Documentation Projet Avancé

Projet réalisé par l'Equipe 17 :
Wilkens Marc Johnley JOSEPH
Adrien MORELLATO
Assane NDOYE
Yoan SMETS

19 Décembre 2023

UE Projet Avancé

Résumé

Ce projet informatique est une application de vote qui prend en entrée un fichier CSV contenant des données de vote et effectue le calcul pour élire un candidat selon la méthode de vote de votre choix parmi celles disponibles.

Table des matières

I – Fonctionnalités Principales

I.1 Entrée des Données

Le programme accepte deux types de fichiers CSV en tant qu'entrée. Le premier type contient des bulletins de vote, où chaque ligne représente un électeur et chaque colonne représente le choix de l'électeur pour un candidat. Le second type contient une matrice de duels, où chaque cellule indique le nombre de fois que le candidat de la ligne a été préféré au candidat de la colonne. L'utilisation du second type de fichier désactive l'option uni1, uni2, et jm.

I.2 Méthodes de Vote

I.2.1 Uninominal à un tour (uni1)

Chaque électeur vote pour un seul candidat, et le candidat avec le plus grand nombre de votes remporte l'élection.

I.2.2 Uninominal à deux tours (uni2)

Les deux candidats avec le plus grand nombre de votes au premier tour passent au deuxième tour, où le candidat avec le plus grand nombre de votes est élu.

I.2.3 Condorcet minimax (cm)

En quête d'un vainqueur selon la méthode de Condorcet, cette méthode, en l'absence d'un vainqueur évident, sélectionne le candidat gagnant en identifiant la paire de candidats présentant la marge de défaite potentielle la plus réduite. L'objectif est de minimiser le risque de subir la pire défaite possible.

I.2.4 Condorcet Schulze (cs)

À la recherche d'un vainqueur selon la méthode de Condorcet, cette méthode, en cas d'absence de vainqueur, sélectionne le gagnant en évaluant les marges de victoire dans les duels un contre un entre tous les candidats. Le choix se porte sur le candidat ayant les marges de victoire les plus fortes dans l'ensemble des comparaisons. Cette approche prend en considération la puissance relative des victoires individuelles pour déterminer le vainqueur.

I.2.5 Jugement majoritaire (jm)

Le gagnant est déterminé en attribuant des jugements qualitatifs, puis en choisissant le candidat ayant la médiane la plus élevée . En cas d'égalité , on détermine la nouvelle mention majoritaire de ces candidats en enlevant le vote de l'électeur médian.

I.2.6 Options d'Affichage

L'application propose une option pour afficher les calculs intermédiaires dans un fichier log. Si l'option -o de la ligne de commande n'est pas mentionné , l'affichage se fera sur la sortir standard.

I.2.7 Programme indépendant verify_my_vote

Cherche dans le fichier de votes, le vote correspondant au hash d'un votant , crypté avec son nom,prénom,clef secrète.

II – Utilisation - Exemple d'Utilisation - Compilation - Documentation Doxygen - Test

II.1 Utilisation

II.1.1 Programme principal Scrutin

Pour exécuter le programme scrutin, utilisez la commande suivante :
bin/scrutin -i <filename.csv> | -d <filename.csv> -m uni1, uni2, cm, cp, cs, jm, all [-o <log_file>]

II.1.2 Programme indépendant verify_my_vote

Pour exécuter le programme indepandant verify_my_vote, utilisez la commande suivante :
bin/verify_my_vote <filename.csv>

Paramètres :
-i : Spécifie le nom du fichier csv où se trouvent les ballots de vote.
-d : Spécifie le nom du fichier csv où se trouvent la matrice de duel.
-m : Sélectionne la méthode de vote (uni1, uni2, cm, cp, cs, jm, all).
-o : Optionnel. Spécifie le fichier log pour afficher les calculs intermédiaires.
Note L'option -i et -d ne peuvent pas être présente en même temps.

II.2 Exemple d'Utilisation

II.2.1 Programme principal Scrutin

bin/uninominales -i donnees_votes.csv -m uni2 -o log.txt
Cette commande lance le programme avec le fichier CSV "donnees_votes.csv", utilise la méthode de vote uninominal à deux tours, et écrit les calculs dans le fichier log "log.txt".

