

L3-II – TP

Graphes

Objet du TP 5 : Algorithme de Prim

1 Réticulum endoplasmique

Le réticulum endoplasmique est un réseau reliant différents points à la surface des cellules vivantes servant au transport vers l'intérieur de la cellule de nutriments, minéraux et autres ... La figure 1 montre un tel réseau au microscope. Ce réseau est dynamique et se construit en garantissant la connectivité entre tous les points d'entrée dans la cellule. Les biologistes ont tenté de modéliser sa construction en reprenant l'idée que il ressemble à un arbre de recouvrement de longueur minimale.

Q1: Copier le fichier

```
/home/commun_depinfo/enseignants/lemarchand/TPGraphesII/TP5/tp5.c
```

chez vous et regarder la fonction

```
int readPositions(char fname[], int nodeX[MAXNODE], int nodeY[MAXNODE])
```

qui lit la suite des positions des noeuds associés à une image ER et les stocke dans les tableaux `nodeX[MAXNODE]` et `nodeY[MAXNODE]`.

La fonction renvoie le nombre *nodes* de positions lues. Copier également le fichier d'exemple `tp5_er.dat` du même répertoire.

Q2: Implanter la fonction

```
float computeDistance(int i, int j, int nodeX[MAXNODE], int nodeY[MAXNODE])
```

qui pour 2 indexes *i* et *j* calcule la distance euclidienne entre les sommets *i* et *j*.

Q3: Implanter la fonction

```
void fillDistances(int nodes, int nodeX[MAXNODE], int nodeY[MAXNODE], float as[MAXNODE][MAXNODE])
```

qui remplit la matrice `as` (symétrique) des distances entre les différents sommets. La distance d'un noeud à lui-même vaut 0.

Q4: Implanter l'algorithme de Prim pour faire le calcul d'un arbre de recouvrement à partir d'un sommet quelconque. La fonction a pour entête :

```
void prim(int root, int nodes, float as[MAXNODE][MAXNODE], int edgelist[MAXNODE])
```

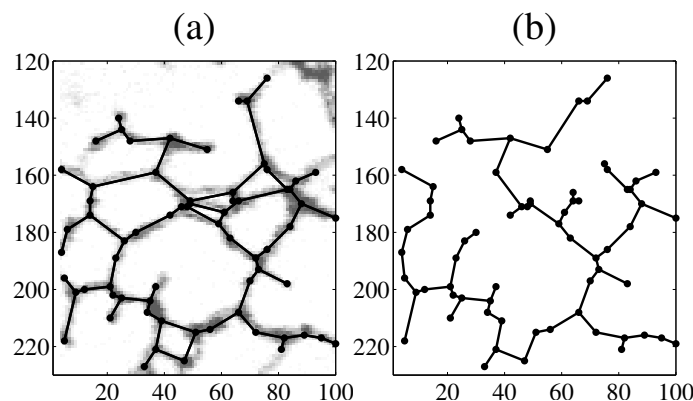


FIGURE 1 – (a) Réticulum endoplasmique, (b) Arbre de recouvrement minimum.

Pour chaque sommet $i \neq root$, si $edgelist[i] = j$, alors l'arête $i \rightarrow j$ est dans l'arbre.

Q5: Appliquer l'algorithme en utilisant les données de positionnement de `er.dat`. Puis utiliser la fonction

```
void drawGraph(int nodes, int root, int edgelist[MAXNODE], int nodeX[MAXNODE], int nodeY[MAXNODE])
```

pour dessiner l'arbre obtenu. Le tableau `edgelist[]` est la liste des arêtes de l'arbre, résultat de l'algorithme de Prim.