RAPPORT PROJET

**Les structures de données utilisées :**

Pour réaliser ce projet au mieux nous avons opté pour l’utilisation de listes d'adjacence pour stocker nos données. Chacun des éléments de cette liste d’adjacence est un nœud. Un nœud possède plusieurs attributs: son nom, son indice (0<i<nbr\_noeuds-1), son nombre de voisins dans le graph relationnel, ses coordonnées, son semestre au plus tôt et au plus tard, son numéro de GP et un booléen permettant de savoir si c’est un GP.

Toutes les données sont renseignées pour chaque case de la liste d’adjacence. Ensuite chaque nœud est relié à ses voisins qui eux ont seulement les attributs de nom, indice, coordonnées de correct. Pour accéder aux autres attributs il faut souvent faire appel à la liste d’adjacence sous la forme : liste\_adjacence[noeudDontOnChercherL’attribut->indice]->attributRecherché

D’autres structures de données sont utilisées comme par exemple une matrice d’adjacence dans le programme de dépendance des GP.

Pour ce qui est des structures. Il y a la structure “nodes” qui représente chaque noeuds et chaque voisin des cases de la liste d’adjacence, une structure de coordonnées “coord” permettant de stocker les deux entiers et une structure de file “File” contenant un entier et l’élément suivant, utile pour la fonction visiter. Enfin il y a une structure “bordFile” pour identifier la tête et la queue de la file.

**Les algorithmes proposés :**

LireNoeuds: Créer et retourne la liste d’adjacence avec lecture du fichier nodes.csv ligne par ligne pour remplir la liste d’adjacence.

O(V): Parcours du fichier de noeuds et ajout des noeuds

LireAretes : Lis le fichier edges.csv ligne par ligne. Pour chaque ligne, le programme vérifie l’intégrité du fichier (les deux nœuds existent, la ligne n’existe pas déjà) grâce à la liste d'adjacence. Ensuite, le programme ajoute les voisins à chaque nœud si l’arête est valide.

LireAretes calcule aussi le degré de chaque nœud et retourne le degré maximum trouvé.

O(V+E): Parcours du fichier d’arêtes puis recherche du noeud correspondant aux coordonnées

TrouverCluster : A partir de graph relationnel, trouve le nombre de nœuds isolés et de clusters puis les affiches dans mainResults.

O(V+E): Parcours de chaques voisins de chaque noeuds

InterdependanceGP et DependanceEntreGP : l’utilité de ces fonctions est dans leur nom. Pour InterdependanceGP, il a fallu rajouter des informations dans la structure nodes comme le numéro du GP correspondant au nœud, que ce soit un UP ou un GP. Ceci permet d’éviter une boucle en plus et donc de réduire la complexité temporelle.

Complexité pour les deux programmes : O(V+E²) : On parcourt les nœuds pour trouver les GP. Quand on trouve un GP on parcourt les voisins des voisins de ce GP.

CreerGraphPrecendence : On parcourt la liste d’adjacence donc les voisins de chaque nœud afin de savoir si le semestre du voisin est inférieur ou égale au semestre du nœud. Si oui, alors la liaison avec le voisin est supprimée.

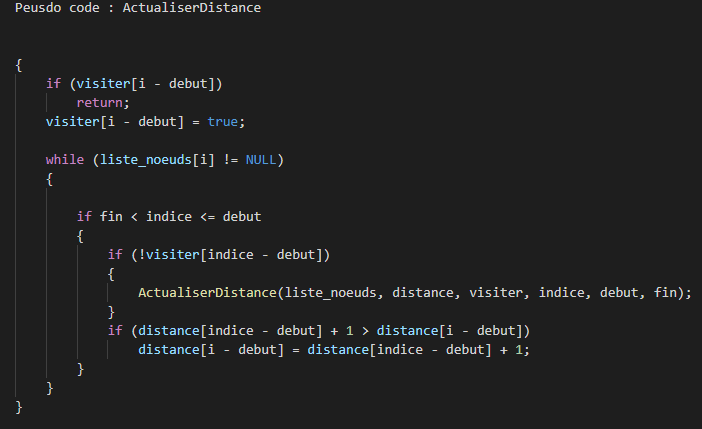
O(V+E) Parcours de chaque nœud et de chaque arête

TrouverPrerequis: Recherche des UP qui ont des prérequis éloignés mais aussi des UP et GP critiques et des chemins critiques.

