# Rapport projet informatique S3

Pour ce qui est de l'organisation au sein du groupe et la répartition des taches nous avons décidé de travailler ensemble à chaque fois donc dès que l'on trouvait un peu de temps libre nous nous réunissions dans une salle ou dans l'un de nos logements ou durant les cours d'informatique dédiés au projet.

Donc pour commencer nous avons lu et essayé de comprendre bien le sujet qui paraissait au début assez complexe donc nous avons essayé de bien comprendre les attendus la façon dont on devait trier le fichier et la façon dont il était organisé. Après cette phase finie nous avons décidé de commencer par la partie C plus précisément histogramme en lui donnant directement le fichier avec les données. On a observé que les données utiles à cette partie étaient seulement les lignes de la forme source vers usine ou usine elle-même donc on ne traitait que celles-ci.

Donc à chaque fois qu'on vérifiait si l'usine de la ligne existait dans l'index (AVL comporte usine et ses données) et sinon la créait, donc à chaque fois qu'on avait une ligne source vers usine on implémentait les données utiles pour histo src et real donc src eau transmise vers usine avant considération fuite et real après considération des fuites. Et le max qui lui est obtenu dans les lignes usines. Et à la fin au cas où le max serait dépassé on vérifie que src et real ne sont pas au-dessus de cette valeur et si c'est le cas on remet à max.

Puis par la suite quand on a raccordé avec un script shell dont on parlera après on a rajouté une fonction exportation qui ouvre en mode w le fichier affiche l'en-tête avec idusine;opt(max,src,real) puis ensuite comme demandé les usines sont retournées dans l'ordre lexicographique inverse donc avec un affichage infixe inverse.

Donc maintenant concernant la partie leaks nous avons eu besoin de pas mal de temps de réflexion il était indiqué qu'un index (AVL) serait très utile mais quelle autre structure utiliser nous avons recherché plein de possibilités et questionné l'IA et donc nous sommes arrivés à une structure Acteur qui contient l'id de l'usine et une structure lien acteur qui va contenir une liste chaînée de tous les fils de cet acteur avec leur id leur % de fuite et eux-mêmes leur structure acteur contenant leur liste chaînée de fils et ainsi de suite et cette structure acteur est contenue dans l'index pour l'usine correspondante.

Après de là on a traité le fichier ici dans le cas présent c'est seulement pour une usine donc quand on traite une ligne on vérifie que c'est bien l'usine voulue sinon on la passe et après du coup on fait comme dans la partie d'avant avec max et real pour l'usine puis ensuite pour toutes les différentes parties usine vers stockage, stockage vers jonction … on vérifie que l'acteur amont

(col2) existe dans l'index sinon on le crée et rajoute dans l'index de même pour l'aval (col3) et après on crée le lien entre les deux donc en gros on rajoute l'aval en début de la liste chaînée des enfants de l'amont.

Puis quand on a fini tout le traitement on calcule le plus important les fuites donc on vérifie que l'usine existe et on récupère ses données dans l'AVL et donc on lance une fonction récursive de propagation qui à chaque fois parcourt la liste chaînée de fils et divise la quantité d'eau également puis ensuite avec leurs données de % de fuite pour chaque fils on calcule sa fuite et son eau après fuite on implémente la fuite dans fuite totale et on va dans ce fils et on refait les mêmes calculs pour ses enfants vice versa. Et à la fin on exporte dans un fichier historique qui vérifie si celui-ci est vide et dans ce cas-là rajoute l'en-tête puis écrit l'id et sa fuite sur son réseau aval.

Donc en fonction du mode voulu par le script shell on lance le traitement soit en mode histo soit en mode leaks et à la fin tous les AVL et structures sont libérés.

Maintenant pour finir avec la partie script shell, d'abord pour le temps on le prend au début et à la fin de l'exécution du shell et on le retourne à la fin après soustraction début-fin ensuite par la suite on a géré la rentrée des arguments on vérifie bien qu'ils correspondent à ce qui est attendu sinon retourne des messages d'erreur et indique ce qui a mal été renseigné.

De plus en fonction de la commande rentrée on génère le nom de fichier de sortie, ensuite le shell compile avec le makefile et là on a 2 cas donc leak le plus simple qui lance avec 4 arguments leak, id usine, fichier entrée, fichier sortie. Si tout s'est bien passé le C a retourné 0 et le doc historique mis à jour et exit à la fin.

Dans l'autre cas histo il se lance avec histo, opt(max,src,real), fichier entrée et fichier sortie et là de même pour le code de retour et ensuite on doit trier le fichier obtenu (col2) par ordre croissant et on prend les 50 premières et 1 dernière qu'on met dans des fichiers temp supprimés par la suite et après à partir de ces nouveaux fichiers on crée les histogrammes en bâton en définissant certaines choses pour notre graphique échelle … et on utilise col1 pour nom de la barre et 2 pour la hauteur de la barre et quand les 2 histos sont créés les fichiers temp supprimés et exit.

Donc on a enfin obtenu nos histogramme mais en comparant avec ce envoyer avec le sujet pour les plus petites usines on a remarqué que certain identifiant était inversé 2 consécutivement et après consultation du fichier dat retourné par le c on a remarqué qu'en fait les 2 valeurs était égale donc cela ne relève pas réellement d'une erreur.

Et on a fini par un des deux bonus celui pour la partie leak cela nous est venu directement a chaque fois qu'on ajoute la propagation des fuite on avait simplement a le comparer avec un max fuite au début 0 et si supérieur on met cette valeur a jour ainsi que les id amont et aval et on rajoute de nouvelle colonne pour exportation. En revanche pour le second bonus on a esssayé mais on a eu des problemes pour les petites usine et la répartition de barre et par

manque de temps nous avons pas réussi a rendre cette partie fonctionnelle donc non présent dans notre rendu.