IN403 - Tables de routage

RAMAROMANANATOANDRO Thomas TD2

Mai 2019

1 – Les structures de données choisies

Stocker le graphe correspondant au réseau

Structure pour la reconstitution du chemin

2 – Les fonctions / algorithmes utilisés

Vérifier la connexité du réseau

Déterminer la table de routage de chaque noeud

1.) Les structures de données choisies:

1.1 Stocker le graphe du réseau

Le projet est structuré en 1 fichier .c : graphe.c .

La principale structure de données est la structure graphe.
Celle-ci regroupe: le nombre de sommets du graphe, la représentation du graphe par une matrice d'adjacence, la matrice des temps qui stocke les temps de chaque arêtes,la matrice des prédecesseurs et la matrice modifiable pour l'algorithme de parcours de plus court chemin,le tableau couleur pour effectué le parcours en profondeur et également les entiers depart et destination qui stockeront les noeuds saisis par l'utilisateur pour la reconstitution du plus court chemin.

```
//structure du graphe
typedef struct
{
  int nb_sommets;       //nombre de sommets
  int **matrice_adj;        //matrice adjacence
  int **temps;          // matrice des poids
  int *couleur;         //tableau pour parcours profondeur
  int depart;           //depart du routage
  int destination;      //destination du routage
  int **P;          //matrice des predecesseurs
  int **W;          //matrice modifiée pour Floyd Warshall
} graphe;
```

1.2 La structure pour reconstituer le chemin

Afin de reconstituer le chemin que doit prendre l'utilisateur pour aller d'un noeud émetteur à un noeud destinataire , j'empile les noeuds mis dans la matrice des prédecesseurs (P) lors de l'appel à la fonction de calcul des tables de routages de chaque noeud. Ainsi, pour reconstituer le chemin dans le bon sens entre les deux noeuds , je parcours la pile.

```
typedef struct Element Element;
struct Element
{
  int nombre;
  Element *suivant;
};

typedef struct Pile Pile;
struct Pile
{
  Element *premier;
};
```

2.) Les fonctions/algorithmes utilisés

2.1 Vérifier la connexité du réseau

La vérification de la connexité du réseau a été réalisé à l'aide de l'algorithme de coloriage des composantes connexes vu au TD3 (voir ci-dessous).
On colorie chaque noeud du graphe d'une même couleur tant qu'elle fait partie de la même composante connexe,si une seule couleur est utilisé cela signifie que le graphe est bien connexe, dans le cas contraire le graphe n'est pas connexe et on relance une création de réseau.

Colorier(Graphe G, Sommet s,Couleur c)
couleur[s] = c
Pour chaque sommet v, voisin de s, faire
Si (couleur[v] ==0)alors
Colorier(G,v,c)

Composantes_ connexes(Graphe G)
couleur = 1
Pour chaque sommet de x de v faire
couleur[x] = 0
Pour chaque sommet de x de v faire
Si couleur[x] == 0
Colorier(G,x,couleur)
couleur = couleur+1

2.2 Déterminer la table de routage de chaque noeud

Pour déterminer la table de routage de chaque noeud j'utilise l'algorithme de Floyd-Warshall.
Celui-ci permet de déterminer à l'aide des tableaux W et P le plus court chemin d'un noeud à tous les autres (all to all). Dans le tableau W on stockera les temps minimaux pour chaque noeud et dans P on stockera les prédecesseurs de chaque noeud.

La fonction Floyd Warshall a été implémenter à l'aide de ces algorithmes:

```
// M : matrice des plus courts chemins
// P : matrice des prédécesseurs pour les plus courts chemins
Initialiser M à +∞
Initialiser P à 0
pour i allant de 1 à N faire
    M_{i,i} ← 0
    P_{i,i} ← i
    pour tout successeur j de i faire
        M_{i,j} ← W[i,j]
// Calcul des matrices successives
pour k allant de 1 à N faire
    pour i allant de 1 à N faire
        pour j allant de 1 à N faire
            si M_{i,k} + M_{k,j} < M_{i,j} alors
                M_{i,j} = M_{i,k} + M_{k,j}
                P_{i,j} = P_{k,j}
si ∃i|M_{i,i} < 0 alors
    retourner Il existe un circuit de longueur négative passant par i
sinon
    retourner M
```

Le graphe est représenté sous forme d'une matrice d'adjacence avec :

- $M[i,j]$ = la valeur de l'arc (i,j) si cet arc existe.
- $M[i,j] = +\infty$ sinon.

Les arcs du graphe peuvent avoir des poids négatifs. L'algo, très simple est le suivant :

- $W = M$ (pour garder la matrice d'adjacence intacte)
- Pour k allant de 1 à n faire
  - Pour i allant de 1 à n faire
    - Pour j allant de 1 à n faire
    - $W[i,j] = Min(W[i,j], W[i,k] + W[k,j])$

Bis) Commentaires: Une fonction appelée auxiliaire permet de créer un fichier temps.txt décrivant les temps valuant chaque liens entre noeuds du réseau. Le réseau est représenté à l'aide de l'utilitaire graphviz et plus précisément sfdp.