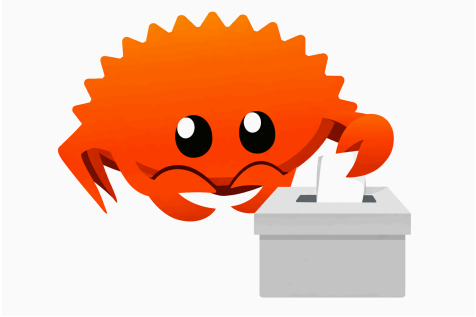
Rust : projet URNE: Universal Rust Nomination Environment



https://rust-lang.org/

1. Introduction

Le but du projet est d'écrire un programme en rust qui applique, sur un ensemble de votes, un ensemble de méthodes de votes.

Un **scrutin** est composée d'un ensemble non vide de **candidats** et d'un ensemble de **bulletins de vote**. Un bulletin de vote est une suite ordonnée strictement de préférences sur tout ou partie des candidats du scrutin.

Par exemple, le fichier suivant présente un scrutin valide avec 5 candidats et 6 bulletins :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Le but du projet est d'appliquer différentes méthodes de vote (présentées plus bas) sur un scrutin passé en paramètre au programme. Par exemple, sur le bulletin ci-dessus, le programme devra afficher le résultat suivant :

```
Plurality: A
TRS: A
IRV: A
Borda: A
Bucklin: D
Baldwin: A
Copeland: B
Copeland+Borda: D
Schulze: D
Smith+IRV: B
```

2. Méthodes de Vote

Pour chacune des méthodes suivantes, en cas d'égalité, on départagera les vainqueurs selon l'ordre lexicographique croissant (en d'autres termes, on sélectionnera toujours le candidat dont le nom est le plus proche du début de l'alphabet).

2.1. Vote à un tour (*Plurality*)

Principe : Le candidat avec le plus de voix en première position l'emporte.

Algorithme :
1. Compter le nombre de fois où chaque candidat est en première position.
2. Sélectionner le candidat avec le score le plus élevé.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Ici, les scores sont :

Candidat	Score
A	**2**
B	1
C	2
D	1
E	0
suffrages exprimés	6
majorité absolue	4

Ici le gagnant est A, arrivé premier *ex aequo* et premier dans l'ordre lexicographique.

2.2. Vote à deux tours (*Two-round system* ou *TRS*)

Principe : Si aucun candidat n'obtient la majorité absolue au premier tour, les deux candidats arrivés en tête s'affrontent au second tour.

Algorithme :
1. Premier tour : compter les voix en première position.
2. Si un candidat a la majorité absolue, il est élu.
3. Sinon, organiser un second tour entre les deux candidats arrivés en tête.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Ici, les scores sont :

Candidat	Tour 1	Tour 2
A	**2**	**3**
B	1	—

Candidat	Tour 1	Tour 2
C	**2**	2
D	1	—
E	0	—
suffrages exprimés	6	5
majorité absolue	4	3

A et C accèdent au second tour car ils sont premiers, sans qu'aucun d'entre eux n'aie la majorité absolue à ce tour. On reprend donc tous les bulletins en ne gardant que A et C à chaque fois :

```
A>B>C       -> A>C
A>D>E>B     -> A
B>A>D>E     -> A
C>D>B>A>E   -> C>A
C>D>B       -> C
D           -> (abstention)
```

A est le gagnant car il est plus souvent préféré à C que l'inverse.

2.3. IRV (Instant Runoff Voting)

Principe : Élimination successive des candidats avec le moins de voix en première position, jusqu'à ce qu'un candidat obtienne la majorité absolue.

