Support de présentation de Ainji Joma'a :

1- Modules et architecture du programme :

En plus du programme principal, les 3 modules de notre programme sont :

a) Le module Input_Output qui globalement se charge de l'analyse de la ligne de commande.
b) Le module matrice_pleine qui effectue les calculs pour les matrices pleines.
c) Le module matrice_creuse qui effectue les calculs pour les matrices creuses.

En plus des calculs matriciels, les modules matrice_pleine et matrice_creuse contiennent aussi la lecture des fichiers en entrée et l'écriture des fichiers poids et classement (après avoir généré les vecteurs poids et classement).

Nous avons fait ce choix parce que nous avions besoin d'utiliser un paramètre (la taille, qu'on ne découvre que pendant le déroulement du programme) pour instancier le module matrice_pleine. Comme les instanciations de modules ne peuvent se faire que dans la partie déclarations du programme, ceci n'a donc pas été possible.

Architecture de programme :

Le programme principal appelle le sous-programme qui analyse la ligne de commande. Il récupère les valeurs des paramètres donnés par l'utilisateur qu'il transmet ensuite au module matrice_pleine ou matrice_creuse selon le choix. A partir de ces valeurs, les différents calculs matriciels vont être effectués afin de donner les vecteurs de poids et de classement. Ces derniers vont être ensuite écrits dans des fichiers en sorite qui sont renvoyés à l'utilisateur.

2-Démonstration de l'analyse de la ligne de commande :

L'analyse de la ligne de colonne se fait dans le module Input_Output.

Le programme parcourt les paramètres de la ligne de la ligne de commande un par un. Si le nom du fichier n'est pas renseigné le programme lève directement une exception accompagnée d'un message d'erreur.

Dans le tableau suivant on considère que le nom du fichier est renseigné.

Cas	Erreur	Ligne de commande
Aucun paramètre n'est donné.	Non	./test_cmd_line sujet.net
Certains paramètres sont donnés et d'autres non	Non	./ test_cmd_line -P -A 0.8 -R test sujet.net
Une option qui requiert une valeur est donnée sans la valeur associée.	Oui	./ test_cmd_line -E 0.8 -R test sujet.net
Une option inexistante est renseignée.	Oui	./ test_cmd_line -T sujet.net
Une valeur hors des limites acceptées est donnée.	Oui	./ test_cmd_line -A 3 sujet.net

Certains cas problématiques dans l'analyse de la ligne de commande n'ont pas été traités :

- Si l'utilisateur met une valeur après une option qui n'en requiert pas une le programme ne renvoie pas d'erreur.

  Ligne de commande testée pour illustrer : ./ test_cmd_line -P 0.8 sujet.net

- Si l'utilisateur met une option inexistante qui est composée de plus que deux caractères le programme ne renvoie pas d'erreur.
  Ligne de commande testée pour illustrer : ./ test_cmd_line -0.83 sujet.net

Dans les deux cas ci-dessus le programme ne renvoie par d'erreur alorsqu'il est pourtant sensé le faire.

Support de présentation de Arthur Bongiovanni :

1- Explication des structures des données choisies et des parcours :

La structure de donnée Matrice creuse en ada:

```
Type T_Cell_mat;
Type ptr_Matrice is access T_Cell_mat;
Type T_Cell_mat is record
   valeur      : T_reel;
   ind_L       : Integer;
   ind_C       : Integer;
   next_below : ptr_Matrice;
   next_side   : ptr_Matrice;
   -- Invariant :
   --    TAILLE >= ind_L >= 0;
   --    TAILLE >= ind_C >= 0;
   --    next_below = Null or else next_below.all.ind_L > ind_L and next_below.all.ind_C =
ind_C ;
   --    next_side = NuLL or else next_side.all.ind_C > ind_C
   --                  and for i in 0..next_side.all.ind_L -1 :
   --                    get_value(Matrice,i,next_side.all.ind_C) = 0
   -- cellules sont stockées dans l'ordre croissant des indices en ligne et en colonnes
end record;
```

On représente donc notre matrice creuse comme un seul vecteur creux mais avec différents ponts entre les valeurs (un peu comme une 'skip_liste')
Chaque élément "pointe" sur le prochain élément non nul de sa colonne (ou null si il n'y a plus d'élément non null après) et sur le premier élément non null de la prochaine colonne contenant un élément (ou null si il n'y a plus de colonne non vide après)

Pour modifier un élément de la matrice :

- On regarde d'abords si la matrice est vide (i.e. si elle ne contient que des zéros).
  Dans ce cas on créer une nouvelle cellule dont l'adresse est donnée à notre matrice.
- Sinon, on regarde si l'élément à modifier est sur une colonne précèdent la 'première valeur' de la matrice.

