Jugement majoritaire

INFORMATIQUE COMMUNE - Devoir n° 1 - Olivier Reynet

Consignes :

1. utiliser une copie différente pour chaque partie du sujet,
2. écrire son nom sur chaque copie,
3. écrire de manière lisible et intelligible,
4. préparer une réponse au brouillon avant de la reporter sur la feuille.

L'objectif de cet examen est de programmer les fonctions nécessaires à la création d'un logiciel permettant de calculer le gagnant d'une élection selon un scrutin à jugement majoritaire.

A Scrutin à jugement majoritaire

Le jugement majoritaire [1] est un mode de scrutin, c'est-à-dire une méthode de vote pour faire un choix. Lors d'un tel scrutin, **les électeurs attribuent une mention à chaque candidat** et peuvent attribuer la même mention à plusieurs candidats. Les échelles de mentions constituent un ordre et peuvent être verbales ou numériques :

- très bien > bien > assez bien > passable > insuffisant > à rejeter,
- A > B > C > D > E > F,
- 0 > 1 > 2 > 3 > 4 > 5.

Pour déterminer le vainqueur de l'élection (cf. partie II), on cherche à savoir quelle est la mention approuvée par une majorité d'électeurs pour un candidat donné. On appelle cette mention **mention majoritaire du candidat**. Le candidat qui obtient la meilleure mention majoritaire remporte l'élection. L'ensemble des profils des candidats donne une image des opinions des électeurs et permet d'évaluer les soutiens dont bénéficie le candidat élu (cf. figure 1).

Le tableau 1 et la figure 1 illustrent le fonctionnement de ce scrutin pour lequel il n'est pas rare d'obtenir des candidats *ex aequo*, c'est à dire ayant obtenu la même mention majoritaire. Dans ce cas, une procédure évaluant le rapport de force entre partisans et opposants des candidats est mise en place pour les départager (cf. partie III).

B Modélisation

On choisit de modéliser les votes d'une élection selon le scrutin du jugement majoritaire de la manière suivante :

1. Le jugement majoritaire a été inventé par Michel Balinski et Rida Laraki, chercheurs au CNRS.

Candidats	A	B	C	D	E	F
Hugo		x				
Lucille		x				
Balthazar				x		

TABLE 1 – Exemple de bulletin de vote pour le scrutin jugement majoritaire

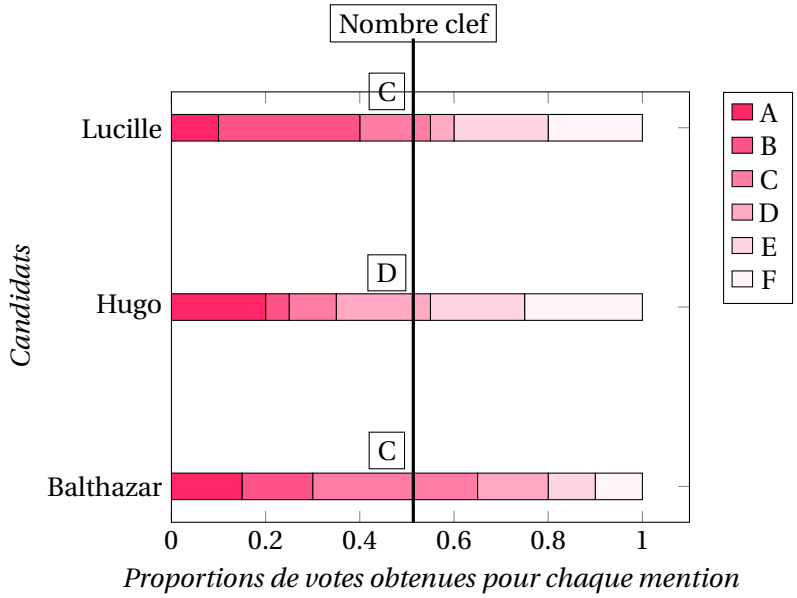

FIGURE 1 – Profils des candidats à l'issue du vote selon le scrutin jugement majoritaire. Les mentions majoritaires obtenues par les candidats sont : C pour Balthazar et Lucille, D pour Hugo. Le vainqueur de l'élection est Balthazar car Lucille a plus d'opposants que lui (45% contre 35%).

