Table des matières

solution :

Précis sur les fonctions à employer. Tout le module List et Array (donc les Array.iter et cie). Mais pas de Hashtbl ni de Queue.

solution : Je noterai encore en majuscule les coefficients d'une matrice : $M_{i,j}$ et pas $m_{i,j}$ car je trouve cette notation plus cohérente.

1 Vote par préférence

Nous considérons une élection, à laquelle se présentent n candidats. Chaque électeur inscrit sur son bulletin l'ensemble des candidats (tous les candidats sont classés), par ordre de préférence. L'ensemble des bulletins est rassemblé dans une urne. Le nombre de votants est noté p, l'urne contient donc p bulletins.

Les trois types de données suivants sont utilisés pour représenter un vote :

- `type candidat = int ;;` chaque candidat est désigné par un numéro de 0 à $n-1$;
- `type bulletin = candidat list ;;` un bulletin de vote est une liste (ordonnée) de candidats ;
- `type urne = bulletin list ;;` une urne est un ensemble de bulletins de vote.

Par exemple, s'il y a trois candidats et qu'un électeur préfère le candidat 2, puis le candidat 0 et enfin considère que le candidat 1 est le moins souhaitable, son bulletin de vote sera `[2 ; 0 ; 1]`.

Une fois le vote effectué, on compare les résultats de deux candidats particuliers i et j en comptant le nombre de bulletins qui classent le candidat i avant le candidat j. On exprime le résultat de la comparaison entre les candidats i et j en calculant la différence entre le nombre de bulletins de vote qui placent i avant j et le nombre de ceux qui placent j avant i.

Premier exemple : On considère une élection avec trois candidat et trois votants : $n = 3$, $p = 3$. Les bulletins sont `[2 ; 0 ; 1]`, `[2 ; 1 ; 0]` et `[1 ; 2 ; 0]`. La comparaison entre le candidat numéro 0 et le candidat numéro 1 donne -1 car le candidat 0 est placé une fois avant le candidat 1 et deux fois après.

1.1 Graphe de préférences

1. Écrire une fonction `duel : candidat -> candidat -> urne -> int`, telle que `duel i j u` renvoie la comparaison entre le candidat i et le candidat j à partir des bulletins contenus dans l'urne u.

solution :

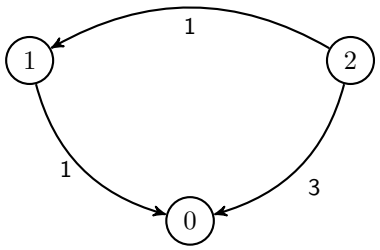
```
let rec preference i j b =
  (* Renvoie 1 si le bulletin b place i avant j et -1 sinon *)
  (* Précondition : ≠ij *)
  match b with
  |[] -> failwith "bulletin nul"
  |c::suite when c=i -> 1
  |c::suite when c=j -> -1
  |_::suite -> preference i j suite
;;

let rec duel i j u=
  (* Renvoie nb de votants qui préfèrent i - nb de votants qui préfèrent j, dans l'urne u. *)
  match u with
```

```
  |[] -> 0
  |b::suite -> preference i j b + duel i j suite
;;
(* cxté en O(nk) *)
```

On peut alors synthétiser le contenu d'une urne u en construisant le graphe de préférence des votants : ses sommets correspondent aux candidats et l'arc du sommet i vers le sommet j est pondéré par la comparaison entre le candidat i et le candidat j, c'est-à-dire la valeur de `duel i j u`.

La figure 1 donne le graphe de préférence obtenu à partir du vote de l'exemple 1 ainsi que sa matrice d'adjacence M. Afin d'alléger le schéma, seuls les arcs avec un poids strictement positif sont représentés.



$$M = \begin{pmatrix} 0 & -1 & -3 \\ 1 & 0 & -1 \\ 3 & 1 & 0 \end{pmatrix}$$

FIGURE 1 – Exemple de graphe de préférence

Deuxième exemple : On considère une élection avec trois candidats et 4 votants : n = 3, p = 4. À l'issue du vote, le contenu de l'urne est `[[0 ; 1 ; 2] ; [1 ; 2 ; 0] ; [0 ; 2 ; 1] ; [0 ; 2 ; 1]]`.