Algorithme :
1. Compter les voix en première position.
2. Éliminer le candidat avec le moins de voix.
3. Réattribuer les voix des votants dont le candidat préféré a été éliminé.
4. Répéter jusqu'à ce qu'un candidat obtienne la majorité absolue.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Ici, les scores successifs sont :

Candidat	Score			
A	**2**	**2**	**2**	**3**
B	**1**	**1**	1	—
C	**2**	**2**	**2**	2
D	**1**	1	—	—
E	0	—	—	—
suffrages exprimés	6	6	5	5
majorité absolue	4	4	3	3

Au premier tour, personne n'ayant la majorité absolue, E est éliminé. Les bulletins deviennent donc :

```
A>B>C      -> A>B>C
A>D>E>B    -> A>D>B
B>A>D>E    -> B>A>D
C>D>B>A>E -> C>D>B>A
C>D>B      -> C>D>B
D          -> D
```

Au deuxième tour, personne n'a la majorité absolue, c'est D qui est éliminé :

```
A>B>C    -> A>B>C
A>D>B    -> A>B
B>A>D    -> B>A
C>D>B>A -> C>B>A
C>D>B    -> C>B
D        -> (abstention)
```

Au troisième tour, toujours pas de majorité absolue, B est éliminé à son tour.

```
A>B>C         -> A>C
A>B           -> A
B>A           -> A
C>B>A         -> C
C>B           -> C
(abstention) -> (abstention)
```

C'est au quatrième tour (un duel) que A remporte la majorité absolue et donc le scrutin.

2.4. Borda

Principe : Chaque candidat reçoit un nombre de points égal au nombre de candidats qu'il surclasse dans chaque classement.

Algorithme :

1. Pour chaque bulletin classant m candidats, attribuer m points au premier choix, $m-1$ au second, etc. Les candidats non-classés ne reçoivent pas de points.
2. Sommer les points pour chaque candidat.
3. Sélectionner le candidat avec le score total le plus élevé.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Les scores seront :

Candidat	Score
A	**12**
B	11
C	9
D	12
E	4

Par exemple, pour le premier bulletin `A>B>C`, comme il y a 3 bulletins, on compte 3 points pour A, 2 pour B, 1 pour C, et aucun pour D et E qui ne sont pas classés. Pour le suivant, `A>D>E>B`, comme il y a 4 bulletins, on compte 4 points supplémentaires pour A, 3 pour D, 2 pour E et 1 pour B.

Au final, A et D ont le meilleur score (12 points), la règle de l'ordre lexicographique fait que c'est A qui remporte le scrutin.

2.5. Bucklin

Principe : Itérer sur les rangs de préférence jusqu'à ce qu'un candidat obtienne la majorité absolue.

Algorithme :

1. Pour chaque rang (k) : a. Compter le nombre de fois où chaque candidat est classé au rang (k) ou mieux. b. Si un candidat obtient la majorité absolue, il est élu.
2. Répéter jusqu'à ce qu'un candidat soit élu ou que tous les bulletins soient épuisés.

Il peut arriver que plusieurs candidats dépassent la majorité absolue sur un tour donné. Dans ce cas, celui qui a le score le plus élevé est élu.

Il peut arriver que les bulletins soient épuisés sans qu'un candidat n'atteigne la majorité absolue. Dans ce cas encore, celui qui a le score le plus élevé est élu.

Exemple 1 :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Ici, les scores sont :

Candidat	Tour 1	Tour 2
A	2	3
B	1	2
C	2	2
D	1	**4**
E	0	0
suffrages exprimés	6	—
majorité absolue	4	4

Au premier tour, on ne compte que les préférences du premier rang, comme dans un vote à un seul tour. Par exemple, le bulletin `A>B>C` apporte une voix à A.

Comme personne n'a la majorité absolue, au deuxième tour, on compte les préférences des deux premiers rangs. Ainsi, le bulletin `A>B>C` apporte une voix à A et une voix à B. La majorité à atteindre ne change pas par rapport au premier tour : il faut quatre voix. Cette fois, D obtient le score nécessaire et il est le seul, il est donc élu.

