Université du Québec à Trois-Rivières
Savoir. Surprendre.

RAPPORT TP1 INF008- Analyse et conception d'algorithmes

Présenter par :

- Jiadong Jin
- Yujie Lu
- Marianne Balde
- Ramatoulaye Barry
- Maty Mbow

Présenter à :

Ramla Ghali

SESSION Hiver 2023

I. Introduction

Notre labyrinthe utilise des idées de programmation orientée objet pour stocker les classes de nœuds et d'arêtes. En outre, nous avons optimisé la présentation du labyrinthe pour le rendre plus intuitif. Une telle structure de stockage nous permet de présenter l'algorithme de Prim de manière plus intuitive.

II. Problèmes et difficultés rencontrés

Pour décider de la structure à utiliser pour stocker le labyrinthe, nous avons considéré à la fois la matrice et la table d'adjacence, qui, comme décrit dans le problème, comporte une grande quantité de redondance spatiale. Nous avons donc choisi la table d'adjacence, mais il n'existe pas de structure de données en C# qui puisse représenter directement une table d'adjacence. Nous avons donc utilisé deux classes différentes pour composer le labyrinthe, guidés par la pensée de la conception orientée objet.

III. Justification détaillée et exemples

a) Démontrez que votre implémentation pour la partie de l'application qui exécute l'algorithme de Prim est bien de complexité algorithmique $\boldsymbol{\Theta}(n^2)$

Tout d'abord, nous ajoutons le nœud (0, 0) à la liste des nœuds qui sont passés, puis nous ajoutons tous les autres nœuds à la liste des nœuds qui ne sont pas passés. La complexité de toutes ces opérations est O(1). Nous ajoutons les arêtes connectées au nœud (0, 0) à la liste des arêtes voisines et les trions. Nous utilisons ici un algorithme de tri rapide et la complexité est O(nlogn). Toutes les autres arêtes sont ajoutées à la liste des arêtes non passées et triées du plus petit au plus grand poids. Nous démarrons ensuite la boucle et lorsque la liste des arêtes non parcourues n'est pas vide, le premier élément de la liste des arêtes voisines est ajouté à la liste des arêtes déjà parcourues [O(1)]. Ensuite, le nœud connecté à l'arête qui vient d'être ajoutée à la liste des arêtes déjà passées est ajouté à la liste des nœuds déjà passés. La complexité de cette étape est O(n) en raison de la nécessité d'affiner la validité des nœuds ajoutés. Le nœud qui vient d'être rejoint est ensuite retiré de la liste des nœuds non passés [O(n)]. Le nœud qui vient d'être rejoint est connecté à quatre arêtes au maximum. Nous ajoutons donc toutes les arêtes connectées à ce nœud à la liste des arêtes voisines dans l'ordre [O(n)]. Enfin, nous mettons fin à la boucle en supprimant de la liste des arêtes adjacentes

toutes les arêtes dupliquées ou les arêtes qui formeraient un cycle. Étant donné que la complexité la plus élevée du cycle est O(n), la complexité de l'ensemble de l'algorithme Prim est $O(n^2)$.

```
public List<Arc> Prim()
{
    List<Node> nodeparsed = new() { nodes[0] };//Déjà passé le noeud initial (0, 0)
    List<Node> nodeleft = nodes.Skip(1).ToList();//A l'exception du noeud initial (0, 0), les autres noeuds n'ont pas encore passé
    List<Arc> arcparsed = new();//Toutes les arêtes ne sont pas encore passées
    //Classer toutes les arêtes adjacentes aux noeuds qui sont déjà passés dans l'ordre croissant de leur poids.
    List<Arc> arcadj = nodes[0].Relates.OrderBy(n => n.Weight).ToList();//la méthode OrderBy utilise un algorithme de tri rapide O(n*logn)
    while (nodeleft.Count > 0)
    {
        arcparsed.Add(arcadj[0]);//Sélectionner la plus petite arête adjacente O(1)
        nodeparsed.Add(arcparsed.Last().Relates.Single(n => nodeleft.Contains(n)));//Passer le noeud adjacent à l'arête O(n)
        nodeleft.Remove(nodeparsed.Last());//Le noeud a été passé O(n)
        //Ajouter les arêtes adjacentes du noeud à la liste des arêtes adjacentes par ordre.
        foreach (Arc a in nodeparsed.Last().Relates)
        {//Nombre constant de cycles avec un maximum de 4
            var index = arcadj.BinarySearch(a);//O(logn)
            if (index < 0) index = ~index;
            arcadj.Insert(index, a);//O(n)
        }
        //Supprimer toutes les arêtes qui rejoignent des noeuds qui ont déjà été passés.
        arcadj.RemoveAll(n => nodeparsed.Contains(n.Relates[0]) && nodeparsed.Contains(n.Relates[1]));//O(n)
    }
    return arcparsed;
}
```