2. Tracer le graphe de préférence de l'urne de l'exemple 2 (en ne dessinant que les arcs ayant un poids strictement positif) et donner sa matrice d'adjacence.
3. Expliquer pourquoi la matrice d'adjacence d'un graphe de préférence est antisymétrique et pourquoi tous ses coefficients non-diagonaux ont la même parité.
   solution : Soit M une telle matrice. Si je note pour tout $(i,j) \in [\![0,n[\![^2$, $p_{i,j}$ le nombre de votants qui préfèrent i à j, alors, pour tout $(i,j) \in [\![0,n[\![^2$, si $i \neq j$, $m_{i,j} = p_{i,j} - p_{j,i} = -\left(p_{j,i} - p_{i,j}\right) = -m_{j,i}$. Et si $i = j$, $m_{i,j} = 0$ par convention (signalée en introduction).
   Ainsi M est antisymétrique.

   De plus, pour tout $(i,j) \in [\![0,n[\![^2$ tel que $i \neq j$, $p_{i,j} = n - p_{j,i}$ (tous les candidats sont classés, et pas d'égalité autorisée). Donc $m_{i,j} = n - 2p_{j,i} \equiv n[2]$. Ainsi tous les termes non diagonaux de la matrice ont la parité de n.

Étant donné une urne U, on note $\mathrm{Mat}(U)$ la matrice du graphe de préférence associée à cette urne.

4. Écrire une fonction `depouillement : int -> urne -> int array array` qui prend en paramètres le nombre de candidats n et une urne U et renvoie la matrice $\mathrm{Mat}(U)$.
   solution :

```
≠
let depouillement n u =
  let m = Array.make_matrix n n 0 in
  for i = 1 to n-1 do
    for j = 0 to i-1 do (* J'évite la diagonale. Et j'utilise l'antisymétrie.*)
      m.(i).(j) <- duel i j u;
      m.(j).(i) <- -duel i j u
    done
  done;
  m
;;
(* O(n^3×k) *)
```

1.2 Théorème de McGarvey

Le but de cette sous-partie est de démontrer le théorème de McGarvey : « Pour toute matrice antisymétrique à coefficients pairs M, il existe une urne U telle que $M = \mathrm{Mat}(U)$. »

Soient i, j et n trois entiers naturels tels que $i < n$, $j < n$ et $i \neq j$. On note $E_{i,j,n}$ la matrice carrée de taille n dont tous les coefficients sont nuls sauf les coefficients d'indices (i, j) et (j, i) qui valent respectivement 2 et -2.

5. Soit n un nombre de candidats et i et j deux entiers naturels strictement inférieurs à n. Montrer qu'il existe une urne $U_{i,j,n}$ contenant deux votes telle que $\text{Mat}(U_{i,j,n}) = E_{i,j,n}$.
   solution : On propose les deux bulletins b_1 et b_2 suivants :
   - $b_1 = (i, j, 1, 2, 3, ..., n)$
   - $b_2 = (n, n-1, ..., 2, 1, i, j)$

   où dans les pointillés on a pris soin de retirer i et j.
   Soit $u = \{b_1, b_2\}$ et M la matrice correspondante. On vérifie directement que :
   - $M_{i,j} = 2 = -M_{j,i}$.
   - Pour tout $(k, j) \in \left([\![0, n[\![\backslash \{i, j\}\right)^2$ tq $k \neq l$, $M_{i,j} = 0$.
   - Pour tout $k \in ([\![0, n[\![\backslash \{i, j\}$, $M_{i,k} = 0$.

