Université Paul Sabatier
TOULOUSE III

L2 Informatique **Projet S3**

REV party
(République Expérimentale Virtuelle)
Condorcet et les procédures de vote

Octobre 2018

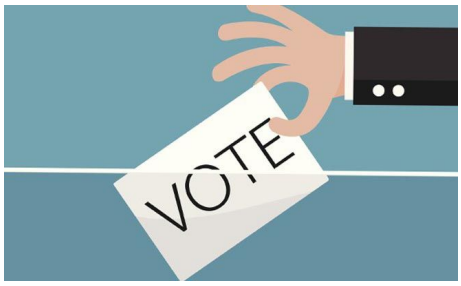

Table des matières

Après une présentation générale du projet, vous trouverez la partie intitulée "Travail à faire : cahier des charges et spécifications" qui précise exactement ce qu'il faut programmer. Prenez le temps de bien lire cette partie. Le respect des spécifications sera pris en compte dans l'évaluation (Cf. document de gestion de projet). Certains aspects sont (involontairement) flous. Ça n'est pas si rare dans un cahier des charges. Vous devez demander l'interprétation de ces points à votre tuteur.

Certaines parties du sujet donneront lieu à des présentations en amphi d'environ 30mn.

1 Présentation

De nombreux pays, entreprises, organisations diverses recourent au vote pour désigner des délégués, représentants, dirigeants, faire des choix divers. Il existe de nombreux modes de scrutins différents. Ces différences portent sur les modes opératoires mais aussi sur les protocoles et modes de calculs. Ces deux derniers points ont fait, et font toujours, l'objet de travaux mathématiques visant à en cerner les propriétés.

Au final, les modes opératoires et modes de calculs des scrutins ne sont rien de plus que des algorithmes que nous allons implémenter en langage C.

1.1 Exemple d'introduction

Tiré de :

`http://http://images.math.cnrs.fr/La-democratie-objet-d-etude.html`

Prenons l'exemple d'un pays imaginaire : la Sudie méridionale qui est une république démocratique. La Sudie consulte ses citoyens pour le choix d'une nouvelle capitale. Trois villes ont déposé leur candidature : Superrural, Nostrae-civitatis et Yolo. Les sondages indiquent que les électeurs ont tendance à voter pour la ville la plus proche de chez eux. Ainsi [voir la carte ci-dessous], les habitants de N, à défaut de voir leur ville choisie, préfèreraient S à Y, les habitants de S préfèreraient Y à N, et les habitants de Y préfèreraient S à N. Les autres habitants suivent cette logique selon leur proximité de ces villes.

