

# Rapport projet Meteo

## Pre-Ing 2 Groupe 5

Flavio MARTINS SOARES

Rayane Seddiki

Renaud  
VERIN



## TABLE DES MATIERES

Presentation du projet_____	3
Partie code SHell _____	4
Partie code C _____	8
Exemple d'execution_____	11
Planning et realisation du projet _____	14
Limitations fonctionelles _____	15
Bibliographie _____	16

# PRESENTATION DU PROJET

## Explication détaillée du projet Météo :

Le but de ce projet était de réaliser un programme permettant de traiter un fichier contenant des données météorologiques et d'en ressortir certaines informations choisies par l'utilisateur. Le fichier envoyé par l'utilisateur était un fichier « csv » contenant 15 colonnes d'information, comme le numéro de station, la température moyenne, la direction du vent ou encore la latitude et la longitude. Grâce à l'application, l'utilisateur a pour choix de vouloir un groupe d'information, que ce soit pour une zone exacte, comme la France et la Corse, ou pour toutes les coordonnées présentes dans le fichier. Les groupes d'informations disponibles étant la température et la pression, avec plusieurs modes, le vent, l'altitude, la température et l'humidité. En plus de cela l'application permet à l'utilisateur de choisir le mode de tri parmi 3, l'AVL, l'arbre binaire de recherche et un tri tableau. A la fin du processus de tri, l'application va créer un graphique, grâce aux données ressorties, grâce au programme GNUPLOT.

# PARTIE CODE SHELL

## Utilité et avantage du code :

La partie du programme de l'application en langage « Shell », est avantageuse afin de gérer les arguments entrant lors de l'exécution du programme. Si les arguments ne sont pas compatibles ou qu'il en manque pour une exécution minimale du programme le Shell renverra un message d'erreur, consultable dans un fichier. De plus le « Shell » permet d'appeler les programmes C et GNUPLOT, qui vont trier et créer des graphiques.

## Fonctionnement détaillé du code :

Premièrement le code « Shell », permettait de gérer l'exécution du programme en vérifiant les arguments. Si aucun argument, ou qu'il en manquait, lors de l'exécution le fichier le terminal affichait un message d'erreur. Le programme permet à l'utilisateur d'afficher une aide pour l'exécution en lançant la commande --help. De plus, chaque erreur d'option va pouvoir être lu dans le fichier erreur, chaque erreur correspond à un retour.

```
rssshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$ ./Meteo.bash
Usage : ./Meteo [parametre] [Zone] -f [fichier.csv]
--help pour voir les options
'cat erreur.txt' pour afficher l'erreur
```

(Exemple d'exécution sans argument)

Si l'exécution du programme est correcte, l'application va tout d'abord préparer l'arrivée de fichier temporaire, en vidant les fichiers de l'exécution précédente. De plus, il va compiler les fichiers lors de l'exécution. Ensuite le programme va initialiser toutes les variables.

```
initialisation des variables
sortie=out.csv
ZONE=0
TEMPERATURE=0
PRESSION=0
VENT=0
HUMIDITE=0
ALTITUDE=0
parametre=0          #le nombre de parametre
mode_tri=1
p=0
parametre1=0
temp=temp
tableau_arg=()
reverse=0
#tant que l'on a pas fait passer tous les arguments dans le case on boucle jusqu'au '--' qui est le dernier élément
```

(Liste des arguments présent dans le Shell)

Afin de gérer les arguments nous avons utilisé la fonction « Getopts », elle permet d'implémenter les arguments valide dans le code, et de leur attribuer une fonction.

```
while getopts t:hp:wmFAQGSOf:-:r name
do
    p=$((p+1))
    if [[ ${name} = "-" ]]      #on dit que - est une option qui a pour parametre avl
    then
        case "${OPTARG%%=*}" in
            avl );;
            abr ) mode_tri=2
                ;;
            tab ) mode_tri=3
                ;;
            * )
                echo "option longue n'existe pas --${OPTARG%%=*}">&2 >> erreur.txt
                usage
                ;;
        esac
    fi
done
```

(Exemple d'utilisation du Getopts)