Exemple 2

```
A;B;C;D
A>B>C>D
```

```
B>A>C
C>B>A
```

Ici, les scores sont :

Candidat	Tour 1	Tour 2
A	1	2
B	1	**3**
C	1	1
D	0	0
suffrages exprimés	3	—
majorité absolue	2	2

Ici, A et B atteignent tous deux la majorité absolue au second tour. Mais B a plus de voix que A, il est donc vainqueur.

Exemple 3

```
A;B;C;D
A>B
B>C
D>C>A
D
```

Cette fois, les scores sont :

Candidat	Tour 1	Tour 2	Tour 3
A	1	1	**2**
B	1	2	2
C	0	2	2
D	2	2	2
suffrages exprimés	4	—	–
majorité absolue	3	3	3

Ici, tous les bulletins sont épuisés avant que l'on n'atteigne la majorité absolue. C'est le candidat qui a le plus de voix qui est élu. Comme ils sont tous *ex aequo*, c'est A qui gagne.

2.6. Baldwin

Principe : Version itérative de la méthode Borda, où le candidat avec le score Borda le plus faible est éliminé à chaque itération. C'est un hybride entre Borda et IRV.

Algorithme :
1. Calculer les scores de Borda pour chaque candidat.
2. Éliminer le candidat avec le score Borda le plus faible.
3. Répéter jusqu'à ce qu'il ne reste qu'un candidat.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
```

```
C>D>B
D
```

Les scores seront :

Candidat	Tour 1	Tour 2	Tour 3	Tour 4
A	**12**	**9**	**8**	**6**
B	**11**	**9**	8	–
C	**9**	8	–	–
D	**12**	**9**	**9**	6
E	4	–	–	

Ici, au premier tour, pour le bulletin `A>D>E>B`, A marque 4, D marque 3, E marque 2 et B marque 1. Comme E est éliminé à la fin de ce tour, le bulletin devient `A>D>B`, donc A marque 3, D marque 2 et B marque 1.

2.7. Méthodes de Condorcet

Les méthodes suivantes sont toutes basées sur la méthode de Condorcet, qui va être présentée ici.

Partant du principe qu'il est facile de déterminer le gagnant d'un vote dans une élection à deux options (deux candidats, oui/non, pour/contre, coupable/innocent, etc.), puisqu'il suffit de demander à chaque électeur un choix et de déterminer lequel des deux choix a le plus de scrutins (et aura forcément la majorité absolue), il propose la procédure suivante :

1. Faire s'affronter tous les candidats deux à deux,
2. Si un candidat remporte tous ses duels, alors il sera gagnant.

Exemple 1

Soit le scrutin suivant :

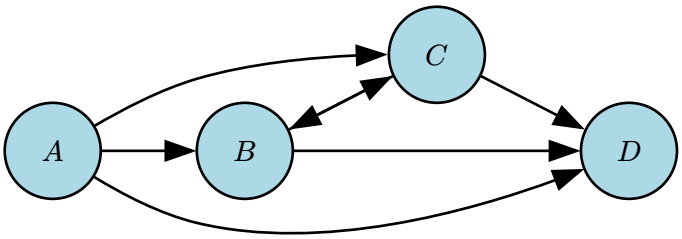
```
A;B;C;D
A>B>C
B>A>C>D
C>A>D
A
```

Pour une élection de n candidats on aura $\frac{n*n-1}{2}$ duels à traiter. On aura donc $\frac{4*3}{2} = 6$ duels : A vs B, A vs C, A vs D, B vs C, B vs D et C vs D.

Voici le tableau des résultat :

Candidat 1	Candidat 2	Vainqueur
A	B	A
A	C	A
A	D	A
B	C	–
B	D	B
C	D	C

On représente habituellement ces duels sous forme d'un graphe orienté du vainqueur vers le perdant. Les égalités sont représentées par des doubles flèches.



Comme on le voit, il y a un noeud qui domine tous les autres (de manière moins formelle : aucune flèche ne pointe vers ce noeud) : A. C'est celui qui gagne tous ses duels. C'est le gagnant de Condorcet.

Exemple 2

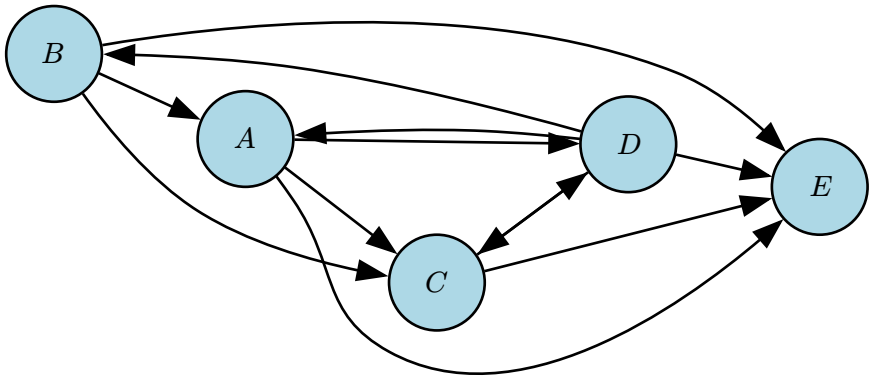
Dans l'exemple suivant, on aura 5 candidats, donc $\frac{5*4}{2} = 10$ duels.

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

Voici les résultats :

Candidat 1	Candidat 2	Vainqueur
A	B	B
A	C	A
A	D	—
A	E	A
B	C	B
B	D	D
B	E	B
C	D	—
C	E	C
D	E	D

Et voici le graphe associé :



Problème de la méthode de Condorcet : parfois, il n'y a pas de vainqueur. Ici par exemple, personne ne remporte tous ses duels. Qui doit l'emporter ? B qui a remporté 3 duel mais a échoué une fois ? Ou bien D qui n'a perdu aucun duel mais s'est retrouvé deux fois à égalité ?

C'est à cause de l'éventuelle présence de ces cycles que, lorsque l'on utilise la méthode de Condorcet, on doit utiliser une procédure adaptée qui permet de départager les candidats en l'absence d'un gagnant de Condorcet.

2.8. Copeland

Principe : Comparer chaque paire de candidats et attribuer deux points pour chaque victoire, un point pour chaque nul, comme dans un tournoi sportif.

Algorithme :

1. Construire un graphe de préférences où un arc de A vers B signifie que A est préféré à B par une majorité de votants,
2. Pour chaque candidat, compter le nombre de victoires (arcs sortants, 2 points) et le nombre de matchs nuls (1 point),
3. Sélectionner le candidat avec le score le plus élevé.

Exemple 1