   D'où $M = E_{i,j,n}$.
6. On considère deux urnes U_1 et U_2 . Exprimez $M_3 = \text{Mat}(U_1 \cup U_2)$ en fonction de $M_1 = \text{Mat}(U_1)$ et de $M_2 = \text{Mat}(U_2)$.
   solution : Avec les notations Caml, on voit que pour tout i, j, `duel i j (u1@u2)= duel i j u1 + duel i j u2`. On déduit ainsi que $M_3 = M_2 + M_1$.
7. Démontrer le théorème de McGarvey. *Indication :* Attention aux coefficients négatifs.

   solution : Soit M une matrice antisymétrique à coefficients entiers pairs. On vérifie directement que :

$$M = \sum_{0 \leqslant i < j < n} \frac{m_{i,j}}{2} E_{i,j,n}$$

   Posons pour tout $(i, j) \in [\![0, n[\![^2$ tel que $i < j$, $U_{i,j}$ une urne dont la matrice est $E_{i,j,n}$.
   Puis posons pour tout $(i, j) \in [\![0, n[\![^2$ tel que $i < j$, $V_{i,j}$ une urne obtenue en créant $\frac{m_{i,j}}{2}$ copies de $U_{i,j}$ si $m_{i,j} \geqslant 0$, ou $\frac{-m_{i,j}}{2}$ copies de $U_{j,i}$ sinon. Donc $\text{Mat}(V_{i,j}) = \frac{m_{i,j}}{2} E_{i,j,n}$.
   Enfin, posons $U = \bigcup_{0 \leqslant i < j < n} V_{i,j}$. On a alors $\text{Mat}(U) = M$.
8. Écrire une fonction `mcgarvey : int array array -> urne` prenant en paramètre une matrice antisymétrique M de coefficients tous pairs et renvoyant une urne U telle que $M = \text{Mat}(U)$.
   solution : Je suis la structure de la preuve avec des petites fonctions intermédiaires...

```
1  ≠
48 let rec intervalle n accu =
49   (* Renvoie …[0,1;;n-1] @ accu *)
50   if n=0 then accu
51   else intervalle (n-1) ((n-1)::accu)
52 ;;
53
54 let u_elem i j n=
55   (* Urne dont la matrice est E_{i,j,n}. *)
56   let p x = not (List.mem x [i;j]) in
57   let tout_sauf_ij =  List.filter p (intervalle n []) in
58   [
59     i::j::tout_sauf_ij ;
60     List.rev (j::i:: tout_sauf_ij)
61   ]
62 ;;
63
64 u_elem 2 4 9;;
65
66 let rec k_copies l k=
67   (* Renvoie la concaténation de  k fois de la liste l *)
68   if k=0 then []
69   else l @ k_copies l (k-1)
70 ;;
71
72 let mcgarvey m=
73   let n = Array.length m
```

```
  and u = ref [] in
  for j = 1 to n-1 do
    for i= 0 to j-1 do
      u :=
        (
          if m.(i).(j) >=0 then (* Attention au cas m.(i).(j) <0 *)
          (k_copies (u_elem i j n) (m.(i).(j)/2))
        else
          (k_copies (u_elem j i n) (-m.(i).(j)/2))
        )
        @ !u
    done
  done;
  !u
;;
```

9. Estimer la complexité de la fonction `mcgarvey` en fonction de n, la taille de la matrice (c'est-à-dire le nombre de candidats), et de q, le maximum des coefficients de la matrice.
solution :
   - Un appel à `intervalle` coûte $O(n)$.
   - Un appel à `u_elem` coûte $O(n)$ (un appel à `intervalle`, un `List.filter`, et un `List.rev`).
   - Un appel à `k_copies` pour concaténer k fois une liste de longueur n coûte $O(nk)$ (k concaténation avec une liste de longueur n à gauche).
   - Ainsi la complexité d'une itération de la boucle interne de `mcgarvey` coûte $O(n) + O(nk) + O(nk)$ (dans l'ordre l'appel à `u_elem`, à `k_copies`, et le `@`), ce qui fait $O(nk)$. Mais $k = O(q)$.
   Enfin la complexité finale est donc en $n^2O(nq)$, qui fait $O(n^3q)$.

