**Chapitre 6 Ordonnancement graphes orienté sans circuits**

ordonnancement séquentiel : exécuter les tâches l’une après l’autre en respectant les dépendances

ordonnancement parallèle : exécuter les tâches en parallèle tout en respectant les dépendances

ordonnancement parallèle contrant : un nb borné de personnes

existance de cycles : si G a un cycle, la tâche c1 ne peut être réaliser avant c2, c2 ne peut être réaliser avant c1, donc ce projet est irréalisable

Propriétés :

1 - G sans circuit, sous graphes sans circuit

2 - G sans circuit, graphe inverse sans circuit

3 - G sans circuit, tous ses chemins sont élémentaires

(Rappel : élémentaires : passe pas deux fois par le même sommet)

Théorème : tout graphe sans circuit possède une source et un puits

Tri topologique : gruitaphe orienté G, toute numérotation des sommets respectant l’ordre des arcs

Théorème : un graphe orienté admet un tri topo ssi il est sans circuit

Algo de Tri topo : un ordonnance séquentiel

S := [ ];

T := [ ];

Pour x de 1 à n faire

Degre[x] := degré sortant(x);

si Degre[x] = 0 alors

ajouter(S,x);

{S contient alors toutes les sources de G}

tant que S <> [ ] faire

x := enleverTete(S);

Pour chaque successeur y de c dans G faire

Degre(y) := Degre(y) - 1;

si Degre(y) = 0 alors

ajouter(S,y);

Tri par niveau : G sans circuit, ssi X admet une partition en niveaux

Algo de Tri par niveau : un ordonnance en parallèle optimal

N1 := [ ];

T := [ ];

Pour x de 1 à n faire

Degre[x] := d^-(x) dans G;

Si Degre[x] = 0 alors

ajouter(N1,x);

{S contient alors toute les sources de G}

tant que N1<>[ ] faire

ajouterFin(N1,T);

N2 :=[ ];

pour tout x dans N1 faire

{Mettre à jour les degré entrants des successeurs de x}

Degre[y] := Degre[y] - 1;

si Degre[y] = 0 alors

ajouter(y,N2);

N1:=N2;