Systèmes de vote

Alexis Delin
Mathieu Leroux
Léo Oger
Gabriel Rouge

Introduction

Ce document résume le fruit de notre travail d'une semaine, ainsi que des travaux préliminaires que nous avons fournis dans l'intérêt d'augmenter notre efficacité et de pouvoir rendre un devoir dont nous sommes satisfaits en temps et en heure. On pourra ainsi décrire le fruit de nos efforts en 5 parties, où quatre parties concernent chaque système de votes que l'on a choisi, et la cinquième concerne toutes les fonctions annexes à ces quatres parties. L'on a choisi d'écrire les variables et commentaires en anglais afin d'éviter tout souci liés à une potentielle sensibilité à la casse, et puisque l'anglais au sein de la programmation est une langue plus professionnelle qui rend le code universel. Bien entendu, une sémantique et une rigueur étaient recherchées dans la nomination des variables et la rédaction des commentaires pour rendre le programme agréable à lire et simple à comprendre aux yeux d'un individu compétent.

Partie 1 : Vote par approbation - Gabriel Rouge

Dans le cadre de cette partie du projet, nos travaux préliminaires concernaient la prise en connaissance dudit système de vote, la décision de l'architecture que prendraient les fonctions en s'articulant entre elles, et la contenance de celles-ci par le biais d'algorithme sur papier qui ont permis de commencer rapidement la programmation en C++ puisqu'il ne suffisait que de traduire un programme déjà fait.

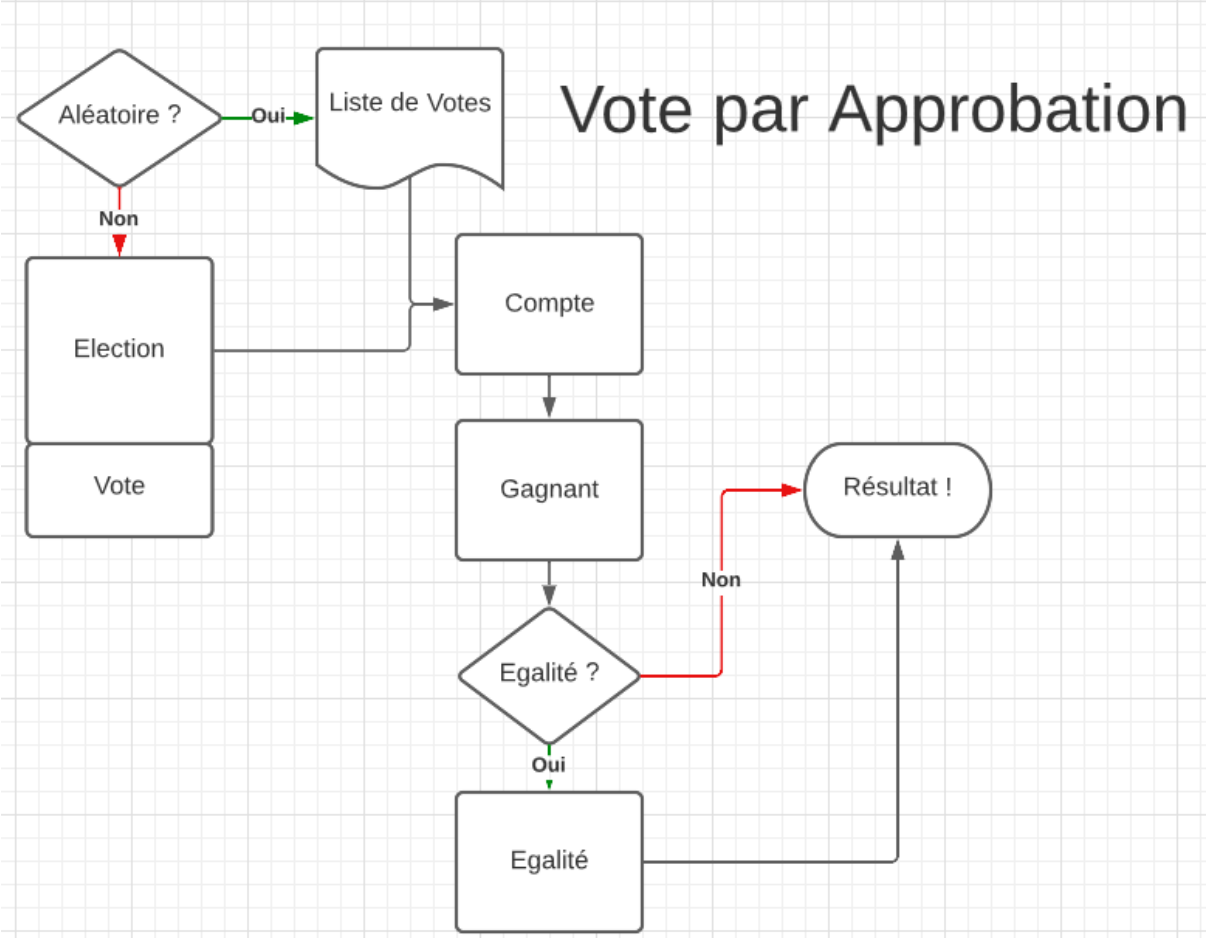
Ce système de vote a été utilisé en République de Venise ainsi qu'en Angleterre au 19e siècle. Il est aujourd'hui utilisé dans plusieurs associations réputées, dans certaines villes ou petits villages, et dans certains pays d'Europe de l'Est. Il est assez apprécié, si bien que plusieurs groupes aimeraient le voir remplacer le système de vote mis en place dans leur pays par le vote par approbation. On retrouve ce mouvement par exemple aux Etats-Unis.