Avec cela nous pouvons modifier chaque arguments afin de les utiliser plus tard pour lancer le programme correspondant.

```
#Choix des colonnes en fonction des arguments
while [[ $parametre -ne 0 ]];do
    if [[ ${tableau_arg[$parametre1]} == t1 ]]; then      #temper
        awk -F';' 'BEGIN {OFS=";"} { print $1,$11,$12,$13,$10 }' < $tab > temp${tableau_arg[$parametre1]}.csv
        parametre=$((($parametre-1))
        parametre1=$((1+$parametre1))                  #A trier en fonction du
    fi
done
```

(Exemple d'utilisation d'arguments)

Grâce à cela, nous pouvons exécuter un programme en C, correspondant aux attentes de l'utilisateur.

```
./Fichier_C/Meteo -f temp${tableau_arg[$parametre]}.csv -o REGION_${tableau_arg[$parametre]}.csv -l $nblignes -c $nbcolonnes -z F -k 4
```

(Exemple d'exécution par le Shell du programme C)

## GNUPLOT :

Enfin nous pouvons faire un graphique grâce au programme Gnuplot.

```
fi
if [[ ${tableau_arg[$p]} == t2 ]]
then
gnuplot << EOF
set terminal jpeg size 800,600
set output 'graphiqueGnuplot/${$zone}_${tableau_arg[$p]}_Graphique.jpeg'
set style data lines
set title "moyenne temperature par jour en année/mois/jours"
set datafile separator ";"
set xlabel 'jours'
set ylabel 'temperature'
set xrange [ * : * ]
set xtics rotate by 45 right
NO_ANIMATION = 1
plot "resultat_tri/OK_tri_${tableau_arg[$p]}.csv" u 1:2
quit
EOF
```

« Set terminal jpeg size 800, 600 » permet de définir l'extension du fichier de sortie en « jpeg » avec une taille de 800 sur 600.

« Set output » permet de définir le nom du fichier en sortie.

« Set style », permet de définir le style des lignes sur le graphique.

« Set title » permet de donner un titre au graphique.

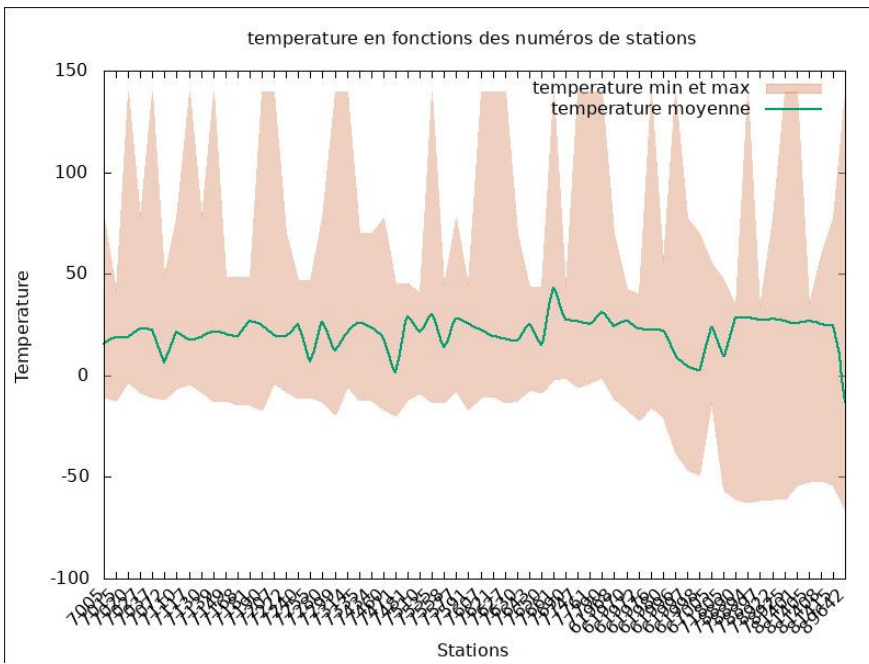
« Set datafile separator ';' » permet d'indiquer à Gnuplot le séparateur des cases du fichier, il est originellement initialisé à ' '.

« Set x/y label permet de nommer l'axe des abscisses/ordonnées.

« Set xrange » permet d'indiquer l'échelle de l'axe des abscisses.

« Set xtics rotate by 45 right » permet de tourner de 45 degrés la légende de l'axe des abscisses.

« plot » permet d'indiquer le fichier que Gnuplot doit traiter suivi de « u » pour « using » permettant d'indiquer les colonnes à utiliser.