```
A;B;C;D
A>B>C
B>A>C>D
C>A>D
A
```

Voici les résultats :

Candidat 1	Candidat 2	Vainqueur
A	B	A
A	C	A
A	D	A
B	C	–
B	D	B
C	D	C

Voici les scores :

Candidat	Victoires	Nuls	Score
A	3	0	**6**
B	1	1	3
C	1	1	3
D	0	0	0

A, qui est le vainqueur de Condorcet, a aussi le score le plus élevé et remporte l'élection.

Exemple 2 :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
```

```
C>D>B>A>E
C>D>B
D
```

Voici les résultats :

Candidat 1	Candidat 2	Vainqueur
A	B	B
A	C	A
A	D	—
A	E	A
B	C	B
B	D	D
B	E	B
C	D	—
C	E	C
D	E	D

Voici les scores :

Candidat	Victoires	Nuls	Score
A	2	1	5
B	3	0	**6**
C	1	1	3
D	2	2	**6**
E	0	0	0

Ici B et D ont le score le plus élevé, c'est donc B qui remporte le scrutin.

Cette méthode garantit de sélectionner le gagnant de Condorcet s'il y en a un (il aura alors forcément le score le plus élevé, sans égalité), et de sélectionner un ou plusieurs vainqueurs en l'absence d'un gagnant de Condorcet.

Le problème de la méthode de Copeland est qu'elle donne facilement lieu à des égalités, même dans des scrutins avec beaucoup de bulletins (ce qui est paradoxal étant donné son objectif initial). Pour les casser, on utilise souvent une seconde méthode en guise de **tie break**.