Voici comment nous avons conceptualisé le vote par approbation ; chaque individu, au moment de voter, attribue une note entre 0 et 1 à chaque candidat. 0 correspond à une désapprobation, et 1 correspond à une approbation. Un vote prend alors la forme d'un vecteur qui contient des 0 et des 1, avec l'index de celui renvoyant à l'index des candidats.

Partant de cela, il a été créé une fonction qui compte le nombre de voix que chaque candidat a reçu, puis une fonction qui prend en paramètre ce résultat pour renvoyer le candidat qui a remporté l'élection, et enfin une fonction récursive appelée en cas d'égalité pour donner tous les candidats remportant l'élection. A posteriori, l'on a remarqué avec une agréable surprise que ces deux dernières fonctions étaient applicables à tous les systèmes de votes dans le décompte de voix.

Deux autres fonctions ont été créées, qui ont permis une résolution de bug plus simple. Il s'agissait d'une fonction qui, sur base d'aléatoire, simulait un vote unique, et une autre fonction qui se chargeait d'appeler cette dernière autant de fois que nécessaire pour simuler toute l'élection par l'aléatoire, pour enfin stocker les résultats dans la liste de votes qui est utilisée dans le reste des fonctions.

Pour fin d'une meilleure compréhension, voici un schéma de ce à quoi ressemble notre architecture du vote par approbation :



Explication :

- Soit, on met en entrée une liste de votes, ou bien, on utilise la fonction élection pour en créer une de manière aléatoire. Cette fonction utilise une sous fonction Vote.
- La fonction Compte fait le décompte des voix pour chaque candidat.
- La fonction gagnant renvoie le gagnant s'il est unique, ou appelle la fonction égalité si nécessaire.
- Voilà le résultat !

Ce schéma est applicable à la plupart des votes, et certains en font usage d'une variante légère. Ce qui est notable ici est la fonction qui crée une liste de votes aléatoire, n'étant pas nécessairement utilisée, et la manière dont ce système de vote prend en main les cas d'égalités ; celui-ci permet l'existence de plusieurs gagnants.

Les fichiers de tests correspondant à ce vote sont au nombre de 4 ; l'un ne contient que des 0, l'autre que des 1, un troisième renvoie une égalité, et un dernier un gagnant unique avec une seule voix de plus que les autres.

Partie 2 : Le scrutin uninominal majoritaire à un tour - Léo Oger

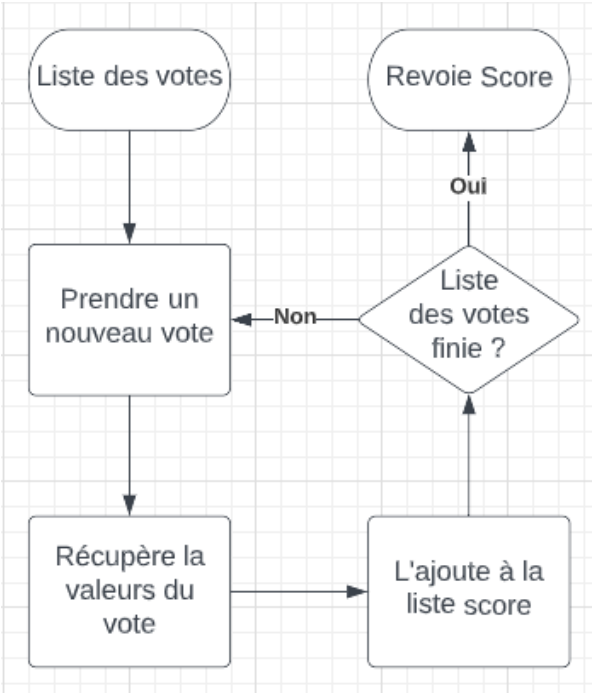
Dans cette partie du projet, nous allons voir comment nous avons programmé le scrutin uninominal majoritaire ou First Past The Post en anglais. Il est beaucoup utilisé dans la politique, notamment durant les élections présidentielles en France. Il est simplement divisé en deux tours, un tour où l'on garde les deux candidats ayant le plus de voix, et un autre ou l'on élit le président. Mais celui que l'on fait exactement se déroule en un seul tour et est utilisé pour élire un président en Asie, en Amérique du Sud, en Afrique et même au Royaume-Uni, aux États-Unis et au Canada, dans le cadre d'une élection parlementaire ou législative.

