Université Lille1 Sciences et Technologies

ALGO

15 novembre 2010
Licence ST-A
François Lemaire

TP Graphes / plus courte distance

---

1 Plus courte chaîne

Q 1 . Ecrivez une procédure `plusCourteChaine` qui prend en entrée un graphe non orienté et un sommet de départ, et qui calcule les tableaux `d` et `pred` comme expliqué dans le polycopié.

Conseil : le plus simple est d'utiliser des paramètres `d` et `pred` de la forme `int d[MAX_SOMMETS]` et et `tNumeroSommet pred[MAX_SOMMETS]`.

Q 2 . Testez votre procédure sur un exemple simple, en vérifiant bien les valeurs de `d` et `pred`.

2 Complexité

Q 3 . Comme la librairie fournit des opérations de base en coût constant, la complexité en temps de `plusCourteChaine` devrait être en $O(m+n)$.

Nous allons générer des graphes aléatoirement avec la procédure grapheAleatoire. Cette procédure prend un nombre de sommets, ainsi qu'une probabilité d'avoir un arc entre deux sommets. Si la probabilité est 1, le graphe aura tous les arcs/arêtes possibles. Si la probabilité est 0, il n'y aura aucun arc/arête. Vous ne pouvez donc pas directement fixer la valeur m du nombre d'arcs/arêtes, mais vous pourrez consulter m une fois le graphe créé. Ce sera suffisant pour les questions qui suivent.

Confirmez expérimentalement la complexité en $O(m+n)$:

- en utilisant la fonction de génération aléatoire de graphe, testez jusque quelles valeurs de n et m votre procédure fonctionne dans un temps raisonnable. Cette approche est plus pragmatique et ne permet pas de savoir si la complexité est bien linéaire, mais c'est un bon test à effectuer en général
- (optionnel) faire varier n et m et calculer le rapport $t(n,m)/(n+m)$ pour des valeurs différentes de n et m. $t(n,m)$ est le temps d'exécution de la procédure `plusCourteChaine` pour un graphe à n sommets et m arêtes. Que s'attend-on à trouver concernant ce rapport ? Le temps peut se mesurer en C grâce à la fonction qui suit (qui renvoie le nombre de milli-secondes écoulées depuis minuit).

```
long nbMilliSecondesDepuisMinuit() {
  struct timeval tv;
  struct timezone tz;
  long ms;
  gettimeofday(&tv, &tz);
  tv.tv_sec = tv.tv_sec % (24*3600);
  ms = (tv.tv_sec*1000)+tv.tv_usec/1000;
  return ms;
}
```

3 Tracé des chemins de distance minimale

Si vous avez terminé, vous pouvez vous servir du tableau `pred` calculé pour mettre en couleur les arêtes faisant partie des chemins de distance minimale.

Pour cela, il suffit d'écrire une variante de `graphe2visu` :

```
graphe2visuPlusCourtsChemins(tGraphe graphe, char *outfile,
                             tNumeroSommet depart, tNumeroSommet pred[MAX_SOMMETS);
```

qui crée un graphe en colorant le sommet de départ (en bleu par exemple) et les arêtes des chemins minimaux en bleu.

Indice, le coloriage d'arête pour `dot` se fait en rajoutant `[color= ...]` lors de la déclaration de l'arc/arête.

```
graph {
A [color=blue];
B [color=black];
C [color=black];
D [color=black];
```

```
E [color=black];
A -- B [color=blue];
A -- C [color=blue];
B -- C [color=black];
B -- D [color=blue];
C -- D [color=black];
C -- E [color=black];
B -- E [color=blue];
}
```