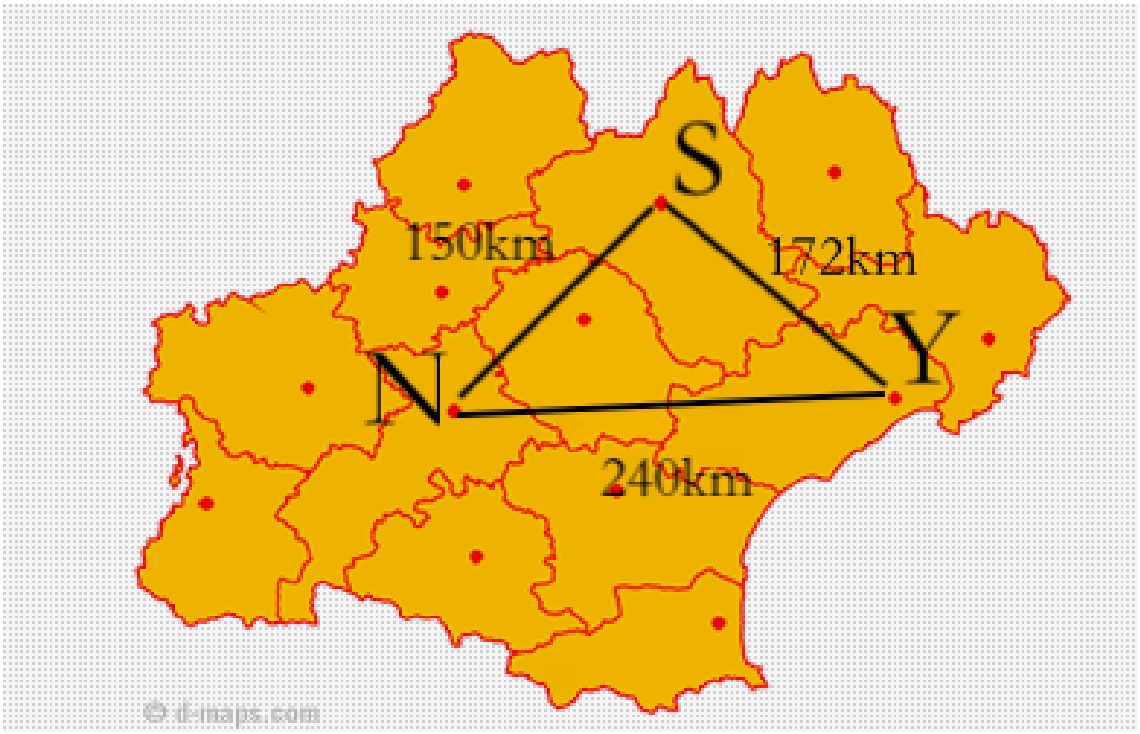

FIGURE 1 – Carte de la Sudie méridionale avec les distances des villes candidates.

Si la Constitution dit bien que c'est par une élection démocratique que doit être désignée la capitale, elle ne précise en revanche pas quelle méthode électorale doit être suivie. Plusieurs méthodes concurrentes, toutes clairement considérées comme démocratiques, ont été proposées :

- Méthode A : Chaque électeur vote pour une ville ; la ville qui reçoit le plus de suffrages gagne (Uninominal à un tour).
- Méthode B : Dans un premier tour, chaque électeur vote pour une ville, puis un second tour est organisé entre les deux villes ayant reçu le plus de suffrages au premier tour ; la ville qui reçoit le plus de suffrages au second tour gagne (uninominal à deux tours).
- Méthode C : Chaque électeur classe les villes par ordre de préférence ; sa ville préférée marque 2 points, la suivante 1 point et la dernière aucun (Moodle, évaluation par les pairs). La ville qui obtient le meilleur total gagne.

Bien sûr il y a bien d'autres possibilités (possibles ou effectivement utilisées quelque part sur le planète). On peut se dire que peut-être elles donnent en fait la même ville vainqueuse... Hélas, non. Imaginons par exemple que 40 % des électeurs choisissent N, 35 % choisissent Y et 25 % choisissent S. Alors :

- Suivant la méthode A, N gagne par 40 % des suffrages contre 35 % pour Y et 25 % pour S.
- Suivant la méthode B, Y gagne au second tour contre N par 60 % des suffrages contre 40 %, vu que les habitants de S se reportent sur Y au second tour (en supposant un report de voix parfait).
- Suivant la méthode C, S gagne avec une moyenne de 1,25 points par électeur (2 points de la part des habitants de S, et 1 point de ceux de N et de Y), suivi de Y avec 0,95 points (2 points de la part des habitants de Y, 1 point de ceux de S, et aucun de ceux de N), et de N avec 0,80 points (2 points de la part des habitants de N, et aucun de ceux de Y et de S).

Du coup, quelle méthode choisir ? Laquelle est la plus démocratique, la plus juste, la plus susceptible de

minimiser le nombre de mécontents ?

Le décors étant planté, nous allons par la suite détailler plusieurs méthodes de scrutins afin d'en cerner les avantages et inconvénients et de les programmer.

1.2 Election du roi des pirates

Afin d'avoir de la matière pour nos calculs qui ne soit pas limités à des jeux de données de tests, nous allons commencer par organiser un vote au moyen d'un sondage Moodle. C'est un vote "pour jouer", mais il sera organisé comme un vrai vote. Le prétexte est d'élire le roi des pirates de l'animé "One piece" de Eiichiro Oda (尾田 栄一郎).

Le sondage consiste à classer les dix candidats selon une échelle de Likert en classant par ordre de préférence de 1 (le préféré) à 10 (le moins aimé). Il peut y avoir des ex-equo.

Une fois le sondage clos, la base de données des réponses (votes) peut être exportée (par le staff) au format csv (comma separated values). En fait, Moodle sépare les données avec des "tab".

Le fichier débarrassé des informations inutiles sera alors proposé au téléchargement sur Moodle. Ce fichier contenant les ballots du vote par classement préférentiel sera la donnée de vos programmes. C'est donc vous qui allez dire qui gagne l'élection selon les différentes méthodes.

2 Les méthodes de scrutins

2.1 Scrutin uninominal

2.1.1 Uninominal à un tour

C'est la méthode la plus simple à mettre en oeuvre. C'est celle qui est utilisée pour l'élection de vos délégués de groupes de CTD et de TP.

Principe : chaque électeur vote pour un et un seul candidat. A la fin du vote, on compte le nombre de voix de chacun des candidats. Celui qui a obtenu le plus grand nombre de voix l'emporte.

Avantages : très simple à mettre en oeuvre, peu de calculs.