II.2.2 Programme indépendant verify_my_vote

bin/verify_my_vote donnees_votes.csv

II.3 Compilation

II.3.1 Programme principal Scrutin

Pour compiler le programme scrutin, exécutez la commande suivante :
make scrutin

II.3.2 Programme indépendant verify_my_vote

Pour compiler le programme indépendant verify_my_vote, exécutez la commande suivante :
make verify_my_vote
Pour compiler les 2 programme **scrutin** et **verify_my_vote**, exécutez la commande suivante :
make all

II.3.3 Documentation Doxygen

La documentation doxygen peut être générée en utilisant la commande suivante :
make doc

Vous pouvez consulter la documentation générée en ouvrant le fichier documentation/html/index.html dans votre navigateur web.

II.3.4 Test

Les Test,teste toutes les méthodes de scrutin et peut être générée en utilisant la commande suivante :
Test/test_script.sh

III – Interfaces du projet

III.1 scrutin_methods

III.1.1 Condorcet.h

III.1.1.1 methode_Minimax()

int methode_Minimax (Matrice matriceDuel, FILE * logfile)
Description : Applique la méthode Minimax pour résoudre une élection de Condorcet. Cette méthode vise à trouver un vainqueur de Condorcet. Si aucun vainqueur n'est trouvé, elle sélectionne le candidat avec la plus petite marge de défaite potentielle.

Paramètres :
matriceDuel La matrice des duels.
logfile Pointeur vers le fichier journal où les résultats sont affichés.

Renvoie : Le vainqueur de Condorcet .

III.1.1.2 methode_Rangement_Des_Paires()

int methode_Rangement_Des_Paires (Matrice matriceDuel, FILE * logfile)
Description : Applique la méthode de tri des paires pour résoudre une élection de Condorcet. Cette méthode vise à trouver un vainqueur de Condorcet. Si aucun vainqueur n'est trouvé, elle sélectionne le candidat en comparant toutes les paires possibles de candidats et en choisissant celui qui remporte le plus grand nombre de victoires par paires.

Paramètres :
matriceDuel La matrice des duels.
logfile Pointeur vers le fichier journal où les résultats sont affichés.

Renvoie : Le vainqueur de Condorcet.

III.1.1.3 methode_Schulze()

int methode_Schulze (Matrice matriceDuel, FILE * logfile)
Applique la méthode Schulze pour résoudre une élection de Condorcet. Cette méthode vise à trouver un vainqueur de Condorcet. Si aucun vainqueur n'est trouvé, elle sélectionne le candidat en évaluant les marges de victoire dans les duels un contre un entre tous les candidats et en choisissant celui qui a les marges les plus fortes dans l'ensemble des comparaisons.

Paramètres :
matriceDuel La matrice des duels.
logfile Pointeur vers le fichier journal où les résultats sont affichés.

Renvoie : Le vainqueur de Condorcet.

III.1.2 jugement.h

III.1.2.1 trierVotes()

void trierVotes (ListCand * lstCand)
Description : Trie tous les tableaux de votes d'une liste de candidats.

Paramètres :
lstCand : Liste de candidats.

III.1.2.2 mediane()

double mediane (Tableau tab)
Description : Calcule la médiane d'un tableau d'entiers.

Paramètres :
tab : Tableau d'entiers.

Renvoie : La médiane des entiers.

III.1.2.3 afficherVote()

void afficherVote (Tableau tab, FILE * logfile)
Description : Affiche le tableau des votes dans un fichier log.

Paramètres :
tab : Tableau d'entiers représentant les votes.
logfile : Pointeur vers le fichier journal où les logs sont affichés.

III.1.2.4 jugementMajoritaire()

Candidat* jugementMajoritaire (ListCand * lstCand, ListElect * lstElect, char * fichier, FILE * logfile)
Description : Applique le jugement majoritaire pour déterminer le candidat gagnant.

Paramètres :
lstCand : Liste de candidats.
lstElect : Liste d'électeurs.
fichier : Nom du fichier d'entrée.
logfile : Pointeur vers le fichier journal où les logs sont affichés.