2.9. Copeland+Borda

Principe : En cas d'égalité selon la méthode Copeland, utiliser le score Borda pour départager.

Algorithme :
1. Appliquer la méthode Copeland.
2. Si égalité, calculer les scores de Borda *pour tout le monde*.
3. Sélectionner, parmi les premiers *ex aequo* selon Copeland, le candidat avec le score Borda le plus élevé.

Exemple :

Dans l'exemple ci-dessus, on avait les scores de Copeland suivants :

Candidat	Victoires	Nuls	Score
A	2	1	5
B	3	0	**6**
C	1	1	3
D	2	2	**6**
E	0	0	0

Comme B et D sont *ex aequo*, on va regarder celui qui a le score de Borda le plus élevé. Le score de Borda de ce scrutin, pour rappel, était :

Candidat	Score
(A)	*(12)*
B	11
(C)	*(9)*
D	**12**
(E)	*(4)*

On calcule le score de l'ensemble des candidats, pas uniquement celui des vainqueurs.

C'est D qui l'emporte, avec un score de 12. A aussi avait un score de 12, mais comme il n'avait pas gagné selon la méthode de Copeland, il n'était plus en lice.

La méthode de Copeland (et par extention celle de Copeland+Borda) n'est jamais utilisée en pratique. Elle a néanmoins été adoptée pour certaines compétitions sportives. Par exemple la phase de poules de la coupe du monde de football utilise une adaptation de Copeland (victoire 3 points, nul 1 point, défaite 0) et départage les *ex aequo* en sélectionnant celui qui a obtenu le plus de buts sur toute la phase, ce qui est formellement équivalent à un Borda.

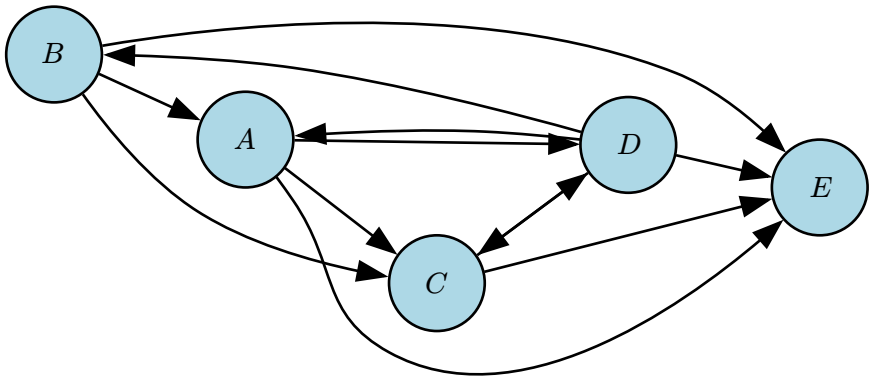
La méthode de Copeland est également utilisée indirectement, dans d'autres méthodes, notamment pour déterminer l'ensemble de Smith.

2.10. Ensemble de Smith

Les graphes des duels de Condorcet sont difficiles à lire en l'absence d'un gagnant. L'idée derrière la notion d'ensemble de Smith est d'extraire le sous-ensemble dans lequel se trouve le cycle de victoires mutuelles.

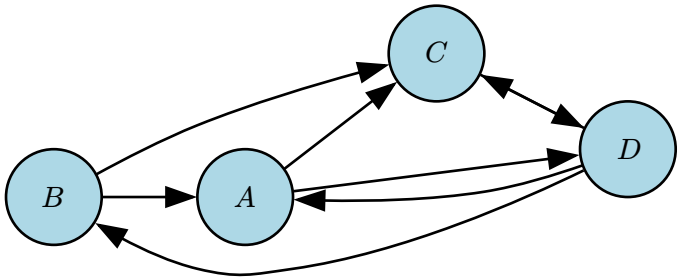
Formellement, l'ensemble de Smith est **le plus petit sous-ensemble de candidats tel qu'aucun candidat hors de l'ensemble ne bat un candidat de l'ensemble**.