Inconvénients : un candidat peut être élu avec un pourcentage assez faible des suffrages, ce qui questionne sa légitimité. De plus il risque d'avoir contre lui une majorité de mécontent et ce dès le début.

2.1.2 Uninominal à deux tours

C'est le système qu'on emploie en France pour, entre autre, élire le président de la république. Come précédemment, chaque électeur vote pour un et un seul candidat. A la fin du vote, on compte le nombre de voix de chacun des candidats. Les deux candidats qui on eu les plus grands nombres de voix restent en compétition pour un second tour (sauf si un candidat a obtenu plus de 50% des voix auquel cas il est élu).

A l'issu du second tour, le candidat qui a obtenu le plus de voix l'emporte.

Avantages : plus de légitimité pour celui qui est élu

Inconvénients : crée des phénomènes de vote utile, de stratégie de reports de voix et peut favoriser l'abstention au second tour.

2.2 Méthodes de Condorcet

`https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Condorcet`

Dans son *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*, le marquis de Condorcet, mathématicien et philosophe français du XVIIIe siècle, met en évidence le fait que le vote à la pluralité (uninominal à un tour) peut très bien ne pas représenter les désirs des électeurs.

Il proposa une méthode, dite méthode de Condorcet (aussi appelée scrutin de Condorcet ou vote Condorcet) qui est un système de vote dans lequel l'unique vainqueur, s'il existe, est le candidat qui, comparé tour à tour à tous les autres candidats, s'avérerait à chaque fois être le candidat préféré, c'est-à-dire le vainqueur de Condorcet.

Un scrutin de Condorcet obéit au principe de Condorcet, qui se veut être le principe démocratique le plus naturel possible, à savoir que « si un choix est préféré à tout autre par une majorité, alors ce choix doit être élu. » En effet, ce principe semble logique dans le cadre d'un scrutin démocratique. Il signifie autrement dit que, lors d'une élection, si une majorité d'électeurs préfère un candidat à tous les autres candidats, alors ce candidat préféré est celui qui doit être désigné vainqueur du scrutin.

Rien ne garantit la présence d'un candidat satisfaisant au critère de victoire. Ainsi, tout système de vote fondé sur la méthode comparative de Condorcet doit prévoir un moyen de résoudre les votes pour lesquels ce candidat idéal n'existe pas.

L'option (ou le candidat) désigné vainqueur par la méthode de Condorcet est appelé vainqueur de Condorcet.

Exemple théorique

Considérons une assemblée de 60 votants ayant le choix entre trois propositions a, b et c. Les préférences se répartissent ainsi (en notant $a > b$, le fait que a est préféré à b) :

— 23 votants préfèrent : $a > c > b$;
— 19 votants préfèrent : $b > c > a$;
— 16 votants préfèrent : $c > b > a$;
— 2 votants préfèrent : $c > a > b$.

Dans une procédure de vote pluraliste, a l'emporte avec 23 voix, sur b avec 19 voix et sur c avec 18 d'où $a > b > c$.

Dans les comparaisons majoritaires par paires, on obtient :
— 35 préfèrent $b > a$ contre 25 pour $a > b$;
— 41 préfèrent $c > b$ contre 19 pour $b > c$;
— 37 préfèrent $c > a$ contre 23 pour $a > c$.

Ce qui conduit à la préférence majoritaire $c > b > a$, l'opposé exact du résultat obtenu avec le choix pluraliste (uninominal à un tour).

On peut représenter ces résultats avec une matrice des duels :

	A	**B**	**C**
A	0	25	23
B	35	0	19
C	37	41	0

Cette matrice à son tour peut être représentée par un graphe orienté dont les candidats sont les sommets. Un arc (a, b) où a et b sont des candidats signifie a est préféré à b (a domine b dans le langage des graphes). On peut de plus associer à cet arc le pourcentage des préférences ou les nombres de voix. L'arc devient un arc étiqueté (a, b, p) où p est un nombre.

On remarque qu'on ne fait figurer dans la graphe que les arcs gagnants. Ce graphe est le plus souvent un graphe de tournoi (cf. annexe).

Déroulement du vote

Chaque électeur classe les candidats par ordre de préférence. Ce vote s'appelle un "ballot".

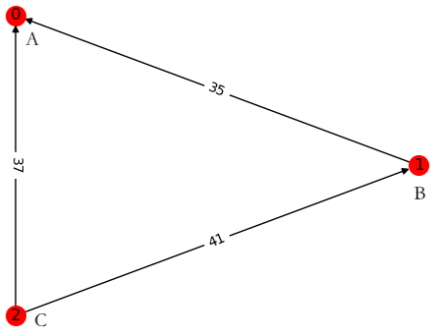

FIGURE 2 – Graphe de l'exemple théorique.