Le vote est constitué de 0 et de 1 et chaque individu peut voter pour un seul candidat. Il ne faut donc que des 0 et un seul 1 dans le bulletin de vote. Le vote prend alors la forme d'un vecteur de 0 et de 1 avec le même index que celui des candidats.

La première fonction créée a été isEntryValid_Unique, une fonction qui permet de vérifier si un vote est valide ou non, en regardant s'il fait la bonne taille et s'il comporte bien qu'un seul choix. Mais sachant par la suite que les votes seront tous justes, nous avons donc mis cette fonction de côté et nous nous sommes attaqués à la fonction calcScore_FPTP.

La fonction calcScoreFPTP a pour but de renvoyer un vecteur scores contenant le résultat obtenu. Il parcourt donc tout le fichier pour ensuite parcourir chaque vote et récupérer l'indice de la position du 1. En récupérant cet indice, il va ajouter 1 au vecteur scores à la même position que le vote.

Pour comprendre plus facilement comment le système de vote marche, il est plus simple de faire un algorigramme ; mais celui-ci ressemblant à Gabriel, il est préférable de proposer un algorigramme montrant le procédé de la fonction calcScore_FPTP.



Pour tester notre programme, nous avons créé des fichiers nommés “test”. Ils sont au nombre de 5 pour mon programme, un avec une égalité, un autre avec seulement des 0, un avec 100 candidats et deux autres aléatoires. Ils permettent de vérifier la véracité de notre programme.

Partie 3 : La méthode Borda - Alexis Delin

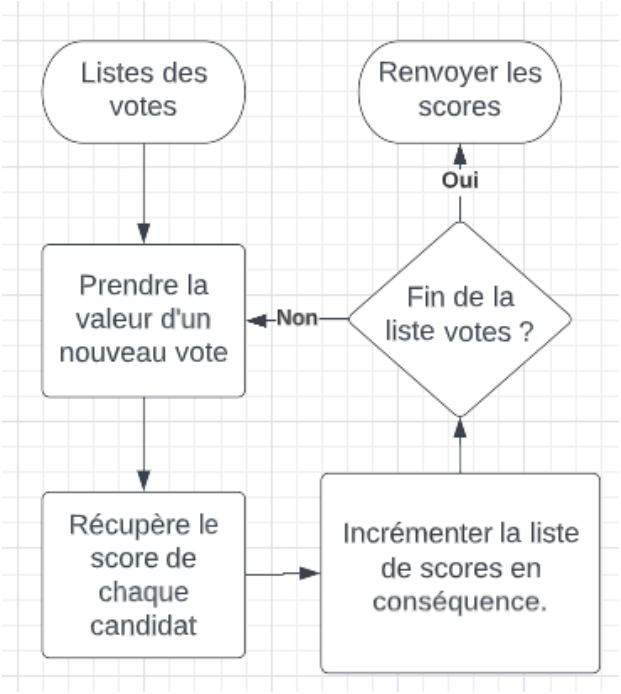
Bien que différent des système de vote précédent, la méthode non binaire de vote tel que celui de cette partie n'est pas compliquée à comprendre ou à mettre en place. La méthode Borda est une méthode de classement visant à numéroter les candidats de 1 à "nombre de candidats".
Des points seront attribués à chaque électeur en fonction de sa position.
Le candidat noté d'un 1 représente le candidat que le voteur veut élire, le nombre de point qu'il reçoit est de :
nombreDeCandidats - numéroVoté
Et par conséquent, plus le nombre est élevé, moins le nombre à de valeur.