Par exemple, dans le graphe suivant :



L'ensemble de Smith est $\{A, B, C, D\}$. En effet, le dernier noeud (E) ne pointe vers aucun des noeuds de l'ensemble. Par contre, il est impossible d'ôter le moindre noeud de $\{A, B, C, D\}$ car chacun de ces noeuds pointe sur au moins un autre noeud de l'ensemble.

Le graphe ressemble désormais à cela :



On peut désormais raisonnablement considérer que le vainqueur du scrutin doit se trouver dans cet ensemble. Reste à savoir comment le déterminer.

S'il y a un gagnant de Condorcet, alors ce sera le seul noeud de l'ensemble. Sinon, c'est la méthode que l'on va utiliser qui va départager les *ex aequo*.

2.10.1. Algorithme de construction de l'ensemble de Smith

Pour construire l'ensemble de Smith, on peut utiliser la procédure itérative suivante :

1. déterminer le ou les noeuds vainqueurs selon la méthode de Copeland et les intégrer à l'ensemble,
2. ajouter tous les noeuds qui pointent vers l'un des vainqueurs,
3. répéter jusqu'à stabilisation de l'ensemble.

Par exemple, ci-dessus, les vainqueurs selon la méthode de Copeland sont B et D. Notre ensemble de Smith sera donc un surensemble de $\{B, D\}$. Comme A et C pointent sur D, on les ajoute à l'ensemble, qui vaut maintenant $\{A, B, C, D\}$. Comme aucun autre noeud ne pointe sur ces 4 noeuds, l'algorithme se termine.

2.10.2. Propriétés de l'ensemble de Smith

S'il y a un gagnant de Condorcet, alors il sera le seul noeud de l'ensemble de Smith, et *vice versa*.

L'ensemble de Smith est forcément un surensemble (pas forcémet strict) de l'ensemble des premiers *ex aequo* déterminés selon la méthode de Copeland.

2.11. Schulze

Principe : Déterminer le chemin de force maximale entre chaque paire de candidats dans l'ensemble de Smith.

On va déterminer, pour chaque duel, de combien de voix le gagnant a battu le perdant.

Algorithme :
1. construire l'ensemble de Smith,
2. déterminer, pour chaque duel, l'avance du gagnant sur le perdant (nombre de voix du gagnant - nombre de voix du perdant)
3. éliminer le lien ayant le score le moins élevé,
4. si un noeud n'est désormais vaincu par personne, il est gagnant,
5. répéter les étapes 3 et 4 jusqu'à obtenir un gagnant.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

L'ensemble de Smith, on l'a vu, est $\{A, B, C, D\}$. Déterminons les résultats des duels et les avances de chaque gagnant.

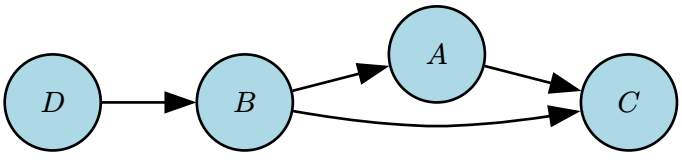
Prenons le duel A vs B. C'est B qui gagne, 3 voix à 2, soit un avantage de 1. Pour le duel B vs D, c'est D qui gagne, 4 vois à 2, soit un avantage de 2. On fait le calcul pour tous les duels :

Candidat 1	Candidat 2	Vainqueur	Avance
A	B	B	1
A	C	A	1
A	D	—	0
B	C	B	1
B	D	D	2
C	D	—	0

Quelle est l'avance la plus faible ? C'est 0 (A vs D et C vs D). Éliminons les lignes concernées :

Candidat 1	Candidat 2	Vainqueur	Avance
A	B	B	1
A	C	A	1
B	C	B	1
B	D	D	2

Traçons le graphe associé :



D n'est plus dominé par personne, il est donc déclaré vainqueur.

