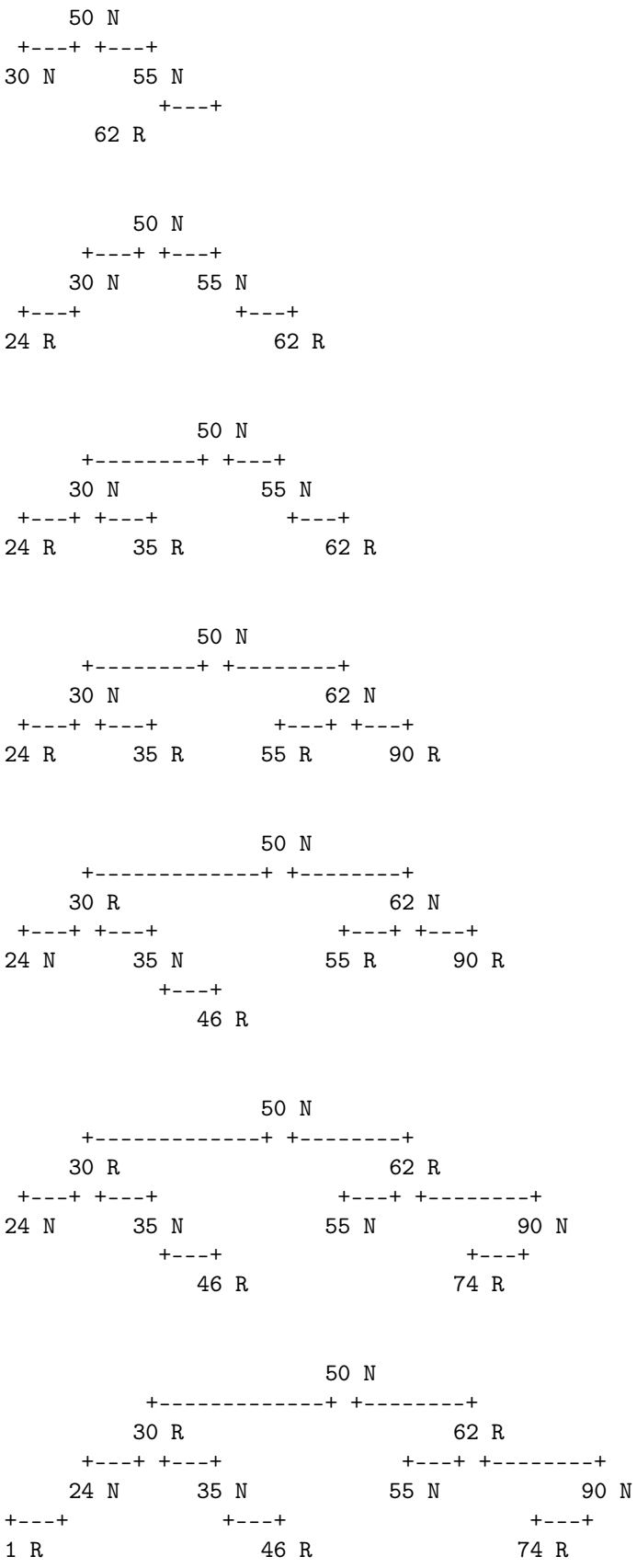
Numéro 3. (0 pts)
Objectif(s) :

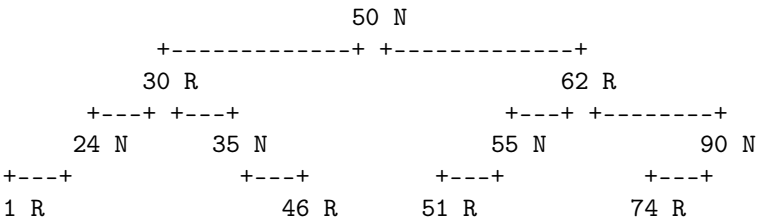
- **Application des connaissances.**
- **Arbre rouge-noir.**

Question : Construisez l'arbre rouge-noir qui résulte de l'ajout des valeurs suivantes dans un arbre vide. Les valeurs sont ajoutées dans l'ordre où elles sont présentées.
50 30 55 62 24 35 90 46 74 94 1 24 51






50 N
30 N
55 N
62 R
50 N
30 N
55 N
24 R
62 R
50 N
30 N
55 N
24 R
35 R
62 R
50 N
30 N
62 N
24 R
35 R
55 R
90 R
50 N
30 R
62 N
24 N
35 N
55 R
90 R
46 R
50 N
30 R
62 R
24 N
35 N
55 N
90 N
46 R
74 R
50 N
30 R
62 R
24 N
35 N
55 N
90 N
1 R
46 R
74 R



Numéro 4. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Algorithme de Floyd.**

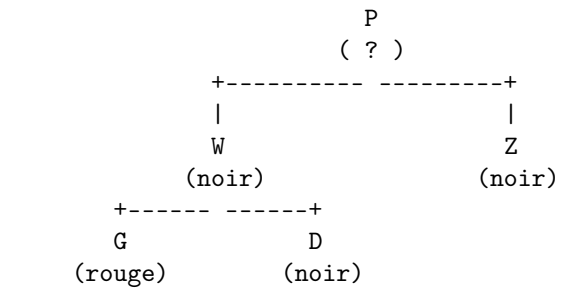
Question : Expliquez précisément le rôle que peut prendre un monceau dans l'algorithme de Kruskal.
Réponse :

L'algorithme demande de parcourir les arcs en ordre croissant. En plaçant les arcs dans un monceau, nous pouvons prendre le minimum a chaque fois. Puisque l'algorithme peut terminer sans parcourir tout les arcs, le monceau nous donne de meilleures performances qu'un trie sur les arcs.