b) Quelle est votre complexité algorithmique pour la lecture/le balayage de votre labyrinthe : **$O(n^2)$**

```
public void Affiche()
{
    int pointer = 0;
    int x = nodes.Max(n => n.X) + 1;
    int y = nodes.Max(n => n.Y) + 1;
    for (int i = 0; i < x; i++)//O(n)
    {
        string stringbuffer = "";
        for (int j = 0; j < y; j++)//O(n)*O(n)
        {
            stringbuffer += String.Format("{0,4},{1,-4:N0}", nodes[pointer].X, nodes[pointer].Y);
            Complexity.Counter++;
            if (j < y - 1)
            {
                int w = nodes[pointer].Relates.First(n => n.Orien.Equals(true)).Weight;
                stringbuffer += String.Format("{2,2}---{0,3}{1,2}---{3,2}", w, "", "", "");//O(n^2)
                Complexity.Counter++;
            }
            pointer++;
        }
        Console.WriteLine(stringbuffer);

        if (i < x - 1)
        {
            for (int k = 0; k < 7; k++)
            {
                stringbuffer = "";
                string w = "|";
                if (k == 0 || k == 2 || k == 4 || k == 6) w = "";
                for (int j = 0; j < y; j++)//O(n)*O(n)
                {
                    if (k == 3) w = nodes[pointer + j].Relates.Last(n => n.Orien.Equals(false)).Weight.ToString();
                    stringbuffer += String.Format("{0,5}{1,4}", w, "");
                    Complexity.Counter++;
                    if (j < y - 1)
                    {
                        stringbuffer += String.Format("{0,8}{1,7}", "", "");
                        Complexity.Counter++;
                    }
                }

                Console.WriteLine(stringbuffer);
            }
        }
    }
    Console.WriteLine("\r\nLe labyrinthe a été généré avec succès");
}
```

c) Quelle est la complexité générale de tout le processus de génération du labyrinthe (incluant l'initialisation, la génération aléatoire des poids pour les arêtes, l'exécution de la matrice de Prim et toute autre instruction reliée)

O(n^2) En supposant qu'il y ait m ligne et n colonne dans le labyrinthe, la fréquence de la première instruction initiale est m*n. Donc la complexité d'initialisions c'est O(n^2). L'attribution de poids aléatoires aux arêtes se fait également en parcourant toutes les lignes et toutes les colonnes. La complexité de cette étape est donc la suivante O(n^2). Aussi la complexité d'affichage est O(n^2).Ainsi, la complexité temporelle totale de l'algorithme PRIM est O(n^2).Toutes les étapes ci-dessus sont relativement indépendantes et ne sont pas imbriquées dans des boucles, de sorte que la complexité totale de l'ensemble du programme est de **O(n^2).**