Renvoie : Le candidat gagnant.

III.1.2.5 voteJugementMajoritaireBallot()

void voteJugementMajoritaireBallot (char * fichier, FILE * logfile)
Description : Effectue un vote majoritaire avec tous les paramètres nécessaires pour simplifier un appel à la fonction.

Paramètres :
fichier : Nom du fichier d'entrée contenant les informations.
logfile : Pointeur vers le fichier journal où les logs sont affichés.

III.1.3 uninominales.h

III.1.3.1 Resultat

typedef struct s_Resultat Resultat
Définition opaque du type de données abstrait Resultat.

III.1.3.2 uninominal_un_tour()

Resultat uninominal_un_tour (ListElect *lstElect, ListCand *lstCand, FILE * fichier)
Description : Fonction de vote uninominal à un tour.

Paramètres :
lstElect Une liste d'électeurs.
lstCand Une liste de candidats.
fichier Un fichier journal permettant de stocker les calculs de l'algorithme.

Renvoie : Une structure Resultat contenant le candidat gagnant et son nombre de votes.

III.1.3.3 finalistes_uninominal_deux_tours()

Candidat** finalistes_uninominal_deux_tours (ListElect *lstElect,ListCand *lstCand,FILE *fichier)
Description : Fonction permettant de récupérer à partir d'un tableau les deux candidats ayant le plus de votes.

Paramètres :
lstElect Une liste d'électeurs.
lstCand Une liste de candidats.
fichier Un fichier journal permettant de stocker les calculs de l'algorithme.

Renvoie : Un tableau contenant les deux candidats avec le plus de votes ou un tableau contenant le candidat ayant eu la majorité absolue de votes.

III.1.3.4 uninominal_2nd_Tours()

Resultat uninominal_2nd_Tours (ListElect* lstElecteurs, Candidat** finalistes, FILE* logfile)
Description : Fonction permettant de récupérer le candidat avec le plus de votes entre les deux candidats finalistes.

Paramètres :
lstElecteurs Une liste d'électeurs.
finalistes Un tableau de finalistes.
fichier Un fichier journal permettant de stocker les calculs de l'algorithme.

renvoie : Une structure Resultat contenant le candidat gagnant et ses votes.

III.1.3.5 createTest()

ListElect* createTest (ListElect* lstElect, ListCand* lstCand, int nbTest)
Description : Fonction de test pour créer des électeurs votants pour des candidats connus de manière aléatoire.

Paramètres :
lstElect Une liste vide d'électeurs.
lstCand Une liste de candidats.
nbTest Un nombre défini de tests.

Renvoie : Une liste contenant nbTest électeurs.

III.1.3.6 unTour()

void unTour (char* fichier, FILE* logfile)
Description : Fonction de mise en forme du vote uninominal à un tour.

Paramètres :
fichier Fichier contenant les données.
fichier Un fichier journal permettant de stocker les calculs de l'algorithme.

III.1.3.7 deuxTours()

void deuxTours (FILE* logfile, char* fichier)
Description : Fonction de mise en forme du vote uninominal à deux tours.

Paramètres :
logfile Un fichier journal permettant de stocker les calculs de l'algorithme.
fichier Fichier contenant les données.

III.2 utility_module

III.2.1 lecture_csv.h

III.2.1.1 removeNewline()

void removeNewline (char* str)
Description : Enlève les sauts de ligne d'une chaine de caractère.

Paramètres :
str la chaine à modifier

Renvoie : la chaine modiée.

III.2.1.2 obtenir_nom_Candidat()

char* obtenir_nom_Candidat (const char* filename, int numColonne, bool isBallot)
Description : Obtient le nom du candidat à partir du fichier spécifié.

Paramètres :
filename Le nom du fichier à lire.
numColonne Le numéro de colonne à extraire.
isBallot Un booléen indiquant s'il s'agit une fichier csv de type balot ou dans cas contraire une matrice de duel

Renvoie : Un pointeur vers la chaîne de caractères représentant le nom du candidat.