# PARTIE CODE C

## Utilité et avantage du code :

La 2ème partie du programme de l'application en langage « C », est avantageuse car elle permet de faire des tris assez rapides. De plus, le langage C peut être appelé grâce au programme précédent en Shell. Cela permet notamment de récupérer les arguments pour les donner au programme C qui pourra effectuer les tris demandés. Le C reçoit un fichier filtré par le shell et en ressort un fichier trié et filtré avec tout ce dont à besoin le gnuplot.

## Fonctionnement détaillé du code :

Le code en langage « C » est compilé par le shell et s'exécute automatiquement, et si une erreur est rencontrée le C renvoie un entier positif qui est géré par le shell et renvoie une erreur. Le code en C est différent pour chaque mode de tri demandé, ABRet AVL, mais aussi pour chaque option choisie par l'utilisateur, comme « w » pour le vent par exemple. La liste des options disponibles sont divisées en plusieurs groupes. Les régions, France et Corse, Antilles ou encore Antarctique, le mode de tri, AVL ou ABR. Enfin les options de tris, qui sont de 9, « t1 », « t2 », « t3 », « p1 », « p2 », « p3 », « w », « h », « m », respectivement pour température 1/2/3, pression 1/2/3, vent, altitude et humidité. Ces derniers sont expliqués dans le fichier help.txt et « readme.md » .

Tout d'abord afin de travailler avec des AVL, nous devons initialiser une structure permettant de gérer les arbres.

```
typedef struct arbre{
    double col1;
    double col2;
    double col3;
    double col4;
    double col5;
    double col6;
    int equilibre;
    struct arbre* doublon;
    struct arbre* fd;
    struct arbre* fg;
}Arbre;
```



La structure prend 6 nombres en plus de l'équilibre et permet de gérer les doublons, et les fils de l'arbre.

Ensuite pour chacun des tris proposés à l'utilisateur un programme est fait. Prenons l'exemple du tri pour le mode « t1 », ce tri a pour but produire en sortie les températures minimales, maximales et moyennes par station dans l'ordre croissant du numéro de station..

```
parbre insererABrt1(parbre a,double b1,double b2,double b3,double b4,double b5,double b6,int* h)
{
    if (a == NULL)
    {
        return creationarbre(b1,b2,b3,b4,b5,b6);
        *h=1;
    }
    else if (b1 < a->col1)
    {
        a->fg = insererABrt1(a->fg,b1,b2,b3,b4,b5,b6,h);
        *h=-*h;
    }
    else if (b1 > a->col1) a->fd = insererABrt1(a->fd,b1,b2,b3,b4,b5,b6,h);
    else
    {
        if (b2 != 0) a->col2=(a->col2+b2)/2;
        if (b3 < a->col3 && b3 != 0) a->col3 = b3;
        else if (a->col3 == 0) a->col3 = b2;
        else if (b2 < a->col3 ) a->col3=b2;
        if (b4 > a->col4) a->col4 = b4;
        else if (b2 > a->col4) a->col4=b2;
        *h=0;
    }
    return a;
}
```

Ce programme servant au tri « t1 » prend en paramètre un arbre, 6 nombres et un pointeur sur un entier, servant à l'équilibre de l'arbre. Ce programme va tout d'abord vérifier si l'arbre mit en paramètre est vide, si il l'est, il retourne un arbre contenant les nombres mis en paramètres du programme. Ensuite s'il n'est pas vide on compare « b1 » et le nombre présent dans « a->col1 ». Si « b1 » est plus petit que le nombre présent dans « a->col1 », alors on relance le programme dans le fils gauche de a. Au contraire si b1 est plus grand alors on relance le programme dans le fils droit de a. Cependant si les nombres sont égaux alors nous faisons plusieurs vérifications. Premièrement, si « b2 » est différent de 0 alors on fais la moyenne entre « a->col2 » et b2. Ensuite si « b3 » est plus petit que « a->col3 » et est différent de 0 alors « a->col3 » devient « b3 ». Sinon si « a->col3 » est égal à 0 alors ce nombre devient b2. Sinon si b2 est plus petit que « a->col3 » alors « a->col3 » devient égal à b2. Enfin si b4 est plus grand que « a->col4 », alors a->col4 devient b4. Sinon si b2 est supérieur à a->col4 alors a->col4 devient b2.

On met le pointeur sur l'entier « h » à 0. Enfin on retourne l'arbre.