Décompte des votes

Le dépouillement du scrutin consiste à simuler l'ensemble des duels possibles : pour chaque paire de candidats, on détermine le nombre d'électeurs ayant voté pour l'un ou l'autre en vérifiant, sur chaque bulletin de vote, comment l'un était classé par rapport à l'autre. Ainsi pour chaque duel, il y a un candidat vainqueur. Dans la plupart des cas, il y a un unique candidat qui remporte tous ses duels : il s'agit du vainqueur du scrutin. La section suivante décrit ce qui se passe dans les rares cas où aucun candidat ne remporte tous ses duels.

2.2.1 Vainqueur de Condorcet

Définition : Si, parmi les candidats à une élection, il en existe un qui, face à n'importe quel autre, lui est préféré par une majorité d'électeurs, alors ce candidat est appelé le vainqueur de Condorcet.

Définition : On dit qu'une méthode électorale satisfait le critère de Concorcet quand, lorsqu'il y a un vainqueur de Condorcet, c'est toujours lui que cette méthode déclare vainqueur (sous réserve que les électeurs aient voté selon leurs préférences véritables).

Théorème : Il ne peut pas y avoir plus d'un vainqueur de Condorcet à une élection.

Dans le graphe des duels un vainqueur de Condorcet aura la particularité de n'avoir que des arcs sortants. On dit qu'il domine tous les autres sommets. S'il y a un conflit alors aucun sommet ne domine tous les autres et le graphe comporte un circuit.

Le site ci-dessous fait des simulations intéressantes :

`https://jorisdeguet.github.io/PrefVote/tideman.html`

Les méthodes suivantes proposent des solutions pour trouver un vainqueur dans la cas où il y a un conflit.

Avantages : les électeurs ont plus tendance à voter selon leurs vraies préférences. Le vainqueur à la légitimité de la préférence majoritaire sur tous les autres candidats.

Inconvénients : plus difficile à mettre en place et à dépouiller.

Résolution des conflits

Il arrive qu'aucun candidat ne soit élu à la suite du décompte des votes : Condorcet avait remarqué cet important paradoxe inhérent à la méthode, appelée le paradoxe de Condorcet : dans une élection dont les candidats sont A, B et C, dès que le nombre des électeurs est supérieur à deux, A peut être préféré à B, lui-même préféré à C, lui-même préféré à A. Plusieurs méthodes sont utilisables pour résoudre ce conflit de circularité.

Remarque : S'il y a un vainqueur de Condorcet, auquel cas la méthode de résolution de conflit n'est pas appliquée et le vainqueur du scrutin sera le vainqueur de Condorcet. S'il y a un conflit, une méthode de résolution de conflit est appliquée.

2.2.2 Résolution du paradoxe : la méthode du minimax

Le protocole est le suivant : pour chacun des candidats, on regarde le pire des scores qu'il ferait dans ses différents faces-à-faces, puis on déclare élu celui des candidats dont le pire des scores est le meilleur. Cette méthode est appelée méthode minimax, car elle regarde qui est le meilleur (« max ») dans la situation où il est le moins bon (« mini »).

Exemple : voir :

`http://images.math.cnrs.fr/Et-le-vainqueur-du-second-tour-est.html`

Inconvénients : peut être biaisée par des stratégies entre partis.

2.2.3 Résolution du paradoxe : rangement des paires par ordre décroissant

`fr.wikipedia.org/wiki/Méthode_Condorcet_avec_rangement_des_paires_par_ordre_décroissant`

Pour cette méthode lisez la page Wikipédia qui en parle très bien.

La recherche des circuits du graphe se fera grâce à un algorithme de recherche en profondeur modifié. Il correspond à la librairie à télécharger sur Moodle. Vous n'avez donc pas à le programmer.

2.2.4 Résolution du paradoxe : la méthode de Schulze

`https://en.wikipedia.org/wiki/Schulze_methoden.wikipedia.org`
`https://fr.wikipedia.org/wiki/M%C3%A9thode_Schulze`

2.3 Vote alternatif

`https://fr.wikipedia.org/wiki/Vote_alternatif`

Afin de na pas avoir un document ridiculement long, les méthodes précédentes seront détaillées dans des tutos en amphi. Lisez les liens donnés. En clair, arrivez aux tutos en ayant déjà réfléchi à la méthode.

3 Travail à faire : cahier des charges et spécifications

Moodle permet des récupérer le résultat d'un sondage (le vote) sous forme d'un fichier CSV contenant les ballots et d'autres informations. Le programme devra construire les structures de données à partir de ce fichier.