III.2.1.3 afficher_vote()

void afficher_vote (const char* filename, char* hash)
Description : Affiche les votes correspondant à un hachage spécifié dans le fichier spécifié.

Paramètres :
filename Le nom du fichier à lire.
hash Le hachage à rechercher et afficher.

III.2.1.4 lireBallot()

int lireBallot (char* filename, Matrice* Matrice)
Description : Lit un fichier CSV contenant les votes sous la forme de bulletins et les transforme en matrice de duels. Cette fonction prend en entrée le nom d'un fichier CSV contenant les bulletins de vote, puis elle crée une matrice de duels représentant les comparaisons deux à deux des candidats. La matrice résultante est stockée dans une structure Matrice.

Paramètres :
filename Le nom du fichier CSV contenant les bulletins de vote.
Matrice Un pointeur vers la structure Matrice pour stocker les données lues.

Renvoie : Le nombre d'électeurs dans le fichier.

III.2.1.5 lireMatriceDuel()

int lireMatriceDuel (char* filename, Matrice* Matrice)
Description : Lit la matrice de duels à partir d'un fichier CSV et remplit la matrice de duels correspondante. Cette fonction prend en entrée le nom d'un fichier CSV contenant une matrice de duels, puis elle remplit la matrice de duels représentant les comparaisons deux à deux des candidats. La matrice résultante est stockée dans une structure Matrice.

Paramètres :
filename Le nom du fichier CSV contenant la matrice de duels.
Matrice Un pointeur vers la structure Matrice pour stocker les données lues.

Renvoie : Le nombre d'électeurs dans le fichier.

III.2.1.6 fichierValide()

int fichierValide (const char* filename)
Description : Vérifie la validité des données présentes dans le fichier spécifié.

Paramètres :
filename Le nom du fichier à vérifier.

Renvoie : 1 si le fichier est valide, 0 sinon.

III.2.1.7 verifFichier()

void verifFichier (const char* filename)
Description : Fonction permettant de factoriser la fonction fichierValide pour faciliter son appel.

Paramètres :
filename Le nom du fichier à vérifier.

III.2.1.8 getElecteur()

ListElect* getElecteur (ListElect * lstElect, ListCand * lstCand, char * fichier, Tableau tab)
Description : Fonction créant une liste d'électeurs à partir d'un fichier et du candidat pour lequel il vote.

Paramètres
lstElect Liste vide d'électeurs
lstCand Liste de candidats possibles
fichier Fichier contenant les données des électeurs et de leurs votes

tab Tableau qui va contenir les différents votes de l'électeur pour chaque candidat

Renvoie : Une liste d'électeurs initialisée.

III.2.1.9 getCandidat()

ListCand* getCandidat (ListCand* lstCand, char* fichier)
Description : Fonction créant une liste de candidats à partir d'un fichier.csv

Paramètres :
lstCand Liste vide de candidats.
fichier Le fichier contenant les données.

Renvoie : Une liste de candidats initialisée.

III.2.1.10 getIndice()

int getIndice (Candidat* candidats, char* fichier)
Description : Fonction renvoyant l'indice colonne d'un fichier.csv de chaque candidat présent dans la liste.

Paramètres
candidats Les candidats présent.
fichier Le fichier.csv à analyser.
tab le tableau vide qui contiendra les indices des candidats.

Renvoie : le tableau contenant les indices des candidats.

III.2.1.11 electeurs2ndTours2Candidats()

ListElect* electeurs2ndTours2Candidats (ListElect* lstElect, Candidat* c1, Candidat* c2, char* fichier, FILE* logfile)
Description : Fonction permettant de créer une liste d'electeurs ayant voté pour 2 candidats différents en choisissant celui qu'ils ont préféré.

Paramètres
lstElect liste vide
c1 Candidat 1
c2 Candidat 2
fichier le fichier contenant les votes et les informations des electeurs.

Renvoie : Une liste d'électeurs.

III.2.1.12 initJugement()

void initJugement (ListCand* lstCand, ListElect* lstElect, char* fichier)
Description : Fonction permettant d'initialiser les votes d'une liste de candidat suivant les votes d'une liste d'électeurs.