Grâce à cette fonction de tri, on pourra créer une fonction permettant de faire créer un arbre en entier.

```

parbre fabricationABRt1(char *ligne, double b[], parbre a, int nbligne, int nbcolonne, FILE * fp, FILE * propre, int reverse)
{
    int h=0;
    int l=1;
    fgets(ligne, 1024, fp);
    char *val = strtok(ligne, ";");
    int c = 0;                                     //variable pour compter le nombre de colonnes
    while (val != NULL)                             //pour initialiser l'arbre
    {
        b[c]= strtod(val,NULL);                     //stockez la dans la matrice en transformant la valeur en double
        val = strtok(NULL, ";");                     //Passez à la prochaine valeur
        c++;
    }
    l++;
    a = creationarbre(b[0],b[1],b[2],b[3],b[4],b[5]);
    while (fgets(ligne, 1024, fp) != NULL)
    {
        int pourcent = (l*100.0)/nbligne;
        printf("avancement : %d %%\r", pourcent);    //permet d'afficher les lignes pour voir l'avancement du traitement
        char *val = strtok(ligne, ";");             //permet de stocker ce qui il y a entre les points virgule dans un char
        int c = 0;                                   //variable pour compter le nombre de colonnes
        while (val != NULL)                           //quand il arrive en fin de ligne
        {
            b[c]= strtod(val,NULL);                   //stockez la dans un tableau en transformant la valeur en double
            val = strtok(NULL, ";");                   //Passez à la prochaine valeur
            c++;
        }
        l++;
        a = insererABRt1(a,b[0],b[1],b[2],b[3],b[4],b[5],&h);
    }
    if (reverse == 0) prefixefichier(a,propre,nbcolonne);
    else prefixefichierdecroissant(a,propre,nbcolonne);
    return a;
}

```

(Fonction permettant de créer l'arbre trié avec le mode « t1 »)

On initialise la hauteur et l'équilibre de l'arbre. Ensuite nous utilisons la fonction « fgets », qui permet de lire le contenu d'une chaîne de caractère. Nous allons ensuite compter le nombre de colonne grâce à la fonction strtok, qui s'arrêtera au premier « ; » croisé, soit le délimiteur d'un fichier « csv ». Ensuite nous stockons cette variable dans la matrice. Après cela, nous entrons dans une boucle qui a pour condition que lorsque la ligne n'est pas vide, on transforme le caractère de la matrice en un type double et nous passons à la valeur suivante. Enfin nous insérons dans l'arbre grâce à la fonction vue précédemment. Si l'utilisateur veut un tri inversé alors on lance l'écriture du fichier en inverse, sinon non. Enfin le programme renverra un fichier que le Gnuplot pourra utiliser afin d'en fournir un graphique comme montré plus haut.

# EXEMPLE D'EXECUTION

Nom	Modifié le	Type	Taille
Fichier_C	03/02/2023 01:24	Dossier de fichiers	
graphiqueGnuplot	03/02/2023 21:08	Dossier de fichiers	
resultat_tri	03/02/2023 21:08	Dossier de fichiers	
help.txt	03/02/2023 01:03	Document texte	2 Ko
Meteo.bash	03/02/2023 18:22	Fichier BASH	20 Ko
meteo_filtered.csv	31/01/2023 15:04	Fichier CSV Micros...	233 262 Ko
README.md	03/02/2023 01:03	Fichier MD	4 Ko
Makefile	03/02/2023 01:03	Fichier	1 Ko

On vérifie que tous les fichiers sont présent.

```
sshanks@LAPTOP-N82DKDH2: ~/Code/unix/ProjetMeteo4
```

```
sshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$ ./Meteo.bash -f meteo_filtered.csv -A -t1_
```

On lance la commande sur le terminal afin qu'on ait un tri « t1 » sur la zone des Antilles.

```
sshanks@LAPTOP-N82DKDH2: ~/Code/unix/ProjetMeteo4
```

```
sshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$ ./Meteo.bash -f meteo_filtered.csv -A -t1
préparation des fichiers
```

Le programme prépare les fichiers.

```
sshanks@LAPTOP-N82DKDH2: ~/Code/unix/ProjetMeteo4
```

```
sshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$ ./Meteo.bash -f meteo_filtered.csv -A -t1
préparation des fichiers
traitement de la zone ANTILLES pour t1 :
ici
entree : tempt1.csv
sortie : REGION_t1.csv
avancement : 47 %
```