3.1 Structures de données

Tous les tableaux employés dans le programme devront être dynamiques. Leur taille sera allouée à la demande avec éventuellement des realloc. Le fichier *squelette.h* donne des structures de données et des prototypes de sous-programmes à utiliser ou implémenter. Les sous-programmes à implémenter sont décrits plus bas.

3.2 Options de la ligne de commande

Le programme doit récupérer les options de la ligne de commande. Ces options sont normalisées sous la forme d'une balise : un tiret "-" suivit d'une lettre, et d'un paramètre de la balise qui est une chaîne de caractères.

Balise	Paramètre	Action
-i	nom de fichier csv	traite le fichier cité
-d	nom de fichier csv	traite le fichier cité
-l optionnel	nom de fichier txt	fichier de log
-m optionnel	méthode	précise la méthode de scrutin à utiliser

où méthode $\in \{uni1, uni2, cm, cp, cs, va\}$ avec :

paramètre	méthode associée
cm	méthode de Condorcet et minimax
cp	méthode de Condorcet et rangement des paires
cs	méthode de Condorcet et méthode de Schulze
va	vote alternatif

Dans l'analyse des options : Si la balise -i est présente, le fichier csv est le résultat d'un vote par classement (ensembles de ballots).

Si la balise -d est présente à la place de -i alors le fichier csv argument est une matrice de duels.

Attention : Les deux options -i et -d sont incompatibles mais l'une des deux doit être présente.

Si la balise -l <nom>.txt est donnée, le fichier <nom>.txt sera considéré comme un fichier de log. Une variable logfp pourra être affectée au résultat de l'ouverture de ce fichier, sinon logfp= stdout. Ce fichier (ou à défaut l'écran) recevra les affichages intermédiaires permettant la vérification du vote (le suivi des calculs).

La balise $-m$ suivie d'un nom de méthode $\in \{uni1, uni2, cm, cp, cs, va\}$ permet d'appliquer une méthode de scrutin. La méthode de Condorcet correspond aux paramètres : $\{cm, cp, cs\}$. Le vote alternatif n'est pas une méthode de Condorcet.

Si l'option $-m$ n'est pas présente, le programme applique toutes les méthodes valides l'une après l'autre et affiche le résultat de chacune.

Si l'option $-d$ est présente, elle désactive l'emploi des argument $uni1$, $uni2$ et va de l'option $-m$ (Ces méthodes ont besoin des ballots).

En cas d'erreur : balise inconnue, paramètre de balise incorrect ou absent, balises incompatibles, fichier inexistant ou mal formé, le programme affichera un message d'erreur et sortira avec un "exit(EXIT_FAILURE)".

Les balises pourront être mises dans un ordre quelconque.

3.3 Principe général du programme

La méthode générale est la suivante :
— Analyse des balises.
— Lire le fichier csv et remplir une matrice de chaînes de type $t\_mat\_char\_star\_dyn$ (pour $-i$ et $-d$). On conserve la ligne des entêtes.

— Créer la matrice des duels à partir du tableau précédent (le traitement tiendra compte de l'option $-i$ ou $-d$). Le type est *t_mat_int_dyn*.
— Créer la liste des arcs correspondants à la matrice (on crée le graphe).
— Créer le fichier python pour afficher le graphe.
— Appliquer la méthode de scrutin sur la matrice des duels ou sur le graphe suivant les méthodes (l'un ou l'autre sont plus naturels suivant les méthodes).
— Afficher les résultats.

Votre programme sera compilé au moyen d'un *Makefile* et l'exécutable portera le nom de *scrutin*.

L'affichage des résultats sera normalisé suivant la forme :
Mode de scrutin : <nom de la méthode>, <nb> candidats, <nb>, votants, vainqueur = <nom du vainqueur>

Dans le cas des scrutins uninominaux on affichera le score en plus :
Mode de scrutin : <nom de la méthode>, <nb> candidats, <nb>, votants, vainqueur = <nom du vainqueur>, score = <nb>%

Pour simplifier les affichages et utiliser le même sous-programme, le scrutin uninominal à deux tours, fera deux affichages pour le tour 1 (un pour chacun des deux candidats de tête), et un affichage pour le second tour.