2 Recherche du vainqueur

L'objectif de cette partie est de déterminer le vainqueur, ou les vainqueurs ex aequo, d'un vote par préférence.

2.1 Vainqueur de Condorcet

On appelle vainqueur de Condorcet tout sommet tel que, dans le graphe de préférence, les arcs sortant de ce sommet ont tous un poids positif ou nul. Ainsi, dans le premier exemple de la partie 1, le candidat 2 est un vainqueur de Condorcet.

10. Expliquer pourquoi un vainqueur de Condorcet peut être qualifié de « vainqueur » de l'élection.
solution : Un vainqueur de Condorcet est préféré (éventuellement ex-aequo) à tous les autres candidats par la majorité des électeurs. C'est donc sans conteste un vainqueur légitime.
11. Écrire une fonction `condorcet : int array array -> candidat list` prenant en paramètre la matrice d'adjacence d'un graphe de préférence et renvoyant la liste des vainqueurs de Condorcet.
solution :

```
≠…
let condorcet m =
  let res= ref [] in
  for i=0 to Array.length m -1 do
    if Array.for_all (fun x -> x>=0) m.(i) then
      res:= i:: !res
  done;
  !res
;;
```

12. En se plaçant dans le cas $n = 3$ et $p = 3$, construire une urne pour laquelle il n'existe pas de vainqueur de Condorcet. Tracer le graphe de préférence correspondant.
solution : Situation bien connue en sport lors des « poules », ou dans les cours de récréation pour le jeu « poule renard vipère » : 0 gagne contre 1, qui gagne contre 2, qui gagne contre 0.
```
let exemple_sans_vainqueur_Condorcet = [ [0 ;1 ;2] ; [1 ;2 ;0] ; [2 ;0 ;1]] ; ;
```

2.2 Graphe intermédiaire de Schultze

À la fin du XX^e siècle, Markus Schulze imagina une méthode permettant de déterminer un vainqueur à l'issue de n'importe quel vote par préférence.

On appelle poids d'un chemin du graphe de préférence, le minimum des poids des arcs constituant ce chemin. Ainsi, dans le premier exemple de la partie 1, le poids du chemin $2 \to 0 \to 1$ est -1.

Le graphe intermédiaire de Schulze est un graphe orienté pondéré complet dont les sommets sont les candidats et dont l'arc du sommet i vers le sommet j (distinct de i) est pondéré par le maximum des poids de tous les chemins allant de i à j dans le graphe de préférence. Sa matrice d'adjacence est notée I.

solution : Je note ρ la fonction qui a un chemin associe son poids et @ la concaténation des chemins.

J'aurais mis une question pour faire remarquer que $\rho(\gamma_1@\gamma_2) = \min\big(\rho(\gamma_1), \rho(\gamma_2)\big)$.

13. Pour i et j, candidats distincts, démontrer que $I[i,j]$ est aussi le maximum des poids des chemins sans boucle de i à j, c'est-à-dire des chemins $i = i_0 \to i_1 \to \cdots \to i_k = j$ avec i_0 , i_1 , ..., i_k deux à deux distincts.
solution : Soit γ un chemin de i à j. Soit γ' obtenu en retirant d'éventuelles boucles à γ. Ainsi γ' est aussi un chemin de i à j, et comme ses arêtes sont toutes des arêtes de γ, on a $\rho(\gamma') \geqslant \rho(\gamma)$ (minimum d'un ensemble inclus dans un autre).
Maintenant, si γ est un chemin de poids maximal de i à j (donc $I_{i,j} = \rho(\gamma)$), alors on a aussi $\rho(\gamma') \leqslant \rho(\gamma)$, par définition d'un maximum, donc $\rho(\gamma') = \rho(\gamma)$.
Donc $I_{i,j} = \rho(\gamma')$. C'est le maximum des poids des chemins de i à j, et a fortiori celui des poids des chemins sans boucle.