Nous pouvons voir l'avancement du projet grâce aux pourcentages



rsshanks@LAPTOP-N82DKDH2: ~/Code/unix/ProjetMeteo4

```
rsshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$ ./Meteo.bash -f meteo_filtered.csv -A -t1
préparation des fichiers
traitement de la zone ANTILLES pour t1 :
ici
entree : tempt1.csv
sortie : REGION_t1.csv
avancement : 100 %

Tri t1 :
ici
entree : REGION_t1.csv
sortie : resultat_tri/OK_tri_t1.csv
avancement : 100 %

rsshanks@LAPTOP-N82DKDH2:~/Code/unix/ProjetMeteo4$
```

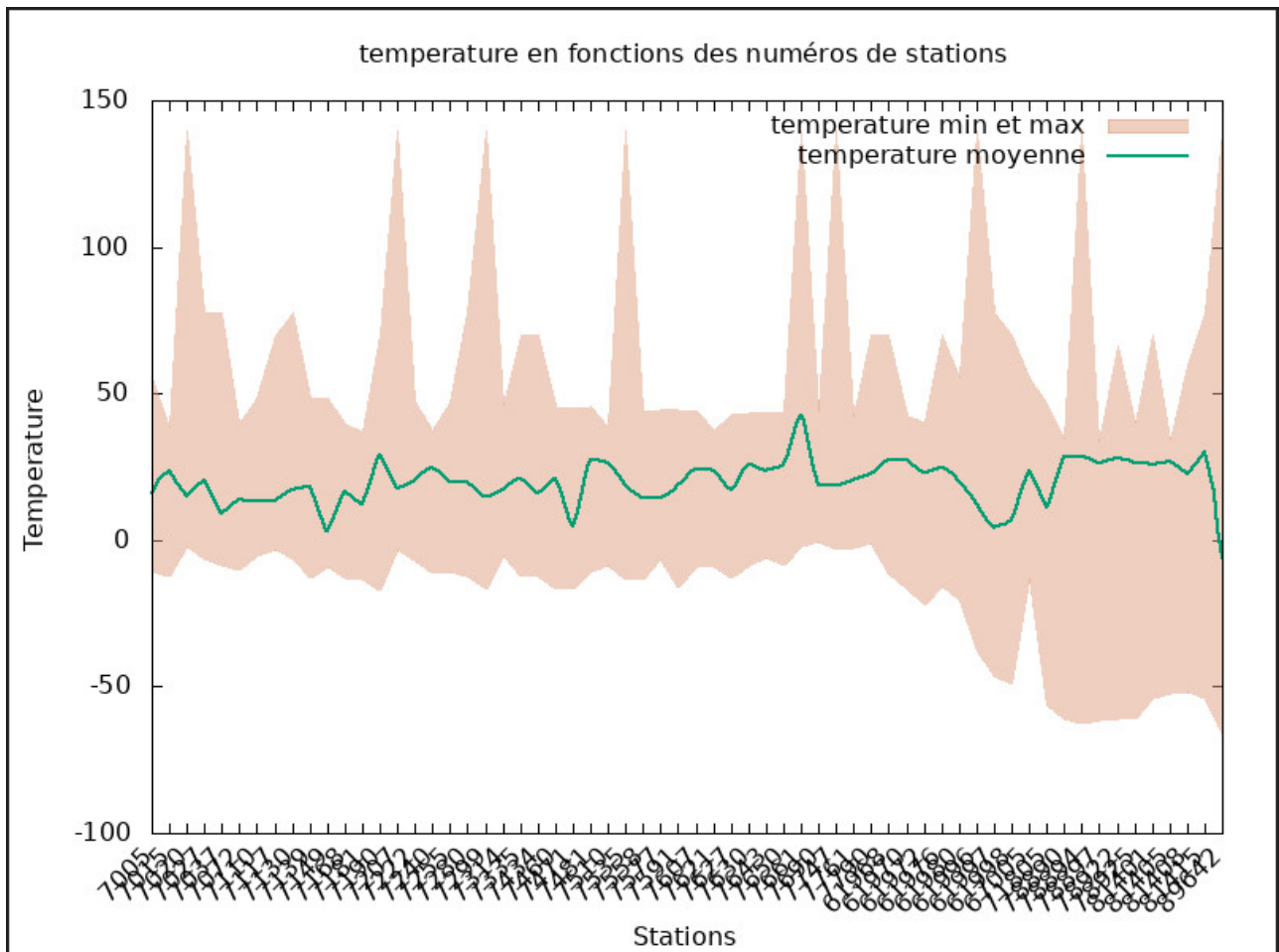
Le programme fini son processus, nous pouvons aller vérifier dans nos fichiers si les fichiers de sortie sont bien à leurs place et regarder le graphique sortie.