Paramètres
lstCand Liste de Candidat
lstElect Liste d'électeurs
fichier Fichier contenant les votes des électeurs

III.2.2 set.h

III.2.2.1 DisjointSet

typedef struct s_disjointSet DisjointSet
Description : Définition opaque du type DisjointSet (Ensemble Disjoint).

III.2.2.2 createDisjointSet()

DisjointSet* createDisjointSet (int n)
Description : Constructeur pour créer un ensemble disjoint.

Paramètres :
n La taille de l'ensemble.

Renvoie : Un pointeur vers l'ensemble disjoint nouvellement créé.

III.2.2.3 freeDisjointSet()

void freeDisjointSet (DisjointSet* set)
Description : Destructeur pour libérer la mémoire allouée à un ensemble disjoint.

Paramètres :
set L'ensemble disjoint à libérer.

III.2.2.4 addEdge()

void addEdge (DisjointSet* set, int x, int y)
Description : Ajoute une arête entre les ensembles qui contiennent les éléments x et y.

Paramètres
set L'ensemble disjoint.
x Élément x.
y Élément y.

III.2.2.5 doesCreateCycle()

int doesCreateCycle (DisjointSet* set, int u, int v)
Description : Vérifie si l'ajout d'une arête entre les ensembles contenant u et v crée un cycle.

Paramètres :
set L'ensemble disjoint.
u Élément u.
v Élément v.

Renvoie : true si l'ajout de l'arête crée un cycle, sinon false.

III.2.2.6 findRoot()

int findRoot (DisjointSet* set)
Description : Trouve la racine (ensemble parent) d'un élément dans l'ensemble disjoint.

Paramètres :
set L'ensemble disjoint.

Renvoie : La racine de l'élément.

III.2.2.7 printSet()

void printSet (DisjointSet* set, FILE* logfile)
Description : Affiche l'ensemble disjoint dans un fichier journal.

Paramètres :
set L'ensemble disjoint.
logfile Pointeur vers le fichier journal où afficher l'ensemble disjoint.

III.2.3 utiles.h

III.2.3.1 ListCand

typedef struct s_List_Cand ListCand
Description : Définition opaque du type abstrait de données ListCand.

III.2.3.2 ListElect

typedef struct s_List_Electeur ListElect
Description : Définition opaque du type abstrait de données ListElect.

III.2.3.3 Candidat

typedef struct s_Candidat Candidat
Description : Définition opaque du type abstrait de données Candidat.

III.2.3.4 Electeur

typedef struct s_Electeur Electeur
Description : Définition opaque du type abstrait de données Electeur.

III.2.3.5 create_electeur()

Electeur* create_electeur (const char* nom, const char* prenom, Candidat* candidat)
Description : Crée un nouvel électeur avec les informations spécifiées.

Paramètres
nom Nom de famille de l'électeur.
prenom Prénom de l'électeur.
candidat Pointeur sur le candidat choisi par l'électeur.

Renvoie : Un pointeur sur le nouvel électeur correctement initialisé.

III.2.3.6 listElect_create()

ListElect* listElect_create (void)
Description : Crée et initialise une liste vide d'électeur.

Renvoie : Un pointeur sur la nouvelle liste correctement initialisé.

III.2.3.7 list_add_Electeur()

ListElect* list_add_Electeur (ListElect* l, Electeur* electeur)
Descriprion : Ajoute un électeur à la liste d'électeurs.

Paramètres
l La liste d'électeurs à modifer.

electeur Pointeur sur l'électeur à ajouter.

Renvoie : La liste d'électeurs correctement modifiée.

III.2.3.8 listElect_delete()

void listElect_delete (ListElect* l)
Description : Supprime entièrement la liste d'électeurs.

Paramètres
l La liste d'électeurs à supprimer.

III.2.3.9 create_candidat()

Candidat* create_candidat (const char* nom, const char* prenom, int age)
Description : Crée un nouveau candidat avec les informations spécifiées.

Paramètres
nom Nom de famille du candidat.
prenom Prénom du candidat.
age Age du candidat.

Renvoie : Un pointeur sur le nouveau candidat correctement initialisé.