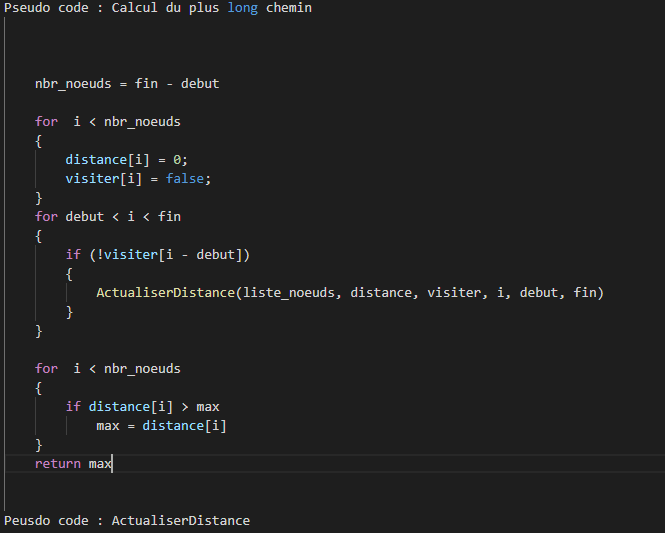
O(V+E) Parcours de chaque nœud et de chaque arête

Plus long chemin (Graph de précédence et relationnel) :

* Pseudo code de l’actualisation de la distance



* Pseudo code plus long chemin :



complexité temporel : O(V+E) car on parcourt toutes les arêtes de tous les noeuds

**Limitation du projet :**

Il y a de plus plusieurs problème avec le fichier d’input nodes.csv, on a donc décidé de le modifier :

Dans les noms d'UP / GP, les "," ont été supprimé

Par convention, le nombre de nœuds à été écrit au début de fichier.

Les "é" ont été changé en "e" sauf pour les GP comme "DECOUVERTE..."

Les "à" ont été changé en "a"

De plus, dans un chemin critique, chacun des sous-chemins est un chemin critique, il est donc affiché dans le fichier ce qui peut poser problème dans la lisibilité du fichier.  
Pour ce qui est du semestre le plus tôt. Il faut savoir que certains nœuds peuvent être validés dès le semestre 5 par exemple mais l’UP/GP correspondant n’est enseigné qu’au semestre 7 ou 8 par exemple. Cet UP/GP ne peut donc pas être validé dans la réalité au semestre 5.

Enfin, pour ce qui est de la dépendance et de l’interdépendance des GP, nous sommes partis du principe que chaque UP ne dépendait que d’un seul GP. Si ceci n’est pas vérifié alors il peut y avoir des problèmes dans la véracité des output. Au départ nous étions aussi parti du principe qu’un GP ne pouvait dépendre d’un autre GP mais il s’avère que dans notre base de donnée il y avait un cas qui correspondait à cette exception, nous avons donc modifié le programme afin de le prendre en compte.

Choix :

Le but de ce projet était de limiter au maximum la complexité spatiale et temporelle. Nous avons donc utilisé une seule liste d’adjacence tout au long du programme et l’avons modifié selon nos besoins (graph relationnel puis graph precedence). Toutes les données sont contenues dans cette liste et sont utilisables très facilement. Pour ce qui est de la complexité des programmes, elle ne dépasse rarement le O(n²) ce qui est bon signe pour la complexité temporelle. Nous avons tout mis en place afin d’éviter les boucles inutiles comme par exemple l’ajout de l’attribut numéroGP qui évite une boucle afin de vérifier si deux voisins d’un GP sont eux-même voisins dans le programme d’interdépendanceGP.

