Exercices sur le cours du 14 novembre.

Exercice 0

Avant toute chose, je vous propose de dessiner un graphe et d'effectuer un parcours en profondeur en traçant l'arborescence du parcours. Puis, vous déterminerez le statut des autres arcs (transverse, avant, arrière) Enfin, vous appliquerez l'algorithme de détection de circuit.

Exercice 1

Question 1:Programmer en C l'algorithme de détection de circuit avec un parcours en profondeur.

Question 2:Ajouter à l'algorithme le calcul d'un tableau pere qui donne, pour chaque sommet, son père dans l'arborescence du parcours.

Question 3: Comment modifier l'algorithme pour, lorsqu'on a détecté un circuit, retourner un circuit sous forme d'une liste chaînée de sommets?

Exercice 2

Question 1: programmer l'algorithme de numérotation topologique avec un parcours en profondeur.

Question 2: est-ce qu'une numérotation topologique est unique?

Question 3: Combiner les deux algorithmes pour produire une fonction C qui, lorsque le graphe possède un circuit, renvoie 1, et sinon calcule la numérotation topologique du graphe et renvoie 0.

Exercice 3

On s'intéresse dans cet exercice à une application des éléments précédents. On a un ensemble de n tâches, chaque tâche i possède une durée 1. Ces tâches sont liées par des contraintes de précédence exprimées à l'aide d'un graphe sans circuit G: (i,j) arc signifie que pour commencer j il faut que i soit terminée.Pour faire ces tâches, on dispose de 2 machines strictement identiques qui peuvent fonctionner en parallèle.On souhaite décider pour chaque tâche d'une date d'exécution t[i] et d'une machine m[i] (entre 1 et 2).

On propose l'algorithme suivant:

1) Faire une numérotation topologique des tâches.
2) Poser t=0
3) Pour i de 1 à n,

   si à la date t, la machine 1 est libre, mettre la tâcheN[i] à t sur la machine 1

   sinon, si la machine 2 est libre et que N[i] n'est pas un successeur de la tâche faite sur la machine 1 à t, mettre la tâche N[i] à t sur la machine 2

   sinon, t++ et placer N[i] à t sur la machine 1.

Question 1 : Montrer que cet algorithme fournit des dates d'exécution qui respectent les contraintes de précédence.

Question 2: A votre avis, cet algorithme permet-il d'obtenir la plus petite durée possible de l'ensemble des tâches (justifier)?

Question 3: Programmer en C l'algorithme.