III.2.3.10 listCand_create()

ListCand* listCand_create (void)
Description : Crée et initialise une liste vide de candidat.

Renvoie : Un pointeur sur la nouvelle liste correctement initialisé.

III.2.3.11 listCand_add()

ListCand* listCand_add (ListCand* l, Candidat* candidat)
Description : Ajoute un candidat à la liste de candidats.

Paramètres
l La liste de candidats à modifer.
candidat Pointeur sur le candidat à ajouter.

Renvoie : La liste de candidats correctement modifiée.

III.2.3.12 listCand_delete()

void listCand_delete (ListCand* l)
Description : Supprime entièrement la liste de candidats.

Paramètres :
l La liste de candidats à supprimer.

III.2.3.13 tableauDelete()

void tableauDelete (ListCand* l)
Description : Supprime le tableau de vote de chaque candidat.

Paramètres
l La liste à supprimer.

III.2.3.14 printList()

void printList (ListCand* l)
Description : Affiche les informations de chaque candidat de la liste.

Paramètres :
l La liste de candidat à afficher.

III.2.3.15 getIndiceVote()

int getIndiceVote (Tableau tab)
Description : Obtenir l'indice de la valeur minimale du tableau.

Paramètres :
tab Le tableau de vote.

Renvoie : L'indice de la valeur minimale du tableau.

III.2.3.16 candidatAssocie()

Candidat* candidatAssocie (int nb, ListCand* lstCand)
Description : Renvoie le candidat à l'indice donné.

Paramètres
nb L'indice du candidat.
lstCand La liste de candidats.

Renvoie : Le candidat à l'indice passé en paramètre.

III.2.3.17 createTableau()

void createTableau (int taille, ListCand* lstCand)
Description : Crée et alloue de la mémoire pour chaque tableau de vote de chaque candidat.

Paramètres
taille La taille du tableau.
lstCand La liste de candidats :

III.2.3.18 isInteger()

int isInteger (const char* str)
Description : Vérifie si la chaine de caractère contient un entier.

Paramètres
str La chaine de caractère.

Renvoie : 1 si elle contient un entier, 0 autrement.

III.2.4 utils_tab.h

III.2.4.1 Matrice

typedef struct t_Matrice* Matrice
Description : Structure représentant une matrice d'entiers.

III.2.4.2 Tableau

typedef struct t_Tableau * Tableau
Description : Structure représentant un tableau d'entiers.

III.2.4.3 create_Matrice()

Matrice create_Matrice (int nb_ligne, int nb_colonne)
Description : Crée une nouvelle matrice d'entiers.

Paramètres :
nb_ligne Nombre de lignes de la matrice.
nb_colonne Nombre de colonnes de la matrice.

Renvoie : Pointeur vers la nouvelle matrice.

III.2.4.4 delete_Matrice()

void delete_Matrice (Matrice matrice)
Description : Libère la mémoire allouée pour une matrice.

Paramètres :
matrice La matrice à libérer.

III.2.4.5 init_Matrice()

void init_Matrice (Matrice matrice, int valeur)
Description : Initialise les valeurs d'une matrice avec une valeur donnée.

Paramètres
matrice La matrice à initialiser.
valeur La valeur avec laquelle initialiser la matrice.

III.2.4.6 create_Tableau()

Tableau create_Tableau (int dim)
Description : Crée un nouveau tableau d'entiers.

Paramètres :
dim Dimension du tableau.

Renvoie : Pointeur vers le nouveau tableau.

III.2.4.7 delete_Tableau()

void delete_Tableau (Tableau Tableau)
Description : Libère la mémoire allouée pour un tableau.

Paramètres :
Tableau Le tableau à libérer.

III.2.4.8 init_Tableau()

void init_Tableau (Tableau tab, int valeur)
Description : Initialise les valeurs d'un tableau avec une valeur donnée.

Paramètres :
tab Le tableau à initialiser.
valeur La valeur avec laquelle initialiser le tableau.