Exemple :

```
Mode de scrutin : uninominal à un tour, 10 candidats, 100 votants, vainqueur = Astérix, score = 51%

Mode de scrutin : uninominal à deux tours, tour 1, 10 candidats, 100 votants, vainqueur = Astérix, score
Mode de scrutin : uninominal à deux tours, tour 1, 10 candidats, 100 votants, vainqueur = Obélix, score
Mode de scrutin : uninominal à deux tours, tour 2, 2 candidats, 100 votants, vainqueur = Astérix, score

Mode de scrutin : Condorcet minimax, 10 candidats, 100 votants, vainqueur = Astérix

Mode de scrutin : Condorcet paires, 10 candidats, 100 votants, vainqueur = Astérix

Mode de scrutin : Condorcet Schulze, 10 candidats, 100 votants, vainqueur = Astérix

Mode de scrutin : vote alternatif, 10 candidats, 100 votants, vainqueur = Astérix
```

Le fichier log recevra les calculs intermédiaires pour vérification : les matrices, le graphe, les calculs de chaque méthode. (le format est libre).

3.4 Sous programmes utilitaires

Il faut implémenter les prototypes de sous-programmes définis dans le fichier "squelette.h". Notamment :
— faire une fonction qui crée un tableau d'entier dynamique à une dimension passée en paramètre (voir squelette.h)
— faire une fonction qui crée un tableau d'entier dynamique à deux dimension passées en paramètre (matrice carrée).
— faire une fonction qui crée un tableau de chaîne de caractères dynamique à deux dimension passées en paramètre. 🔥 (C'est un peu plus chaud)
— faire des procédures qui utilisent les fonctions précédentes dans les *struct*.
— faire une fct qui lit un fichier csv (txt), détecte le nombre de lignes *nbRows* et de colonnes *nbCol* en utilisant un caractère délimiteur passé en paramètre (le tab a priori) et renvoie l'adresse d'un tableau

de chaînes de caractères de $nbrows \times nbCol$ cases contenant chacun des éléments du csv. 🔥 🔥 (Spicy). Il faut faire des realloc.

Remarque : On s'intéressera à la fonction *strtok* de *string.h* pour rechercher les délimiteurs.

3.5 Représentation graphique du graphe des duels

Il existe en Python un package pour gérer et dessiner des graphes. Nous allons l'utiliser pour afficher les graphes de duels. Il faut utiliser le fichier *polTests.py* (donné sur Moodle) et remplacer la ligne de commentaire "écrire les arcs ici" par autant de lignes que nécessaire du type :

```
G.add_edges_from([(i,j)],weight=p)
```

où i et j sont des sommets et p le poids. En clair il faut faire une boucle sur les i, j et p en utilisant la matrice des duels.

Le mieux est de sauver le programme Python modifié dans un nouveau fichier *.py*. On affiche le graphe grâce à la commande "python nouveaufichier.py".

En clair :
- Lire le fichier *polTests.py* et recopier les lignes telles qu'elles dans un autre fichier *.py*
- A la place de la ligne de commentaire Python : "écrire les arcs ici",

```
écrire le texte "G.add_edges_from(["
écrire i,
écrire ","
écrire j,
écrire ")],weight=",
écrire p,
écrire enfin ")"
```

- Répéter le point précédent autant de fois qu'il y a d'arcs (i, j, p) dans le graphe de duels.
- finir de recopier le fichier *polTests.py* dans le nouveau fichier *.py*.

3.6 Liste ordonnée

A partir de votre expérience de pile statique en TP d'algorithmique, créez une liste statique circulaire combinant les possibilités d'une pile et d'une file. On doit pouvoir, entre autre, ajouter ou supprimer un élément en tête ou en queue. Il faut implémenter toutes les fonctions et procédures de "pile.h".

Vous allez gérer deux fichiers .c : element_liste.c et liste.c (et les .h qui vont bien avec et qui sont donnés). En guise de spécification vous utiliserez les fichiers header fournis sur la page Moodle. Le code C devra implémenter les sous-programmes correspondant aux prototypes. Les fichiers *header* ne devrons pas être modifiés. Le type *Element* est défini à partie du type t_arc_p lui-même définit dans *global.h*. Le programme principal fourni servira de test.

La liste d'arcs sera utilisée dans les méthodes minimax et classement des paires où il faudra chercher les circuits. Un code est fourni "circuit.c", donné aussi sous forme de librairie 'libcircuit.a". Dans tous les cas, vous devrez ajouter "bool circuits(liste larcs,int nbCandidats) ;" dans votre header.

larcs est une liste d'arcs représentant le graphe, *nbCandidats* est le nombre de sommets du graphe. Ce nombre doit correspondre aux nombre de candidats du scrutin. La fonction renvoie *true* si le graphe contient un circuit, $false$ sinon.