- les **candidats** sont représentés par une liste de chaînes de caractères. On choisit de repérer un candidat d'après sa position dans la liste. Par exemple, s'il occupe la deuxième position, l'indice qui le représente est le 1.
- **l'échelle des mentions** verbale est une liste triée selon l'ordre décroissant des mentions, de la plus favorable à la plus défavorable. On choisit d'utiliser l'échelle A > B > C > D > E > F, A étant la mention la plus favorable. On utilise également une échelle numérique des mentions 0 > 1 > 2 > 3 > 4 > 5, plus pratique pour la programmation. La mention 0 correspond à la mention A, la mention 1 à B…
- **le vote d'un électeur** est une liste comportant autant de valeurs que de candidats. Les valeurs de cette liste sont des mentions possibles issues de l'échelle des mentions. Par exemple, s'il y a trois candidats, un vote possible est E,B,D, soit E pour le candidat 0, B pour le 1 et D pour le 2. Sur l'échelle numérique des mentions on aurait 4,1,3.
- **l'ensemble des votes** est réuni dans une liste de listes : chaque élément de cette liste est le vote d'un électeur.
- **le profil d'un candidat** est une liste contenant autant d'éléments que de mentions possibles. Chaque élément est un entier qui indique le nombre d'électeurs qui a choisi cette mention pour ce candidat. Par exemple, si on utilise l'échelle A>B>C>D>E>F, le profil 9, 3, 2, 8, 0, 20 signifie que :
  - 9 électeurs ont jugé que ce candidat méritait la mention A,

- 3 la B,
- 2 la C,
- 8 la D,
- 0 la E,
- et 20 la F.

On remarque que la somme des mentions obtenues vaut le nombre d'électeurs (ici 42).

C Partie I : préliminaires

C1. Il y a `n` candidats à l'élection. On souhaite stocker les mentions majoritaires des candidats à l'élection. Créer une liste `M` comportant `n` éléments tous initialisés à `None`.

Solution :

```
n = 42
M = []
for _ in range(n):
    L.append(None)
# M = [None for _ in range(n)]
```

C2. Créer une liste de caractères `echelle` représentant l'échelle des mentions verbales A,B,C,D,E,F.

Solution :

```
echelle = ['A', 'B', 'C', 'D', 'E', 'F']
```

C3. Créer une fonction `echelle_num(echelle)` qui transforme une échelle de mentions verbales en une échelle numérique. Cette fonction renvoie une liste d'entiers qui représente les indices de chaque mention dans la liste `echelle`. Par exemple, pour la liste associée à l'échelle A>B>C>D>E>F, la fonction renvoie `[0,1,2,3,4,5]`.

Solution :

```
def echelle_num(echelle):
  num_echelle = []
  for i in range(len(echelle)):
    num_echelle.append(i)
  return num_echelle
  # return [i for i in range(len(echelle))]
```

C4. On utilise l'échelle numérique `[0,1,2,3,4,5]`. Créer une fonction `to_reject(mention)` dont le paramètre mention est de type `int` et qui renvoie un booléen valant vrai si la mention vaut `5` et faux sinon.

Solution :

```
def to_reject(mention):
  # if mention == 5:
  #    return True
  # else:
  #    return False
  return mention == 5
```

C5. Créer une fonction dont le prototype est `emax_imax(L)` et dont le paramètre `L` est une liste d'entiers. Cette fonction renvoie un **tuple** dont les deux valeurs sont le maximum et l'indice du maximum de la liste `L`. Si la liste `L` est vide, la fonction renvoie `None`. Par exemple, pour une liste `L = [1,6,3,9,5,4]`, l'instruction `emax_imax(L)` renvoie `(9,3)`.

Solution :

```
def emax_imax(L):
  if len(L) == 0:
    return None
  else:
    m = L[0]
    imax = 0
    for i in range(len(L)):
      if L[i] > m:
        m = L[i]
        imax = i
    return m, imax
```

C6. Créer une fonction dont le prototype est `emax_imax_parmi(L, selected)` et dont les paramètres `L` et `selected` sont des listes d'entiers. Cette fonction renvoie un tuple dont les deux valeurs sont le maximum et l'indice du maximum de la liste `L` **en ne considérant que les éléments de la liste dont les indices sont présents dans la liste `selected`.** Si `L` ou `selected` sont vides, la fonction renvoie `None`. Par exemple, pour une liste `L = [1,6,3,9,5,4]` et une autre `selected=[1,2,5]`, l'instruction `emax_imax_parmi(L)` renvoie `(6,1)`.