Nom	Modifié le	Type	Taille
 t1_Graphique.jpeg	03/02/2023 21:08	Fichier JPEG	90 Ko
 OK_tri_t1.csv	03/02/2023 21:08	Fichier CSV Micros...	4 Ko

Les fichiers sont bien à leur place.

Enregistrement automatique OK_tri_t1.csv RAVANE Seddiki																	
Fichier Accueil Insertion Mise en page Formules Données Révision Affichage Automate Aide																	
<div> <div>Calibri 11</div> <div> <div>Police</div> <div>Alignement</div> <div>Nombre</div> </div> <div> <div>Mise en forme conditionnelle</div> <div>Styles</div> <div>Styles de cellules</div> <div>Cellules</div> </div> <div> <div>Insérer</div> <div>Supprimer</div> <div>Format</div> </div> <div> <div>Σ</div> <div>Tri et Rechercher et filtrer</div> <div>Rechercher et sélectionner</div> <div>Édition</div> </div> <div> <div>Analyse de données</div> <div>Niveau de confidentialité</div> </div> </div>																	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	7005	16.024616	-11.000000	55.528667	14.595333	-60.995667											
2	7015	23.812908	-12.300000	38.100000	16.264000	-61.516333											
3	7020	14.996282	-2.500000	140.001000	16.264000	-61.516333											
4	7027	20.651741	-6.800000	77.569167	16.264000	-61.516333											
5	7037	8.809897	-8.700000	77.569167	16.264000	-61.516333											
6	7072	13.934132	-10.100000	39.800000	14.595333	-60.995667											
7	7110	13.139955	-5.300000	48.444167	16.264000	-61.516333											
8	7117	13.608042	-3.473167	70.243333	16.264000	-61.516333											
9	7130	17.354867	-6.700000	77.569167	16.264000	-61.516333											
10	7139	18.110855	-12.800000	48.445500	16.264000	-61.516333											
11	7149	3.006144	-9.400000	48.716833	16.264000	-61.516333											
12	7168	16.478683	-13.000000	40.100000	16.264000	-61.516333											
13	7181	12.128223	-13.600000	37.300000	14.595333	-60.995667											
14	7190	29.031148	-17.300000	70.243333	16.264000	-61.516333											
15	7207	17.882111	-3.218333	140.001000	14.595333	-60.995667											
16	7222	20.544666	-6.900000	47.150000	14.595333	-60.995667											
17	7240	24.798483	-11.300000	37.600000	16.264000	-61.516333											
18	7255	20.121446	-10.800000	47.059167	16.264000	-61.516333											
19	7280	19.889158	-12.400000	77.569167	16.264000	-61.516333											
20	7299	14.844857	-17.000000	140.001000	14.595333	-60.995667											
21	7314	17.172823	-5.300000	46.046833	14.595333	-60.995667											
22	7335	21.015260	-12.700000	70.243333	16.264000	-61.516333											
23	7434	16.004958	-12.500000	70.243333	16.264000	-61.516333											
24	7460	20.930727	-16.600000	45.786833	16.264000	-61.516333											
25	7471	4.938556	-17.100000	45.074500	16.264000	-61.516333											
26	7481	27.297481	-10.700000	45.726500	16.264000	-61.516333											
27	7510	26.365735	-8.700000	39.000000	14.595333	-60.995667											
28	7535	18.974945	-13.700000	140.001000	16.264000	-61.516333											
29	7558	14.424267	-13.400000	44.118500	14.595333	-60.995667											