Pour compiler avec la librairie "libcircuit.a" : "gcc listeDeVosFichiers.c -L. -lcircuit"

3.7 Liste des modules à programmer

— fichier utils_sd : allocation dynamique de tableaux, affichage de ces tableaux avec logfp, sou-programme utilitaires de type chercher le min, le max etc.,
— module liste : element_liste.c, liste.c et la gestion d'une liste
— module uninominales : les deux méthodes uninominales et leurs sous-programmes
— module condorcet_minimax : cherche un vainqueur de Condorcet, applique minimax si besoin
— module condorcet_paires : cherche un vainqueur de Condorcet, applique le classement des paires si besoin
— module condorcet_schulze : cherche un vainqueur de Condorcet, applique la méthode de Schulze si besoin
— module lecture_csv : lecture des fichier, initialisation des structures de données
— module graphe_python : construction du programme Python affichant le graphe des duels
— module main : récupération des arguments de la ligne de commande, analyse des balises
— fichier Makefile
— vote alternatif

A ce découpage on peut éventuellement ajouter un module utils_scrutins qui regroupera des sous-programmes communs aux différentes méthodes de scrutin (optionnel).

3.8 Evaluation des modules

A chaque module est associé un nombre maximal de points à gagner. On rappelle ici (cf. document de gestion de projet) que pour obtenir le maximum des points d'un module il faut que :

1. Le code respecte les spécifications et fonctionne sans bug,
2. Le code respecte les principes du *clean code* et soit commenté d'une manière utile (informative et concise)

Nom	**Points/20**
utils_sd	1,5
listes	2
lecture csv	2
graphe python	1,5
uninominales	2
condorcet minimax	2
condorcet paires	3
condorcet schulze	3
vote alternatif	2
main et balises	2
makefile	1
Bonus	1
Total	23

Le bonus est à la discrétion du tuteur. Il récompense un "plus" comme l'utilisation de git, des shell de test, un makefile poussé, etc.

3.9 Qualité du code

3.9.1 Clean code

La notion de *clean code* recouvre toute les bonnes pratiques pour avoir un code lisible et compréhensible. Les règles sont le plus souvent des règles de bon sens éprouvées par l'expérience. Les développeurs ont souvent tendance à les sacrifier à l'urgence du moment car :

« I have a deadline, I don't have time for clean code »

Voici certains des dix principes pour un code propre de Michael Toppa (un développeur parmi d'autre) :

`https://www.youtube.com/watch?v=ls4iAtOCH8g (en anglais)`

Les slides de la présentation :

`https://www.slideshare.net/mtoppa/wordcamp-nashville-clean-code-for-wordpress`

Selon lui : Clean code...
— "Does one thing well" - Bjarne Stroustrup
— "Reads like well written prose" - Grady Booch
— "Can be read and enhanced by a developer other than its original author" - Dave Thomas
— "Always looks like it was written by someone who cares" - Michael Feathers
— "Contains no duplication" - Ron Jeffries
— "Turns out to be pretty much what you expected" - Ward Cunningham

Ne pas oublier cependant que "l'excès de rigueur nuit à l'efficacité et à la lisibilité". Par exemple si des noms de variables trop courts peuvent ne pas aider à comprendre un programme, des noms trop longs auront le même travers.

3.9.2 Les tests

Il est conseillé de tester les sous-programmes dès que possible (tests unitaires), et de ne les assembler entre eux que quand ils sont raisonnablement débogués. Après assemblage il faut tester ce nouveau code (tests d'intégration) et ainsi de suite avec les assemblages d'assemblages.

Des shells de test sont fournis pour vous aider à valider les codes des méthodes.

3.9.3 Documentation

Plutôt que de rédiger des doc papier nous allons utiliser Doxygen qui permet de compiler les commentaires du code, du moins ceux qui sont précédés des balises reconnues par le soft.

Il y a plusieurs façons de gérer ces commentaires. La plus simple est de commencer le commentaire par trois "slash" "///" suivis de la balise elle-même précédée d'un anti-slash.

Les balises les plus utiles sont :

```
\author  l'auteur
\date la date
\struct to document a C struct.
\union to document a union.
\enum to document an enumeration type.
\fn to document a function.
\var to document a variable or typedef or enum value.
\def to document a define.
\typedef to document a type definition.
\file to document a file.
\brief pour décrire ce que fait le programme
\param[in] to document an incoming parameter
\param[out] to document an outcoming parameter
\param[in,out] to document an in-out parameter
\return to document the return value of a function
\pre preconditions
\post postconditions
```

```
  \invariant invariants
  \note to print "note" before the comment
  \remarks to print "remarks" before the comment
```

Exemple :

```
/// \brief Fonction factorielle
/// \author Gérard Menvuuncodesibo
/// \date 9 octobre 2018
long fact(int n){
/// \param[in] n un entier positif ou nul
/// \return n!
if (n==0)
         return 1;
else
         return n*fact(n-1);
}
```

La doc est générée sous forme de documents *html* consultables avec un navigateur. Si les balises de l'auteur ou de la date sont à mettre seulement en début de fichier, il peut être intéressant de de commenter les paramètre, les valeurs de retour, et de donner une indication sur le rôle des sous-programmes. Sans abus là aussi.