14. Démontrer que $\min\big(I[i,j], I[j,k]\big) \leqslant I[i,k]$ pour tout triplet (i,j,k) de sommets distincts.
solution : Soit $(i,j,k) \in [\![0,n[\![^3$ deux à deux distincts. Soient γ_1 un chemin de poids maximal de i à j, et γ_2 un chemin de poids maximal de j à k.
Comme $\gamma_1@\gamma_2$ est un chemin de i à j, on a $\rho(\gamma_1@\gamma_2) \leqslant I_{i,k}$.
Puis vu la définition de ρ, $\rho(\gamma_1@\gamma_2) = \min\big(\rho(\gamma_1), \rho(\gamma_2)\big)$.
Enfin, comme on a choisi des chemins de poids maximal, $\rho(\gamma_1) = I_{i,j}$ et $\rho(\gamma_2) = I_{j,k}$.

15. En adaptant l'algorithme de Floyd-Warshall, programmer une fonction `intermediaire : int array array -> int array array` prenant en paramètre la matrice d'adjacence d'un graphe de préférence et renvoyant la matrice d'adjacence du graphe intermédiaire de Schulze correspondant.
solution : Pour tout $(i,j,k) \in [\![0,n[\![^3$, je note $I^j_{i,k}$ le maximum des poids des chemins de i à j ne passant que par des sommets intermédiaires dans $[\![0,j[\![$.
Remarque : Les rôles des indices j et k sont inversés par rapport au cours sur l'algorithme de Floyd-Warshall.

Ainsi :

- Pour tout $(i,k) \in [\![0,n[\![^2$, $I^0_{i,k} = m_{i,k}$.
- Pour tout $(i,k) \in [\![0,n[\![^2$, $I^n_{i,k} = I_{i,k}$ (qu'on veut calculer).
- Pour tout $(i,k) \in [\![0,n[\![^2$, et tout $k \in [\![0,n-1[\![$,

$$I^{j+1}_{i,k} = \max\Big(I^j_{i,k}, \min(I^j_{i,j}, I^j_{j,k})\Big)$$

En effet, un chemin sans cycle de poids maximal de i vers j avec étapes dans $[\![0,j+1[\![$ peut être le même que celui avec étapes dans $[\![0,j[\![$, ou être composé d'un chemin de poids maximal de i vers j et d'un de j vers k, avec étapes dans $[\![0,j[\![$.
Le programme s'en déduit immédiatement.

```
1  ≠…
113 let intermediaire m =
114   let n = Array.length m in
115   let inter = Array.make_matrix n n 0
116   and tampon = Array.make_matrix n n 0 in
117
118   let recopie m1 m2 =
119     (* recopie m1 dans m2 *)
120     for i = 0 to n-1 do
121       for j =0 to n-1 do
122         m2.(i).(j) <- m1.(i).(j)
123       done
124     done
125   in
126   recopie m inter;
```

```
127
128   (* boucle principale *)
129   for j =0 to n-1 do
130     (* Invariant de boucle : ici, les inter.(i).(k) contiennent I^j_{i,k} (notations de mon
          ↪ corrigé) *)
131     for i=0 to n-1 do
132       for k=0 to n-1 do
133         tampon.(i).(k) <- max
134                           inter.(i).(k)
135                           (min inter.(i).(j) inter.(j).(k))
136       done
137     done;
138     recopie tampon inter
139             (* Maintenant, les inter.(i).(k) contiennent I_{i,k}^{j+1} *)
140   done;
141   inter
142 ;;
```

16. Estimez la complexité de la fonction `intermediaire` en fonction de n, le nombre de candidats.
solution : Complexité en $O(n^3)$.

2.3 Graphe de préférence de Schultze

Le graphe de préférence de Schulze est défini à partir du graphe intermédiaire de Schulze. Si I est la matrice d'adjacence du graphe intermédiaire de Schulze, alors la matrice d'adjacence S du graphe de préférence de Schulze est définie par $S[i,j] = I[i,j] - I[j,i]$ pour tous entiers naturels i et j strictement inférieurs à n.