De plus la méthode Borda est un système de vote utilisé pendant les élections en Slovénie, ainsi que sur les îles de Kiribati et Nauru situé dans la sous région de Micronesia en Océanie

Avec ces informations il nous a fallu créer une fonction capable de reproduire la fonction montrée ci-dessus “calcScore_Borda”, celle-ci fut notre fonction principale. Dans cette fonction, chaque vote est pris depuis une matrice représentée par la liste des scores puis additionné à chaque électeur.

L’utilisation de différents fichiers Oracle nous a permis de vérifier la véracité de notre programme dans plusieurs cas dont une possible égalité.Il sont au nombre de 3, deux d’entre eux représentent 2 simulation de vote dans lequel il y a un gagnant, le 3ème test visait à vérifier une possible égalité qui s’est avéré fonctionnel.

La méthode de Borda n’ayant pas de second tour, l’algorigramme est presque le même que celui du vote par approbation. Seule l’action “compte” et le vote aléatoire diffèrent. Par conséquent, pour visualiser le schéma, veuillez vous référer au schéma du vote par approbation.



Partie 4 : Vote Alternatif - Mathieu Leroux

Un second nom que l'on donne au vote alternatif est le vote à second tour instantanée. Et en effet, cette autre appellation convient mieux à la compréhension du fonctionnement de celui-ci. Il s'agit d'un cas similaire à la méthode Borda dans la manière de voter, puisque chaque individu constitue un classement des candidats dans son vote. Cependant, une majeure différence apparaît ensuite, puisque seul le candidat ayant reçu le meilleur classement du bulletin (donc la place n°1) obtient un point, tandis que les autres ne sont pas comptabilisés. Après ce premier comptage, s'il est cas d'une majorité absolue, le candidat en question remporte l'élection. Sinon, le ou les candidats avec le score le plus bas sont éliminés. Un nouveau tour se déroule alors à la manière du précédent, à l'exception que les candidats éliminés sont ôtés des bulletins, ayant pour conséquence de changer les résultats. L'opération est répétée jusqu'à l'obtention d'une majorité absolue. L'intérêt du classement intervient ainsi seulement à partir du deuxième tour. Ce vote est probablement le plus complexe dans sa structure parmi les 4 qui ont été choisis pour notre projet.

Ce système électoral est utilisé dans plusieurs pays d'Océanie pour désigner les membres du parlement, notamment l'Australie. Il est aussi utilisé en Irlande, en Inde et au Sri Lanka pour l'élection présidentielle. Il est fréquemment utilisé par de très hautes instances pour départager plusieurs options.

Pour programmer, ce système de vote, j'ai mis au point une fonction principale calc_IROV() (IROV, abréviation de "instant run-off voting" ; c'est la traduction anglaise du système en question). Cette fonction prend en paramètre une liste de vote, présentée sous la forme d'un vecteur de vecteur d'entiers. Ensuite à l'aide d'une boucle for, la liste est parcourue vote-à-vote. Est cherchée la valeur la plus basse qui correspond à un candidat encore en liste pour désigner qui obtient le point du bulletin dépouillé. Après le décompte des votes, on vérifie s'il y a cas de majorité absolue. Si un ou deux candidats l'atteignent (dans le cas de deux 50% parfaits), le vainqueur est désigné. Sinon, le ou les candidats

ayant le score le plus bas voient leur index ajoutés à la liste des candidats éliminés. Pour m'aider à réaliser cet algorithme, j'ai créé une fonction annexe findEliminatedCandidats qui cherche et ajoute le ou les candidats à éliminer.

Partie 5 : Fonctions annexes

D'autres fonctions interviennent dans le déroulement du programme et qui n'entrent pas dans les parties précédentes. Ainsi, on discerne trois types de fonctions annexes : les fonctions générales, utilisées notamment dans le main, les fonctions utilisées dans le cadre des tests, et les fonctions non utilisées ou abandonnées.

1 : Fonctions générales

Elles permettent d'exploiter les fichier d'entrées et les fichiers Oracle pour faire fonctionner nos différents systèmes de votes, ainsi que pour effectuer les tests de nos fichiers et de nos fonctions.

- getListVoteSys : Récupère le système de vote qui sera utilisé pour traiter l'entrée et simuler une élection.
- getListCandidats : Récupère la liste des candidats.
- getListVotes : Récupère la liste de tous les votes.
- separateWords : Rend exploitable les données extraites par les fonctions ci-dessus, en repérant les caractère d'espacement des valeurs (des tirets) et ainsi, séparer et ranger les valeurs comme nécessaire.

Ces cinq fonctions permettent de lire les fichiers d'entrée de manière rigoureuse afin de créer les variables que nos programmes traitent.