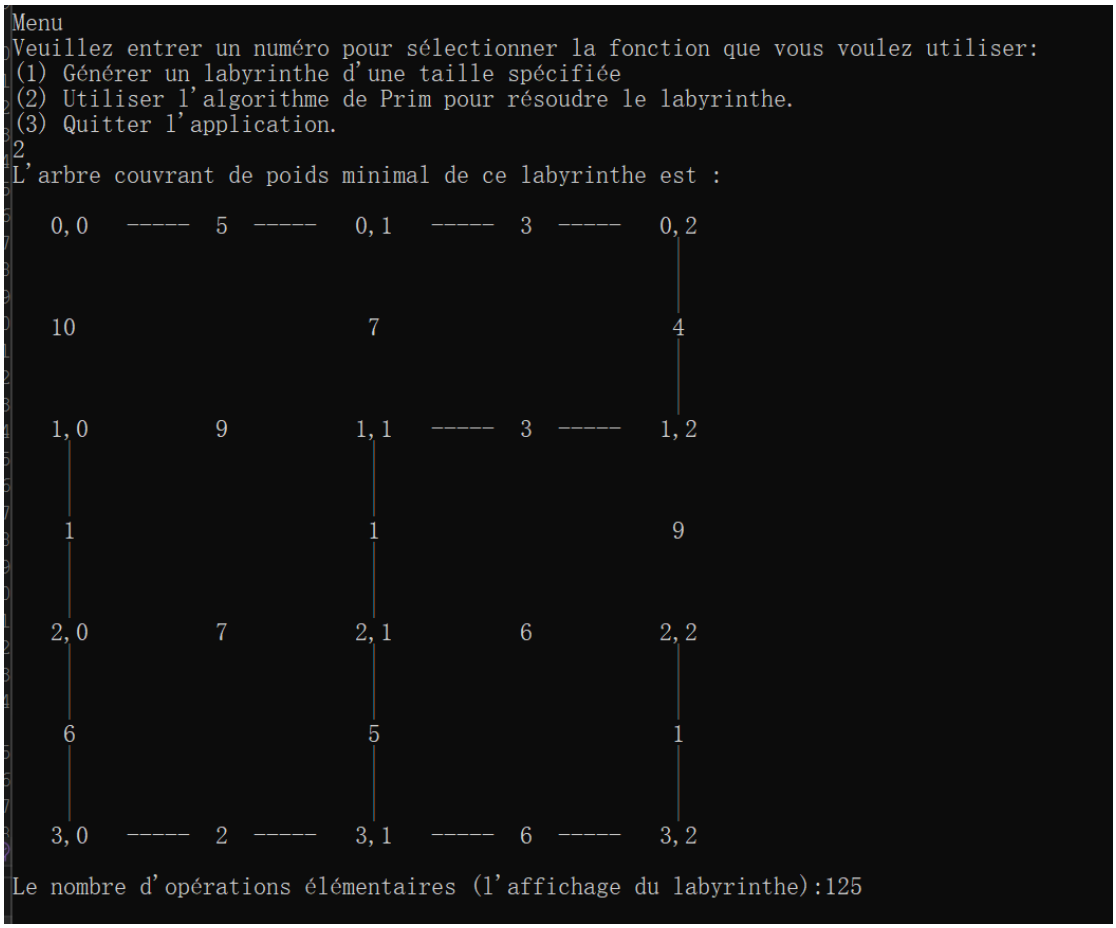
Présentation des résultats：

```
Menu
Veuillez entrer un numéro pour sélectionner la fonction que vous voulez utiliser:
(1) Générer un labyrinthe d'une taille spécifiée
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
1
Veuillez entrer le nombre de colonnes du labyrinthe.:
3

Veuillez entrer le nombre de lignes du labyrinthe.:
4
Le nombre d'opérations élémentaires (l'initialisation de la liste d'adjacence):12
Le nombre d'opérations élémentaires (la génération du labyrinthe):51

   0,0     ---  5  ---     0,1     ---  3  ---     0,2
    |                       |                       |
   10                       7                       4
    |                       |                       |
   1,0     ---  9  ---     1,1     ---  3  ---     1,2
    |                       |                       |
    1                       1                       9
    |                       |                       |
   2,0     ---  7  ---     2,1     ---  6  ---     2,2
    |                       |                       |
    6                       5                       1
    |                       |                       |
   3,0     ---  2  ---     3,1     ---  6  ---     3,2

Le labyrinthe a été généré avec succès
Le nombre d'opérations élémentaires (l'affichage du labyrinthe):125
```