17. Montrer que la matrice d'adjacence d'un graphe de préférence de Schulze est antisymétrique et que tous ses coefficients sont pairs.
solution : Soit $(i,j) \in [\![0,n[\![^2$. On a immédiatement $S_{i,j} = -S_{j,i}$ vu la définition.
En outre, $I_{i,j}$ étant défini uniquement à l'aide de min et de max à partir des coefficients de M, c'est finalement un des coefficients de M. Et $S_{i,j}$ est une soustraction de deux coefficients de M. Comme tous les coefficients de M ont la même parité, on déduit que $S_{i,j}$ est pair.

18. Écrire une fonction `graphe_schulze : int array array -> int array array` qui prend en paramètre un graphe intermédiaire de Schulze et qui renvoie le graphe de préférence de Schulze correspondant.
solution :

```
1 ≠…
147 let graphe_schultze inter =
148   let n = Array.length inter in
149   let shu = Array.make_matrix n n 0 in
150   for i=0 to n-1 do
151     for j=0 to n-1 do
152       shu.(i).(j) <- inter.(i).(j) - inter.(j).(i)
153     done
154   done;
155   shu
156 ;;
```

2.4 Vainqueur de Schultze

Un vainqueur de Schulze est un vainqueur de Condorcet dans le graphe de préférence de Schulze.

19. Écrire une fonction `schulze : int -> urne -> candidat list` qui prend en paramètres le nombre de candidats et un ensemble de bulletins de vote et renvoie la liste des vainqueurs de Schulze.
solution :

```
1 ≠…
161 let schultze n u =
162   let schu = graphe_schultze (intermediaire (depouillement n u)) in
163   condorcet schu
164 ;;
```

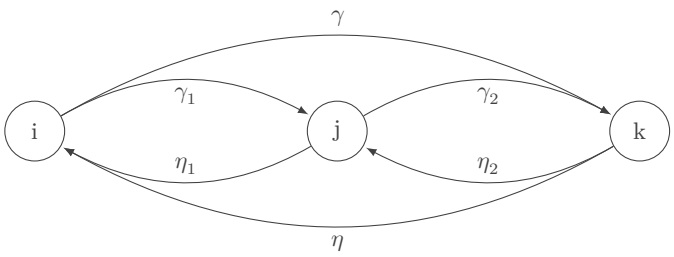
20. Estimer la complexité de la fonction schulze en fonction du nombre de candidats n et du nombre de votants p.
solution : Le dépouillement est en $O(n^3k)$. Le calcul de la matrice intermédiaire est en $O(n^3)$. Le calcul de la matrice de Schultze en $O(n^2)$. Enfin, `condorcet` est en $O(n^2)$. La complexité finale est la somme de tous ceci, elle vaut $O(n^3k)$ (au final, seul le dépouillement compte : les autres opérations sont négligeable devant celle-ci).

À partir d'un graphe de préférence de Schulze représenté par la matrice S, on définit la relation R_S entre les candidats comme suit : iR_Sj si et seulement $S[i,j]$ est strictement positif.

21. Montrer que la relation R_S est transitive, c'est-à-dire que pour tous candidats i, j et k, si iR_Sj et jR_Sk alors iR_Sk. *Indication :* Si I désigne la matrice d'adjacence du graphe intermédiaire, on pourra distinguer les cas $I[i,j] \leqslant I[j,k]$ et $I[i,j] > I[j,k]$.

solution : Soit $(i,j,k) \in [\![0,n[\![^3$ tel que iR_Sj et jR_Sk. Donc $S_{i,j} > 0$ et $S_{j,k} > 0$, ou encore $I_{i,j} > I_{j,i}$ et $I_{j,k} > I_{k,j}$.

L'idée est d'utiliser à deux reprises l'inégalité de 14. Cependant j'ai trouvé plus clair de refaire la démonstration de celle-ci.