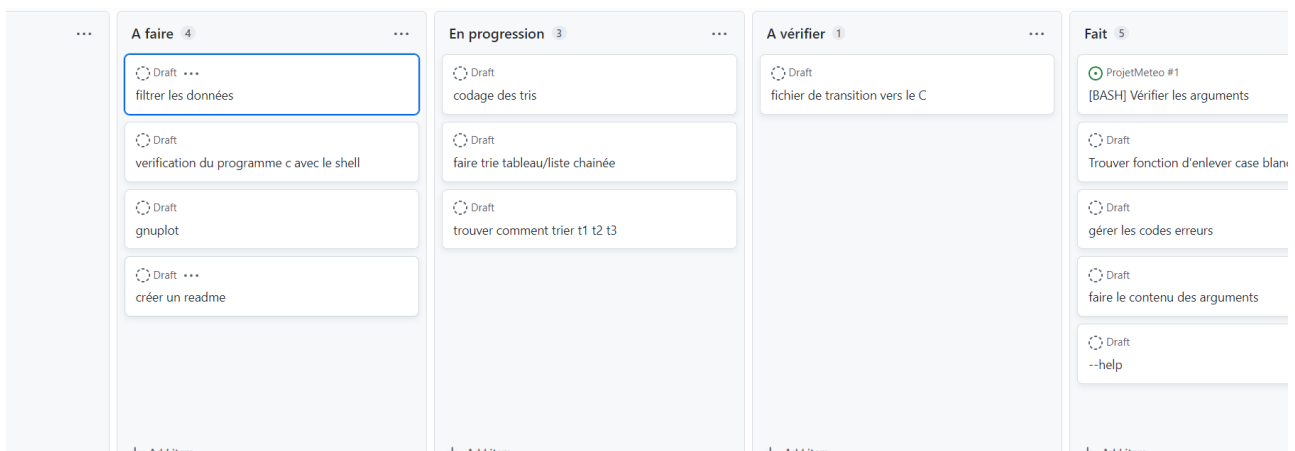
Le fichier ressorti est de la bonne forme.



De même pour le graphique Gnuplot.

# PLANNING ET REALISATION DU PROJET

Pour la réalisation de ce projet nous avons commencé le plus tôt possible. Dès l'annonce des projets nous avons commencé à l'étudier, cependant nous n'avons pas pu nous y pencher pleinement dès le début étant donné que nous avons des contrôles proches. Dès le début du projet, nous avons créé une page sur Github afin de pouvoir y déposer nos avancées. Pendant la période des vacances de Noël, nous avons commencé à faire des réunions très régulièrement. Grâce à cela nous avons pu définir un tableau de ce que nous devons faire. Nous avons utilisé une fonctionnalité de Github permettant d'afficher le travail restant, celui en cours et celui à valider.



(Exemple de la fonctionnalité Github)

.

Cela nous a permis de se coordonner et de savoir l'avancée de notre projet.

Pour la répartition des tâches, nous avons utilisé l'outil de Github, chaque personne pouvait prendre une tâche à faire et la compléter.

Flavio	Rayane	Commun
Codage Arbre	Gnuplot	Shell
Getopts	Gestion des fichiers	
Gestions des erreurs		

# LIMITATIONS FONCTIONELLES

Malheureusement, nous n'avons pas pu remplir totalement le cahier des charges. Le temps et la charges de travail additionnel ne nous a pas permit de nous concentrer autant que l'on voulait sur le projet. De ce fait notre projet est limité sur quelques points. Premièrement, nous n'avons pas pu implémenter un tri par liste chaînée, la complexité de l'algorithme était assez haute. De plus les tris par AVL n'ont pas été assez optimisé, nous aurions du tout d'abord trié les zones géographiques. Enfin nous n'avons pas pu faire le Gnuplot pour les modes de tri « t3 » et « p3 ».

# BIBLIOGRAPHIE

<https://stackoverflow.com/questions/69076568/gnuplot-plot-a-2d-heatmap-in-polar-coordinates-from-matrix-data>

<https://chat.openai.com>

[https://gnuplot.sourceforge.net/demo\\_5.4/simple.html](https://gnuplot.sourceforge.net/demo_5.4/simple.html)

[https://gnuplot.sourceforge.net/demo\\_5.4/errorbars.html](https://gnuplot.sourceforge.net/demo_5.4/errorbars.html)

[https://gnuplot.sourceforge.net/demo\\_5.4/vector.html](https://gnuplot.sourceforge.net/demo_5.4/vector.html)

[https://gnuplot.sourceforge.net/demo\\_5.4/pm3d.html](https://gnuplot.sourceforge.net/demo_5.4/pm3d.html)