2.12. Smith+IRV

Principe : Éliminer successivement de l'ensemble de Smith les candidats ayant été le moins souvent préférés selon le principe de la méthode IRV, jusqu'à ce qu'il y ait un gagnant de Condorcet.

Algorithme :

1. construire l'ensemble de Smith,
2. s'il y a un gagnant de Condorcet, on s'arrête,
3. regarder lequel des membres de l'ensemble a été le moins souvent classé en première position,
4. le retirer de l'ensemble et retourner à l'étape 2.

Exemple :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

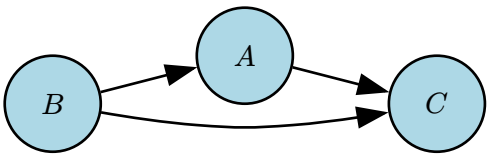
L'ensemble de Smith, on l'a vu, est $\{A, B, C, D\}$. On élimine donc tous les bulletins E du scrutin :

```
A;B;C;D
A>B>C
A>D>B
B>A>D
C>D>B>A
C>D>B
D
```

Puis on regarde qui a été le moins classé en première position :

Candidat	Score
A	**2**
B	**1**
C	**2**
D	1

Ici, B et D sont derniers *ex aequo*, donc pour des raisons lexicographiques D est éliminé. Le graphe des victoires est désormais le suivant :



B est désormais gagnant de Condorcet, et donc gagnant du scrutin.

3. Modalités du projet

Le projet est à réaliser en groupe de 4 (ou moins). Il doit être écrit intégralement en Rust.
L'exécutable en ligne de commande produit reçoit en parmètre le nom d'un fichier contenant un scrutin.

En cas d'erreur, on affiche un message d'erreur sur la sortie d'erreur avant de terminer le programme. Le format des messages d'erreur est libre, mais ils doivent être explicites.

Si le fichier est correct, on produira sur la sortie standard, pour chaque méthode, le nom de la méthode, suivi de deux points, suivis d'une espace, suivi du nom du vainqueur selon cette méthode, suivi d'un retour à la ligne.

Les méthodes auront les noms suivants. Ces noms et l'ordre dans lequel ils seront traités doivent être respectés :

- Plurality
- TRS
- IRV
- Borda
- Bucklin
- Baldwin
- Copeland
- Copeland+Borda
- Schulze
- Smith+IRV

Parfois dans la littérature sur le sujet, les noms des méthodes varient et les définitions de certaines méthodes exotiques sont peu ou pas formalisées. Attention donc aux informations que vous trouverez en ligne, qui peuvent s'avérer contradictoires avec le contenu de ce document. C'est le contenu de ce document qui fait foi. En cas de doute, n'hésitez pas à demander des précisions à l'enseignant.

Attention aussi aux résultats et explications fournis par les LLM. Sur ces sujets, ils commettent souvent des erreurs.

3.1. Format d'entrée

Un fichier représentant un scrutin est composé d'un en-tête et d'une suite de bulletins.

L'en-tête est composé d'une suite de noms séparés par des points-virgules et se termine par un retour à la ligne. Un nom est une chaîne de caractères non-vide et ne comprenant pas de caractère d'espacement, ni de retour à la ligne, ni de point-virgule, ni de signe "supérieur à".

Un bulletin est une suite éventuellement vide de noms séparés par des symboles "supérieur à" et terminé par un retour à la ligne. Le dernier bulletin du fichier peut, ou non, être suivi d'un retour à la ligne.

Un bulletin valide est un bulletin non vide ne comprenant que des noms de candidats mentionnés dans l'en-tête. En outre, un candidat ne peut y être mentionné qu'une fois au plus. Les bulletins non valides sont ignorés.

La syntaxe formelle est :

```
[scrutin] ::= [en-tête]\n[bulletin]*
[en-tête] ::= [string]\s*(;\s*[string])*
[bulletin] ::= \n|[string](\s*>\s*[string])*
```

Où `[string]` est toute suite de caractères ne comprenant ni retour chariot, ni espace, ni signe supérieur, ni point-virgule et `\s` est le cacactère d'espacement standard.