4 Annexes

4.1 Format CSV

CSV signifie "comma separated values". Ce format permet d'écrire un ensemble de données de type feuille de calcul dans un fichier texte. La première ligne donne les entêtes de colonnes. Les lignes suivantes donnent les valeurs. Les lignes doivent avoir le même nombre de valeurs (ou au moins de virgules). Ce type de fichier est reconnu par les tableau qui construisent la feuille de calcul correspondante.

Les valeurs peuvent être séparées par autre chose qu'une virgule (tab, point virgule, espace, etc).

Les fichiers au format csv ne sont que des fichiers texte. Ils peuvent être lus, créés, modifiés sous éditeur.

4.2 Le programme de simulation de votes

Ce programme est donné. Il est programmé en Python et il permet de générer des jeux de données en simulant un vote de classement des candidats. Il génère un fichier csv du même type que celui donné par Moodle où figure les ballots. Il prend en argument un fichier texte où figurent les noms des candidats, le nombre de votes souhaités, et donne en sortie la simulation sous forme d'un fichier texte (csv) où le séparateur est un tab.

Utilisation :

```
python votation.py -i candidats.txt -v nombre_de_votants -o fichier.csv
```

où le fichier *candidats.txt* est un fichier texte donnant la liste des candidats (un par ligne).

4.3 Graphes orientés et graphe de tournoi

Rappel du cours de maths discrètes de L1 :

Un graphe orienté $G = (V, A)$ est un couple formé de V un ensemble de sommets et A un ensemble d'arcs, chaque arc étant un couple de sommets ($A \subset V \times V$).

Définitions : Étant donné un arc (x, y), on dit que x est l'origine (ou la source) de (x, y) et que y est l'extrémité (ou la cible) de (x, y). On peut le représenter par une flèche allant de x vers y.

Le demi-degré extérieur (degré sortant) d'un nœud, noté $d^+(x)$, est le nombre d'arcs ayant ce nœud pour origine.

Le demi-degré intérieur (degré entrant) d'un nœud, noté $d^-(x)$, est le nombre d'arcs ayant ce nœud pour extrémité.

Autres définitions utiles :
- Chaque arc ayant une seule origine et une seule extrémité, le graphe comporte autant de degrés sortants que de degrés entrants : $\sum_{x \in V} d^+(x) = \sum_{x \in V} d^-(x)$.
- $x_1x_2, x_2x_3, \cdots, x_{n-1}x_n$ est un chemin si et seulement si $\forall p \in \{1, 2, \cdots, n-1\}, (x_p, x_{p+1})$ est un arc.
- Le chemin $x_1x_2, x_2x_3, \cdots, x_{n-1}x_n$ est un circuit si et seulement si (x_n, x_1) est un arc.
- Un graphe complet orienté antisymétrique est un tournoi : $\forall x, y \in V, (x, y) \in A$ où $(y, x) \in A$
- On peut définir des graphes orientés étiquetés comme un graphe orienté et une fonction $f : A \to N$ qui a chaque arc associe une nombre entier. Ce nombre est le poids de l'arc.

Le moyen le plus simple de représenter un graphe orienté dans le cadre de ce cahier des charges, est de faire une liste de ses arcs. Pour les méthodes minimax, paires et Schulze, les calculs pourront être faits directement sur le graphe, c'est à dire en analysant la liste des arcs.

4.4 Trucs et astuces

- `grep -v "///" *.c *.h|wc -l` pour estimer le nombre de lignes de votre code. Ça peut servir pour la recette, un CV, un entretien de stage, etc.
- Il est furieusement conseillé de faire des sauvegardes de votre code, et pas sur le même disque dur (un disque dur ça peut crasher).
- Il est aussi conseillé de faire des copier/coller des codes intermédiaires qui marchent (archives de versions). Cela permet d'avoir quelque chose qui marche lors de la validation. Méfiez-vous du "je change un petit truc vite fait" à dix minutes de la recette.

4.5 Projets de tutos en amphi

Ces tutos dureront environ 30mn. Ce sont dans l'ordre :
- Framaboard
- Doxygen
- Affichage d'un graphe en Python
- Méthode de Condorcet avec classement des paires
- Méthode de Condorcet et Schulze
- Vote alternatif

Certains tutos pourront aborder plusieurs thèmes.