Menu
Veuillez entrer un numéro pour sélectionner la fonction que vous voulez utiliser:
(1) Générer un labyrinthe d'une taille spécifiée
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
2
L'arbre couvrant de poids minimal de ce labyrinthe est :
0,0 ----- 5 ----- 0,1 ----- 3 ----- 0,2
10 7 4
1,0 9 1,1 ----- 3 ----- 1,2
1 1 9
2,0 7 2,1 6 2,2
6 5 1
3,0 ----- 2 ----- 3,1 ----- 6 ----- 3,2
Le nombre d'opérations élémentaires (l'affichage du labyrinthe):125

```
Menu
Veuillez entrer un numéro pour sélectionner la fonction que vous voulez utiliser:
(1) Générer un labyrinthe d'une taille spécifiée
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
1
Veuillez entrer le nombre de colonnes du labyrinthe.:
4

Veuillez entrer le nombre de lignes du labyrinthe.:
5
Le nombre d'opérations élémentaires (l'initialisation de la liste d'adjacence):20
Le nombre d'opérations élémentaires (la génération du labyrinthe):93

   0,0     ---  5  ---     0,1     ---  2  ---     0,2     ---  2  ---     0,3
    |                       |                       |                       |
    6                       5                       8                       5
    |                       |                       |                       |
   1,0     ---  6  ---     1,1     ---  4  ---     1,2     --- 10  ---     1,3
    |                       |                       |                       |
    2                       9                       9                       4
    |                       |                       |                       |
   2,0     ---  9  ---     2,1     ---  5  ---     2,2     --- 10  ---     2,3
    |                       |                       |                       |
    8                       5                       8                       9
    |                       |                       |                       |
   3,0     ---  4  ---     3,1     ---  5  ---     3,2     ---  6  ---     3,3
    |                       |                       |                       |
    8                       1                      10                       7
    |                       |                       |                       |
   4,0     ---  5  ---     4,1     ---  5  ---     4,2     ---  9  ---     4,3

Le labyrinthe a été généré avec succès
Le nombre d'opérations élémentaires (l'affichage du labyrinthe):231
```

```
Menu
Veuillez entrer un numéro pour sélectionner la fonction que vous voulez utiliser:
(1) Générer un labyrinthe d'une taille spécifiée
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
2
L'arbre couvrant de poids minimal de ce labyrinthe est :

   0,0   -----  5  -----   0,1   -----  2  -----   0,2   -----  2  -----   0,3
                            |                                              |
    6                       5                       8                      5
                            |                                              |
   1,0   -----  6  -----   1,1   -----  4  -----   1,2         10         1,3
    |                                                                      |
    2                       9                       9                      4
    |                                                                      |
   2,0          9          2,1   -----  5  -----   2,2         10         2,3
    |                       |                                              |
    8                       5                       8                      9
    |                       |                                              |
   3,0   -----  4  -----   3,1   -----  5  -----   3,2   -----  6  -----   3,3
                            |                                              |
    8                       1                      10                      7
                            |                                              |
   4,0   -----  5  -----   4,1   -----  5  -----   4,2          9         4,3

Le nombre d'opérations élémentaires (l'affichage du labyrinthe):231
```

```
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
1
Veuillez entrer le nombre de colonnes du labyrinthe.:
7

Veuillez entrer le nombre de lignes du labyrinthe.:
6
Le nombre d'opérations élémentaires (l'initialisation de la liste d'adjacence):42
Le nombre d'opérations élémentaires (la génération du labyrinthe):213

   0,0     --- 9 ---     0,1     --- 2 ---     0,2     --- 10 ---     0,3     --- 6 ---     0,4     --- 7 ---     0,5     --- 3 ---     0,6
    |                     |                     |                     |                     |                     |                     |
    8                     5                     6                     10                    10                    1                     7
    |                     |                     |                     |                     |                     |                     |
   1,0     --- 9 ---     1,1     --- 9 ---     1,2     --- 6 ---     1,3     --- 2 ---     1,4     --- 6 ---     1,5     --- 7 ---     1,6
    |                     |                     |                     |                     |                     |                     |
    9                     8                     10                    6                     1                     5                     9
    |                     |                     |                     |                     |                     |                     |