Numéro 5. (0 pts)
Objectif(s) :

- **Analyse de problème.**
- **Arbre Rouge-Noir.**

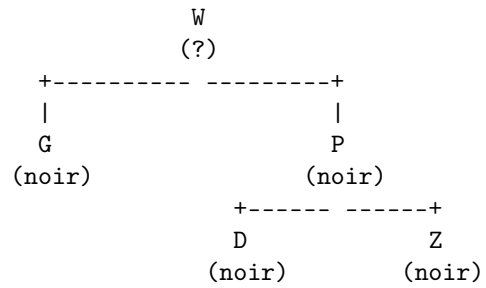
Soit le quatrième cas de la suppression dans les arbres rouge-noir :



où P, W, G, D et Z sont des noeuds. Les couleurs sont entre parenthèses. La réduction à eux lieu à droite de P.

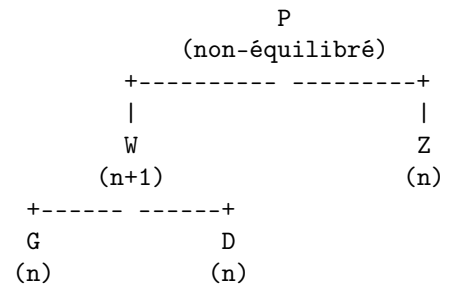
a) (0 pts) Décrivez les modifications à faire pour rééquilibrer l'arbre.
Réponse :

1 - W.couleur = P.couleur
2 - P.couleur = noir
3 - G.couleur = noir
4 - rotation à droite sur le noeud P



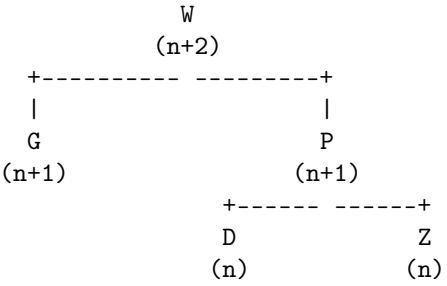
b) (0 pts) Expliquez le changement de nombre de noeuds noir. Décrivez les différents nombres de noeud noir de chaque chemin des sous-arbres et noeuds avant et après modifications.
Réponse :

avant :



Réponse :

après :



Numéro 6. (0 pts)
Objectif(s) :

- **Application des connaissances.**
- **Représentation des graphes.**
- **stl.**

Soit les déclarations suivantes :

```
#include<string>
#include<map>
#include<vector>

using namespace std;

template< typename T >
```

```
class Sommet {
public :
  string identification;
  T information;
};

template< typename T >
class ListeAdjacence {
private :
  vector< Sommet< T > > _liste;
public :
  void ajouterElement( const Sommet< T > & a_sommet );
  bool appartient( const Sommet< T > & a_sommet ) const;
};

template< typename T >
class Graphe {
private :
  map< Sommet< T >, ListeAdjacence< T > > description;
public :
  void ajouterArc( const Sommet< T > & a_sommet1,
                   const Sommet< T > & a_sommet2 );
  vector< Sommet< T > > voisins( const Sommet< T > & a_sommet ) const;
  bool sontVoisins( const Sommet< T > & a_sommet1,
                    const Sommet< T > & a_sommet2 );
};
```

a) (0 pts) Écrivez le code pour la fonction `ListeAdjacence::appartient`.

```
bool ListeAdjacence::appartient( const Sommet< T > & a_sommet ) const {
    bool r = false;
    for( Sommet< T > & s : _liste ) {
        r = r || s == a_sommet;
    }
    return r;
}
```

b) (0 pts) Écrivez le code pour la fonction `Graphe::sontVoisins`.

```
bool Graphe::sontVoisins( const Sommet< T > & a_sommet1,
                          const Sommet< T > & a_sommet2 ) {
    return (* (description.find( a_sommet1 ))).appartient( a_sommet2 );
}
```

Numéro 7. (0 pts)
Objectif(s) :

- **Synthèse de la matière.**
- **Monceau.**

a) (0 pts) Expliquez comment un monceau descendant(maximum en tête) est représenté. Donnez les relations entre pères et fils.

Réponse :

Les fils vont avoir des valeurs plus petites ou égales au père.

b) (0 pts) Expliquez comment un élément est ajouté dans un monceau et donnez l'ordre temporel de l'algorithme.
Réponse :

L'élément est placé à la dernière case du tableau et est percolé vers sa position. C'est un ordre O(log n).

c) (0 pts) Expliquez comment un élément est enlevé dans un monceau et donnez l'ordre temporel de l'algorithme.
Réponse :

La racine est supprimée et remplacée par le dernier élément du tableau. Ensuite, l'élément qui remplace la racine est tamisé vers sa position. C'est un algorithme O(log n).