Dans ce cas, on ajoute juste la nouvelle cellule avant cette 'première valeur'.

- Sinon, on regarde si l'élément à modifier est sur la première colonne de la matrice.
  Dans ce cas, on regarde la ligne à modifier.
- Sinon, on recherche la colonne à modifier.

Lors de la recherche de la colonne :

- On parcourt de colonne en colonne jusqu'à ce que l'élément suivant soit null ou sur une colonne d'indice supérieure à celle qu'on veut modifier.
- Ensuite on regarde si l'élément courant est sur la bonne colonne.
  Dans ce cas, on cherche la ligne où modifier la matrice.
- Sinon, on ajoute la nouvelle cellule entre la colonne courante et la suivante.

Lors de la recherche de la ligne :

- On regarde si la ligne à modifier est 'avant' la première valeur de la colonne courante.
  Dans ce cas, on ajoute juste la cellule en haut de la colonne et on la relie avec la colonne précédente (dans le cas où on n'est pas sur la première colonne), la suivante et l'ancien 'premier' élément de la colonne courante.
- Sinon on parcourt le long de la colonne jusqu'à ce que l'élément suivant soit null ou sur une ligne d'indice supérieur à celle qu'on veut modifier.
- Ensuite, si la ligne courante correspond à celle à modifier, on remplace la valeur.
- Sinon, on ajoute la nouvelle cellule entre l'élément courant et l'élément suivant dans la colonne. On le lie aussi à la colonne suivante.

2- Démonstration du Test de la fonction modifier:

```
$./test_modify_mat_creuse

Initialisation d'une Matrice creuse:
|--E]


[ C :: 1| L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 :  6.0000-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> |--E]


Cas 'fonctionnel' :
modifier une valeur dÃ©ja prÃ©sente
[ C :: 1| L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> |--E]
```

```
ajouter dans une colonne vide
[ C :: 1| L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> |--E]


ajouter au dÃ©but d'une colonne
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> |--E]


ajouter au milieu d'une colone
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> |--E]


ajouter Ã  la fin d'une colonne
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]


Cas 'd'erreur' :
ajouter en dehors des limite thÃ©orique (colone sup)
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]
[ C :: 15| L 9 : 135.0000-> |--E]


ajouter en dehors des limite thÃ©orique (colone inf)
[ C :: -3| L 9 : -27.0000-> |--E]
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]
[ C :: 15| L 9 : 135.0000-> |--E]


ajouter en dehors des limite thÃ©orique (ligne inf)
[ C :: -3| L 9 : -27.0000-> |--E]
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 4| L -3 : -12.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]
[ C :: 15| L 9 : 135.0000-> |--E]


ajouter en dehors des limite thÃ©orique (ligne sup)
[ C :: -3| L 9 : -27.0000-> |--E]
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 4| L -3 : -12.0000-> | L 23 : 92.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  9.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]
[ C :: 15| L 9 : 135.0000-> |--E]


'Supprimer une valeur
[ C :: -3| L 9 : -27.0000-> |--E]
[ C :: 1| L 0 :  1.2340-> | L 1 :  1.0000-> | L 3 :  2.0000-> | L 5 :  3.0000-> | L 7 :  4.0000-> | L 9 :  5.0000-> |--E]
[ C :: 3| L 1 :  2.0000-> | L 3 :  4.0000-> | L 5 :  6.0000-> | L 7 :  8.0000-> | L 8 : 98.7654-> | L 9 : 10.0000-> |--E]
[ C :: 4| L -3 : -12.0000-> | L 23 : 92.0000-> |--E]
[ C :: 5| L 1 :  3.0000-> | L 3 : 123.4560-> | L 5 :  0.0000-> | L 7 : 12.0000-> | L 9 : 15.0000-> |--E]
[ C :: 6| L 4 : 17.5630-> |--E]
[ C :: 7| L 1 :  4.0000-> | L 3 :  8.0000-> | L 5 : 12.0000-> | L 7 : 16.0000-> | L 9 : 20.0000-> |--E]
[ C :: 9| L 1 :  5.0000-> | L 3 : 10.0000-> | L 5 : 15.0000-> | L 7 : 20.0000-> | L 9 : 25.0000-> | L 11 :  0.5000-> |--E]
[ C :: 15| L 9 : 135.0000-> |--E]


vidÃ© la matrice
free(): double free detected in tcache 2

raised PROGRAM_ERROR : unhandled signal
```

Conclusion :

Tous les modules implémentés sont fonctionnels et donnent les bons classements. A noter que matrice_creuse donne le bon classement avec une différence de précision dans la valeur de poids qui pourrait être due à une différence d'itération ou de précision dans les valeurs manipulées.