III.2.4.9 min_Tableau()

int min_Tableau (Tableau tab, int * colonne, int * valeur)
Description : Trouve le minimum dans un tableau et retourne ses coordonnées et sa valeur. Cette fonction recherche le minimum dans un tableau en considérant uniquement les éléments supérieurs ou égaux à zéro. Elle retourne les coordonnées (colonne) et la valeur du minimum trouvé. Si aucun minimum n'est trouvé, la valeur est mise à jour à -1.

Paramètres :
tab Le tableau dans lequel rechercher le minimum.
colonne Variable pour stocker la colonne du minimum.
valeur Variable pour stocker la valeur minimale.

Renvoie : Nombre d'éléments supérieurs à la valeur minimale.

III.2.4.10 max_Tableau()

int max_Tableau (Tableau tab, int* colonne, int* valeur)
Description : Trouve le maximum dans un tableau et retourne ses coordonnées et sa valeur.

Paramètres :
tab Le tableau dans lequel rechercher le maximum.
colonne Variable pour stocker la colonne du maximum.
valeur Variable pour stocker la valeur maximale.

Renvoie : Nombre d'éléments supérieurs à la valeur maximale.

III.2.4.11 remplir_Matrice_Duel()

void remplire_Matrice_Duel (Matrice matrice, Tableau tableau)
Description : Remplit la matrice de duel en fonction des valeurs du tableau.

Paramètres :
matrice La matrice à remplir.
tableau Le tableau contenant les valeurs à utiliser.

III.2.4.12 afficher_Matrice()

void afficher_Matrice (Matrice matrice, FILE* logfile)
Description : Affiche la matrice dans un fichier journal.

Paramètres :
matrice La matrice à afficher.
logfile Pointeur vers le fichier journal.

III.2.4.13 afficher_Tableau()

void afficher_Tableau (Tableau Tableau, FILE* logfile)
Description : Affiche le tableau dans un fichier journal.

Paramètres :
Tableau Le tableau à afficher.
logfile Pointeur vers le fichier journal.

III.2.4.14 trier_Tableau()

void trier_Tableau (Tableau tableau)
Description : Trie un tableau par ordre décroissant.

Paramètres :
tab : Le tableau à trier.

III.2.4.15 afficherVote()

void afficherVote (Tableau tab, FILE* logfile)
Description : Affiche le tableau de vote dans un fichier journal.

Paramètres :
Tableau Le tableau de vote à afficher.
logfile Pointeur vers le fichier journal.

III.2.5 list.h

III.2.5.1 List

typedef struct s_List List
Description : Définition opaque du type List.

III.2.5.2 ptrList

typedef List* ptrList
Description : Définition du type ptrList : pointeur sur un TAD List.

III.2.5.3 Element

typedef struct s_element * Element

III.2.5.4 SimpleFunctor

typedef int(* SimpleFunctor) (int, int, int)
Description : Foncteur simple à utiliser avec la fonction list_map. Ce foncteur reçoit en argument la valeur d'un élément de la liste et doit retourner éventuellement la valeur modifiée de l'élément.

III.2.5.5 ReduceFunctor

typedef int(* ReduceFunctor) (Element, void *)
Description : Foncteur avec des données utilisateur à utiliser avec l'opérateur list_reduce. Ce foncteur reçoit en argument la valeur d'un élément de la liste et un pointeur opaque vers des données fournies par l'utilisateur, et doit retourner éventuellement la valeur modifiée de l'élément.

III.2.5.6 OrderFunctor

typedef bool(* OrderFunctor) (int, int)
Description : Foncteur à utiliser avec l'opérateur list_sort. Ce foncteur doit implémenter une fonction d'ordonnancement total (comp). Lorsque ce foncteur est appelé avec deux éléments de liste a et b, il doit renvoyer vrai si (a comp b).

III.2.5.7 list_create()

List* list_create (void)
Description : Implementation du constructor list.

III.2.5.8 list_push_back()

List* list_push_back (List* l, int p, int candidat1, int candidat2)
Description : Implementation du constructeur push_back.

Paramètres :
l La liste à modifier.
p Le poids du candidat candidat1.
candidat1 La valeur du candidat a.
candidat2 TLa valeur du candidat b.

Renvoie : La liste modifiée.