Votre programme devra pouvoir accepter au minimum des entrées de 30 candidats et de 50 millions de bulletins.

3.2. Format de sortie

3.2.1. Erreur

En cas d'erreur d'entrée, un message explicite doit être affiché sur la sortie standard. Le formet de ce message est libre, néanmoins le programme ne doit pas paniquer. Rien ne doit s'afficher sur la sortie standard.

3.2.2. Succès

Si le fichier est analysé sans erreur, alors le programme doit afficher le vainqueur du scrutin selon chacune des méthodes mentionnées plus haut, dans l'ordre et en respectant la graphie et la casse indiquées. Pour chaque méthode, la syntaxe à respecter est la suivante :

```
[nom]:\s[vainqueur]\n
```

Rien d'autre ne doit être produit sur la sortie standard. La sortie d'erreur, quant à elle, peut être utilisée librement pour afficher des messages informatifs à l'utilisateur. Ce n'est pas obligatoire.

3.3. Exécution du programme

Le nom du fichier scrutin sera passé en paramètre lors de l'appel. Ainsi, si le fichier `mon-scrutin.txt` contient les données suivantes :

```
A;B;C;D;E
A>B>C
A>D>E>B
B>A>D>E
C>D>B>A>E
C>D>B
D
```

On appellera le programme `mon-prog` et il s'exécutera comme suit :

```
$> mon-prog mon-fichier.txt
Plurality: A
TRS: A
IRV: A
Borda: A
Bucklin: D
Baldwin: A
Copeland: B
Copeland+Borda: D
Schulze: D
Smith+IRV: B
```

3.4. Programme de référence

Un programme de référence, baptisé `urne`, sera mis à votre disposition. Il répond aux spécifications du projet et vous permet de tester votre propre implémentation. La trace d'exécution de votre programme sera comparée à celle d'`urne`.

Le code source d'`urne` ne vous sera évidemment pas communiqué.

3.5. Critères d'évaluation

3.5.1. Fonctionnement du programme

Votre programme sera testé sur 10 entrées différentes. Cinq d'entre elles produiront une erreur et seront chacune notée sur un point, les cinq autres produiront un résultat correct et chacune sera notée sur 3 points, pour un total de 20 points.

3.5.1.1. Entrées erronées

Votre programme devra afficher un message d'erreur explicite sur la sortie d'erreur et se terminer proprement. Si le message d'erreur n'est pas explicite, vous perdez 0,5 point. Si d'autres informations sont produites, vous perdez 0,5 point. Si votre programme panique, ou s'il met un temps jugé déraisonnable à s'exécuter, vous aurez automatiquement 0 sur cette question.

3.5.1.2. Entrées correctes

Un `diff` sera effectué entre la sortie produite par votre programme et celle produite par `urne`. Chaque ligne erronnée, ou manquante, ou en trop, vous fera perdre 0,5 point. Si votre programme panique, vous aurez automatiquement 0 sur cette question. Si votre programme met un temps jugé déraisonnable à s'exécuter, il sera *killé* et seules les lignes produites seront prises en compte.

Un des fichiers testés sera volumineux (au maximum 50 millions de bulletins pour 30 candidats), les autres seront de taille plus restreinte.

3.5.2. Entretien oral

Un entretien oral de 15 minutes aura lieu lors de la dernière séance de TP. Au cours de cet entretien, des questions vous seront posées sur votre code, afin de vérifier que vous avez bien compris ce que vous avez écrit. Il est possible que des modifications de votre code vous soient demandées.

À la fin de cet entretien, une note sur 1 vous sera attribuée. Sa valeur pourra être de 1, de $0,8$, de $0,6$, de $0,4$, de $0,2$ ou de 0. Les deux notes que vous avez obtenues seront multipliées et le résultat arrondi au point supérieur pour déterminer votre note finale.