Soient $\gamma_1, \gamma_2, \gamma, \eta, \eta_1, \eta_2$ des chemins de poids maximal entre les sommets indiqués sur le schéma. Notre hypothèse se réécrit :

$$\begin{cases} \rho(\gamma_1) > \rho(\eta_1) \\ \rho(\gamma_2) > \rho(\eta_2) \end{cases}.$$

Et le but est de prouver que $\rho(\gamma) > \rho(\eta)$.

Comme vu à la question 14, on a $\rho(\gamma) \geqslant \min(\rho(\gamma_1), \rho(\gamma_2))$.

- **Supposons** $\rho(\gamma_1) \leqslant \rho(\gamma_2)$ **:** On a alors

$$\begin{aligned} \rho(\gamma) &\geqslant \rho(\gamma_1) && \text{par hypothèse} \\ &> \rho(\eta_1) && \text{Car } \eta_1 \text{ est un chemin de poids maximal de } i \text{ vers } j \\ &\geqslant \rho(\gamma_2 @ \eta) \\ &\geqslant \min(\rho(\gamma_2), \rho(\eta)) \end{aligned}$$

  Maintenant, supposons que $\rho(\gamma_2) < \rho(\eta)$. On obtiendrait alors par le calcul qu'on vient de faire (enlever juste la première étape) $\rho(\gamma_1) > \rho(\gamma_2)$, ce qui est contradictoire puisque nous sommes dans le cas $\rho(\gamma_1) \leqslant \rho(\gamma_2)$.
  Ainsi, $\rho(\gamma_2) \geqslant \rho(\eta)$, donc $\min(\rho(\gamma_2), \rho(\eta)) = \rho(\eta)$, et l'inégalité que nous venons de prouver devient $\rho(\gamma) > \rho(\eta)$, ce qu'il fallait démontrer.
- **Supposons** $\rho(\gamma_1) > \rho(\gamma_2)$ **:** Cette fois nous obtenons $\rho(\gamma) \geqslant \rho(\gamma_2) > \rho(\eta_2) \geqslant \min(\rho(\gamma_1), \rho(\eta))$. Et l'hypothèse $\rho(\gamma_1) < \rho(h)$ est absurde, donc cette inégalité devient $\rho(\gamma) > \rho(\eta)$.

En conclusion, dans tous les cas, $\rho(\gamma) > \rho(\eta)$, donc $S_{i,k} > 0$ donc iR_Sk.

22. Montrer que quelle que soit l'urne non vide considérée, il existe toujours au moins un vainqueur de Schulze.
solution :

Essentiellement, comme $[\![0,n[\![$ est un ensemble fini, il admet un maximum pour R_S. Le seul détail est que R_S n'est pas forcément une relation d'ordre : nous ne savons pas si elle est antisymétrique.
Au passage, un « maximum pour R_S » ne sera donc pas forcément unique.

On prouve ceci par récurrence. Pour tout $k \in [\![0,n[\![$, notons $P(k)$: « Il existe $i \in [\![0,k]\!]$ tel que pour tout $j \in [\![0,k]\!] \setminus \{i\}$, iR_Sj» . Un tel i sera nommé un «vainqueur parmi $[\![0,k]\!]$».

- Pour $k = 0$, $i = 0$ fonctionne.
- Soit $k \in [\![0,n-1[\![$, supposons $P(k)$. Soit i le vainqueur parmi $[\![0,k]\!]$ donné par $P(k)$. Grâce à la transitivité de R_S, on voit que si $(k+1)R_Si$, alors $k+1$ est un vainqueur dans $[\![0,k+1]\!]$. Et dans le cas contraire on a $iR_S(k+1)$ (R_S est totale vu sa définition), alors i est un vainqueur pour $[\![0,k+1]\!]$.
  D'où $P(k+1)$.

Par récurrence, on obtient que $P(n-1)$ est vrai, d'où l'existence de i tel que pour tout $j \in [\![0,n[\!] \setminus \{i\}$, iR_Sj, qui est un vainqueur de Schulze.