III.2.5.9 list_delete()

void list_delete (ptrList * l)
Description : Libère les ressources allouées par les constructeurs.

Paramètres :
l L'adresse de la liste.

III.2.5.10 list_push_front()

List* list_push_front (List* l, int p, int candidat1, int candidat2)
Description : Ajoute un élément au début de la liste.

Paramètres :
l La liste à modifier.
p Le poids du candidat candidat1.
candidat1 La valeur du candidat a.
candidat2 La valeur du candidat b.

Renvoie : La liste modifiée.

III.2.5.11 list_pop_front()

List* list_pop_front (List* l)
Description : Supprime un élément au début de la liste.

Renvoie : La liste modifiée.

III.2.5.12 list_pop_back()

List* list_pop_back (List* l)
Description : Supprime un élément à la fin de la liste.

Renvoie : La liste modifiée.

III.2.5.13 list_insert_at()

List* list_insert_at (List* l, int pos, int p, int candidat1, int candidat2)
Descritpion : Insérer un élément à la position donnée.

Paramètres :
l La liste à modifier.
pos La position de l'élément à insérer.
p Le poids du candidat candidat1.
candidat1 La valeur du candidat a.
candidat2 La valeur du candidat b.

Renvoie : La liste modifiée.

III.2.5.14 list_remove_at()

List* list_remove_at (List* l, int p)
Description : Supprime l'élément à la position donnée.

Paramètres :
l La liste à modifier.
p La position de l'élément à supprimer.

Renvoie : La liste modifiée.

III.2.5.15 list_reduce()

List* list_reduce (List * l, ReduceFunctor f, void * env)
Description : Apply the same operator on each element of the list gieven a user define environment.

Paramètres :
l The list to process.
f The operator (function) to apply to each element

Renvoie : The eventually modified list.

III.2.5.16 list_is_empty()

bool list_is_empty (List* l)
Description : Vérifie si la liste est vide.

III.2.5.17 list_size()

int list_size (List* l)

Renvoie : le nombre d'élements de la liste.

III.2.5.18 triee_liste_decroissant()

int triee_liste_decroissant (List * l)
Descritpion : Trie une liste dans l'ordre décroissante.

Paramètres :
l La liste à trier.

Renvoie : 1 si le trie est réussi, -1 autrement.

III.2.5.19 ListIterator

typedef struct s_ListIterator* ListIterator
Description : Définition opaque du type abstrait de données ListIterator.

III.2.5.20 list_iterator_create()

ListIterator list_iterator_create (List * d, unsigned char w)
Description : Constructeur d'un itérateur.

Paramètres :
d La liste à itérer.
w sens de l'itérateur (FORWARD_ITERATOR or BACKWARD_ITERATOR)

Renvoie : Un itérateur correctement initialisé.

III.2.5.21 list_iterator_delete()

void list_iterator_delete (ListIterator it)
Decription : Destructeur d'un itérateur.

Paramètres it L'itérateur à supprimer

III.2.5.22 list_iterator_next()

Element list_iterator_next (ListIterator it)
Description : Incrémente l'itérateur à la position suivante en fonction de sa direction.

Paramètres : it l'itérateur à modifier.

Renvoie : l'itérateur modifié.

III.2.5.23 list_iterator_value()

Element list_iterator_value (ListIterator it)
Description : Accède à la valeur de l'itérateur.

Paramètres : it it l'itérateur à accéder.

Renvoie : la valeur désignée par l'itérateur.

III.2.5.24 list_iterator_has_next()

int list_iterator_has_next (ListIterator it)
Description : Renvoie s'il y a encore des éléments à parcourir.

III.2.5.25 list_iterator_begin()

ListIterator list_iterator_begin (ListIterator it)
Description : Renvoie l'élément sur lequel l'itérateur commence à itérer.

III.2.5.26 list_iterator_delete_current()

int list_iterator_delete_current (ListIterator it)
Description : Supprime l'élément courant de l'itérateur.

III.2.5.27 list_iterator_end()

int list_iterator_end (ListIterator it)
Description : Renvoie si l'itérateur a atteint la fin de la liste.