- winning : Trouve et affiche le vainqueur grâce à la liste des scores. Si plusieurs candidats ont le même résultat, elle appelle la fonction draw.
- draw : Fonction récursive qui affiche les vainqueurs en cas d'égalité.

Ces fonctions sont utilisées pour déterminer ce que va renvoyer l'algorithme.

- myFind : Notre propre version du find, qui nous a été nécessaire car les fonctions find intégrées à C++ renvoient une adresse mémoire et non un index. Elle facilite ainsi la conception du programme.

2 : Fonctions de tests

Ce sont des fonctions qui nous étaient utiles pour déboguer nos fonctions les plus importantes, qui permettent notamment d'afficher le contenu de différents types de vecteurs.

- printVectorOfVector : Affiche tous les éléments d'une liste de liste d'entiers, elle a été utilisée pour voir les valeurs de la liste votes
- printVectorOfUnsigned : Affiche tous les éléments d'une list d'entier, elle a été utilisée pour voir la liste des scores et les valeurs des listes contenu dans votes
- printVectorOfString : permet d'afficher tous les éléments d'une liste de chaîne de caractères, elle a été utilisée pour la liste des candidats et la liste des systèmes de vote
- isGlobalEntryValid : permet de s'assurer qu' aucune des listes d'entrée sont vide et vérifie que le ou les systèmes de votes entrées sont valides, cette fonction permet

d'en certains cas de nous prévenir directement que le problème vient de l'entrée et permet donc de gagner du temps quand on rencontre un bug.

3 : Fonctions non utilisées ou abandonnées

On y rassemble des fonctions comme celles permettant de vérifier si une entrée est valide, puisque plus tard, une consigne a été ajoutée telle qu'il est considéré que les entrées sont toujours valides, ou bien d'autres fonctions qui étaient utilisées lors de la création de nos programmes, par exemple, la fonction qui permet la génération aléatoire d'une liste de votes pour le vote par approbation.

- isEntryValid_ranked : Vérifie qu'un vote à bien la même taille que la liste des candidats et vérifie que la liste contient les entier qui n'excèdent pas l'index des candidats. Elle est utilisée dans le cas du vote alternatif et de la méthode Borda.
- isEntryValid_unique : Vérifie qu'un vote à bien la même taille que la liste des candidats et vérifie que la liste contient que des 0 et un seul 1. Elle est utilisée dans le cas du scrutin uninominal majoritaire à un tour.
- isEntryValid_binary : Vérifie qu'un vote à bien la même taille que la liste des candidats et vérifie que la liste contient que des 0 et des 1. Elle est utilisée dans le cas du vote par approbation.
- sortValidVoteEntry : Prend un paramètre la liste des votes et supprime les entrées non-valides en fonction du système de vote qui va être utilisé. Elle utilise ainsi la fonction isEntryValid .Cette fonction s'est avérée incompatible avec la capacité d'utiliser deux systèmes de votes pour un même jeu d'entrées. En effet, la faire fonctionner ainsi aurait nuit à l'optimisation du programme.
- les fonctions votingbyApproval, electionsByApproval et completeSimulationApprovalVote sont utilisées pour réaliser une simulation aléatoire du vote par approbation, malheureusement elles se sont révélées inutile car elles ne correspondaient pas aux consignes mais nous avons décidé de les laissées dans le code source pour laisser une trace de notre travail.

Conclusion

Voici alors le résumé de toutes nos tentatives et de notre travail final au sujet de la simulation de 4 systèmes de votes. Tout au long du développement, des coïncidences et des idées nous ont permis d'optimiser notre travail, et ce, grâce à une communication efficace ; par exemple, les fonctions winning et draw, dont à posteriori nous avons remarqué qu'elles étaient utilisables pour tous les systèmes. Lorsque des problèmes survenaient, les regards extérieurs de nos camarades permettaient de les résoudre rapidement et avec minutie, permettant la plupart du temps d'aboutir sur une amélioration significative des fonctions sujettes à nos soucis. La conceptualisation des programmes et l'écriture de ceux-ci étaient sans doute ce qui nous a le plus amusé, tandis que la rédaction des fichiers tests pouvait parfois se montrer un peu plus éprouvante puisque cela mobilise nettement moins de capacités cérébrales, et un peu plus de patience. Cela nous a néanmoins permis d'acquérir de l'expérience, et de nous améliorer dans les compétences que nous avons mobilisé.

Ainsi, nous pourrions résumer notre expérience lors de la production de ce devoir par quelques mots : une communication fluide, et un travail d'équipe efficace.