Pour limiter la complexité spatiale, le fichier edges n'est pas converti en base de données, il est lu puis les voisins des nœuds sont directement ajoutés.

Pour limiter la complexité temporelle, l'intégrité des fichiers est vérifiée à l’intérieur du programme d’ajout des voisins des nœuds.

**Analyse des résultats obtenues :**

Vitesse d’exécution :

On à une vitesse d’exécution assez rapide avec un temps d’environ 0.288s sur notre machine.

Résultats principaux obtenus :

Nombre de noeuds = 227

Nombre d'UP = 182

Nombre de GP = 45

Nombre d'arêtes = 198

Degré maximum = 12

Nombre de UP/GP isolés : 102

Nombre de clusters : 2

Le chemin le plus long pour tout le programme est : 5. Passant par :

DEGRE INDICE X Y SMSTRE plus tôt SMSTRE plus tard NOM

6 3 1 5 5 6 Electronique Analogique

11 58 2 40 6 7 Electronique Analogique

5 95 3 41 7 8 Capteurs et Actionneurs

3 99 4 2 8 9 Gestion de Projet

3 187 5 65 9 10 Energies Renouvelables

3 211 6 1 10 10 Electronics and Energy - Project

----------------TOP 5 degré (modifiable via NBR\_TOPDEGRE) ----------------

DEGRE INDICE X Y SMSTRE plus tôt SMSTRE plus tard NOM

12 78 3 21 7 8 Systemes a Microcontrôleurs

11 58 2 40 6 7 Electronique Analogique

11 87 3 32 7 9 Programmation Orientee Objet

10 28 2 4 6 7 Algorithmique et Programmation

10 85 3 29 7 9 Introduction a la Cryptographie

Analyse :

La première partie est seulement une indication sur les données importantes du graphe.

On s’aperçoit que beaucoup de nœuds sont isolés, presque la moitié. Cette isolation fait que le semestre de validation maximal de cet UP/GP n’a pas d’importance et n’influe pas sur le cursus.

Pour ce qui est des clusters, plus le clusters est gros, plus les GP seront facile à valider car une mauvaise note dans un UP peut être rattrapée dans les autres UP.

D’après le degré maximum qui est de 12 sur le graphe relationnel et de 6 sur le graphe de précédence, on se rend compte que certains UP/GP sont primordiaux puisque ce sont des prérequis pour plusieurs UP/GP suivants, il est donc important de les valider au plus vite.

Par la suite, nous avons le plus long chemin retrouvé grâce à notre algorithme vu précédemment.

La distance dans ce chemin entre les deux extremums est 5, c’est donc un chemin critique. Il est donc important de valider chacun des UP le semestre où ils apparaissent pour ne pas nuire au bon déroulement du cycle ISMIN.

On remarque grâce au top degré que plus le nœud possède de voisin, plus il a de chance d’être un nœud critique, il faut donc aussi prioriser les UP avec le plus de voisins.

Pour la partie dépendance de deux GP, cela signifie que l'échec d’une matière à l’intérieur du GP peut porter préjudice pour un GP différent. En constatant lesquels des GP sont les plus liés, on peut retrouver les nœuds qui relient deux GP différents et ainsi les validés aussi en priorité.

Enfin, à l’intérieur d’un même GP, les UP les plus importants à valider sont ceux qui sont source d’une interdépendance.

Il faut donc valider en priorité les GP qui ont un haut score d’interdépendance et de dépendance en se focalisant sur les UP qui augmentent ce score.

Enfin, les résultats des chemins les plus longs dans Others.txt peuvent aussi nous indiquer quels UP il faut valider en premier pour chaque semestre, année, programme. Sachant que sur le programme, le premier UP qui détermine le chemin le plus long est vision et stratégie de l’entreprise en S5 qui est donc à prioriser.