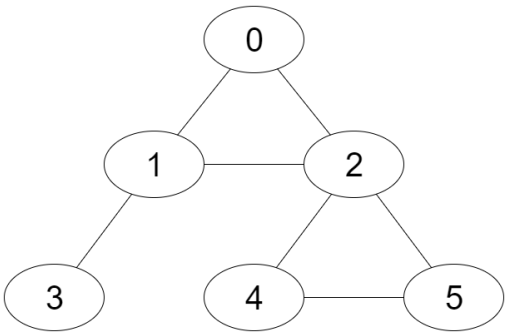
Protocoles de parcours distribués avec détection de la terminaison

Introduction

L'objectif de ce TP est d'implémenter des protocoles de parcours distribués avec détection de la terminaison. L'algorithme distribué à implémenter va effectuer la diffusion d'un message et la constructions d'un arbre de recouvrement par la même occasion.

I. Création topologie

Voici la typologie à parcourir et qui sera utilisée dans les parcours ci-dessous.



La topologie précédente sera créée à l'aide de la méthode « *MPI_Graph_create* ». Il faudra créer le tableau d'index, celui-ci s'écrit de la façon suivante :

- Le premier index correspond au nombre de nœuds voisins à ce nœud
- Le deuxième correspond au nombre de nœuds voisins à ce nœud + l'index précédent

On aura donc pour cette typologie : `graph_index[6] = {2, 5, 9, 10, 12, 14}`

Pour les arcs, on se place sur le premier nœud, on dit qui sont ses voisins, puis on continue pour le deuxième nœud et ainsi de suite. On aura donc : `graph_edges[16] = {1, 2, 0, 2, 3, 0, 1, 4, 5, 1, 2, 5, 2, 4}`

On pourra alors passer ces 2 tableaux en paramètre de la fonction pour créer le graphe.

II. Parcours en profondeur

A. Présentation

L'algorithme de parcours en profondeur est un algorithme de recherche qui progresse à partir d'un sommet S en s'appelant récursivement pour chaque sommet voisin de S. Il explore entièrement les chemins un par un : pour chaque sommet, il marque le sommet actuel, et il prend le premier

sommet voisin jusqu'à ce qu'un voisin n'ait plus de voisins (ou que tous ses voisins soient marqués), il revient alors au sommet père (cette description est tirée de la page Wikipédia).

Le code est disponible à l'adresse suivante ou dans le fichier *« TP2_RE51_profondeur_Vincent_MERAT.c »*. Le résultat de l'exécution de ce code est disponible à l'adresse suivante dans lequel les résultats ont été triés dans l'ordre chronologique.

B. Explications

Le code présenté reprend l'algorithme travaillé en TD. On peut remarquer d'après les résultats que le maître commence puis que les branches sont parcourues les unes après les autres.

III. Parcours en largeur

A. Présentation

L'algorithme de parcours en largeur est différent de celui en profondeur vu précédemment par le fait que, à partir d'un nœud source S, il liste d'abord les voisins de S pour ensuite les explorer un par un. Ce mode de fonctionnement utilise donc une liste dans laquelle il prend le premier sommet et place en dernier ses voisins non encore explorés.

Les nœuds déjà visités sont marqués afin d'éviter qu'un nœud ne soit exploré plusieurs fois, cependant, ce marquage n'est pas toujours nécessaire (cette description est tirée de la page Wikipédia).

Le code est disponible à l'adresse suivante ou dans le fichier *« TP2_RE51_largeur_Vincent_MERAT.c »*. Le résultat de l'exécution de ce code est disponible à l'adresse suivante dans lequel les résultats ont été triés dans l'ordre chronologique.

B. Explications

Le code présenté reprend l'algorithme travaillé en TD, cependant, celui-ci ne fonctionne pas avec une liste comme dans la présentation précédente. Cependant la topologie est tout de même correctement parcourue.

Conclusion

Nous pouvons remarquer que les 2 méthodes de parcours produisent sensiblement le même nombre de messages. En revanche, le temps de parcours est plus long pour le parcours en profondeur, ce qui confirme les conclusions émises lors du TD.

Le parcours en largeur ou en profondeur peuvent être choisis dans à peu près tous les cas. Globalement, si l'un est possible, l'autre également. Il faudra donc prendre en compte les fonctions et besoins impliquant l'implémentation d'un des 2 algorithmes. Au vu des résultats de ce TP, on pourrait cependant pencher sur le parcours en largeur, celui-ci étant globalement légèrement plus rapide.