   2,0     --- 7 ---     2,1     --- 2 ---     2,2     --- 3 ---     2,3     --- 8 ---     2,4     --- 9 ---     2,5     --- 9 ---     2,6
    |                     |                     |                     |                     |                     |                     |
    9                     8                     5                     10                    5                     8                     8
    |                     |                     |                     |                     |                     |                     |
   3,0     --- 3 ---     3,1     --- 3 ---     3,2     --- 8 ---     3,3     --- 10 ---    3,4     --- 8 ---     3,5     --- 2 ---     3,6
    |                     |                     |                     |                     |                     |                     |
    4                     10                    3                     8                     9                     7                     1
    |                     |                     |                     |                     |                     |                     |
   4,0     --- 8 ---     4,1     --- 1 ---     4,2     --- 1 ---     4,3     --- 10 ---    4,4     --- 6 ---     4,5     --- 3 ---     4,6
    |                     |                     |                     |                     |                     |                     |
    1                     3                     9                     6                     1                     4                     8
    |                     |                     |                     |                     |                     |                     |
   5,0     --- 10 ---    5,1     --- 1 ---     5,2     --- 10 ---    5,3     --- 10 ---    5,4     --- 9 ---     5,5     --- 9 ---     5,6

Le labyrinthe a été généré avec succès
Le nombre d'opérations élémentaires (l'affichage du labyrinthe):533
```

```
Menu
Veuillez entrer un numéro pour sélectionner la fonction que vous voulez utiliser:
(1) Générer un labyrinthe d'une taille spécifiée
(2) Utiliser l'algorithme de Prim pour résoudre le labyrinthe.
(3) Quitter l'application.
2
L'arbre couvrant de poids minimal de ce labyrinthe est :

   0,0          9          0,1     -----  2  -----     0,2          10         0,3     -----  6  -----     0,4     -----  7  -----     0,5     -----  3  -----     0,6
    |                       |                         |                       |                         |                       |                       |
    8                       5                         6                       10                        10                      1                       7
    |                       |                         |                       |                         |                       |                       |
   1,0          9          1,1          9            1,2     -----  6  -----     1,3     -----  2  -----     1,4     -----  6  -----     1,5          7          1,6
    |                       |                         |                       |                         |                       |                       |
    9                       8                         10                      6                         1                       5                       9
    |                       |                         |                       |                         |                       |                       |
   2,0     -----  7  -----    2,1     -----  2  -----     2,2     -----  3  -----     2,3          8          2,4          9          2,5          9          2,6
    |                       |                         |                       |                         |                       |                       |
    9                       8                         5                       10                        5                       8                       8
    |                       |                         |                       |                         |                       |                       |
   3,0     -----  3  -----    3,1     -----  3  -----     3,2          8          3,3          10         3,4     -----  8  -----     3,5     -----  2  -----     3,6
    |                       |                         |                       |                         |                       |                       |
    4                       10                        3                       8                         9                       7                       1
    |                       |                         |                       |                         |                       |                       |
   4,0          8          4,1     -----  1  -----     4,2     -----  1  -----     4,3          10         4,4     -----  6  -----     4,5     -----  3  -----     4,6
    |                       |                         |                       |                         |                       |                       |
    1                       3                         9                       6                         1                       4                       8
    |                       |                         |                       |                         |                       |                       |
   5,0          10         5,1     -----  1  -----     5,2          10         5,3          10         5,4          9          5,5          9          5,6

Le nombre d'opérations élémentaires (l'affichage du labyrinthe):533
```