Solution :

```
def emax_imax_parmi(L, selected):
    if len(L) == 0 or len(selected) == 0:
        return None
    else:
        if selected[0] < len(L):
            m = L[selected[0]]
            im = selected[0]
            for i in selected[1:]:
                if i < len(L):
                    if L[i] > m:
                        im = i
                        m = L[i]
            return m, im
```

D Partie II : mécanique du jugement majoritaire

À l'issue du scrutin, on dispose de l'ensemble des votes des électeurs. On suppose maintenant, pour des raisons de simplicité de codage, qu'on manipule l'échelle de mention numérique de type 0,1,2,3,4,5 pour laquelle la mention 0 est la plus favorable. On n'utilisera les mentions verbales correspondantes A, B, C, D, E, F uniquement pour l'affichage des profils sur l'écran. On suppose enfin qu'on dispose de 165 électeurs et trois candidats : Hugo, Lucille, Balthazar.

D1. Créer la liste de chaînes de caractères `candidats` représentant les candidats.

Solution :

```
candidats = ["Hugo", "Lucille", "Balthazar"]
```

D2. On dispose maintenant de la liste `votes` qui contient tous les votes des électeurs : c'est donc une liste de listes d'entiers. Chaque élément de `votes` est une liste, le vote d'un électeur. Comment peut-on connaître le nombre d'électeurs ayant voté ?

Solution :

```
len(votes)
```

D3. Si `votes` est la liste qui contient tous les votes des électeurs, à quoi correspondent `votes[i]` et `votes[i][j]` ?

Solution : `votes[i]` est le vote du $(i+1)$ ième électeur, celui d'indice i.
`votes[i][j]` est la mention attribuée au candidat d'indice j par l'électeur d'indice i.

D4. Créer une fonction de prototype `profil_candidat(c, echelle_num, votes)` dont les paramètres sont :

- `c` de type `int` : l'indice du candidat dans la liste des candidats,
- `echelle_num` de type `list` d'entiers : l'échelle numérique des mentions,
- `votes` de type `list` de `list` d'entiers : l'ensemble des votes tous les électeurs.

Cette fonction renvoie un type `list` d'entiers qui représente le profil d'un candidat donné, c'est à dire le nombre de votes obtenus pour chaque mention. Par exemple : `[28, 26, 27, 29, 31, 24]`.

Solution :

```
def profil_candidat(c, echelle_num, votes):
  # profil = [0 for _ in range(len(echelle_num))] # init
  profil = []
```

```
    for m in echelle_num:  # init
      profil.append(0)
    for v in votes:
      profil[v[c]] += 1
    return profil
```

D5. On cherche à visualiser graphiquement le rapport de forces entre les candidats sur la console Python, c'est à dire à afficher quelque chose de similaire au graphique de la figure 1. Créer une fonction de prototype `profile_to_string(profil, echelle)` dont les paramètres sont de type :

- `list` d'entiers pour `profil`,
- `list` de caractères pour `echelle`.

Cette fonction **renvoie une chaîne de 80 caractères maximum** représentant, **en proportion**, le profil d'un candidat selon l'échelle de mention verbale donnée. **L'utilisation de la fonction `print` n'est pas autorisée**. Enfin, pour coder correctement cette fonction, il est nécessaire d'utiliser la fonction `floor` (partie entière) de la bibliothèque `math`. **Ne pas oublier de l'importer avant de l'utiliser.**

Par exemple, pour un profil `[29, 21, 21, 34, 27, 33]` et l'échelle `['A', 'B', 'C', 'D', 'E', 'F']`, la fonction renvoie la chaîne

`'AAAAAAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDDDDDDDDEEEEEEEEEEEEEEFFFFFFFFFFFFFFFF'`

Solution :

```
from math import floor
def profil_to_string(profil, echelle=["A", "B", "C", "D", "E", "F"]):
    size = 80
    n = 0
    for v in profil:
        n += v
    scaled_profil = [floor(p * size / n) for p in profil]
    s = ""
    for i in range(len(scaled_profil)):
        s += echelle[i] * scaled_profil[i]
    return s
```

D6. On cherche à ranger dans l'ordre croissant de voix le profil d'un candidat. On applique l'algorithme du tri par insertion. Sur le profil `[29, 21, 21, 34, 27, 33]`, écrire les différentes étapes de cet algorithme qui amène à `[21, 21, 27, 29, 33, 34]`. On écrira chaque étape sur une ligne en expliquant le passage d'une ligne à l'autre.

Solution : cf. cours pour les explications !

1. `[29, 21, 21, 34, 27, 33]`
2. `[21, 29, 21, 34, 27, 33]`
3. `[21, 21, 29, 34, 27, 33]`
4. `[21, 21, 29, 34, 27, 33]`
5. `[21, 21, 27, 29, 34, 33]`
6. `[21, 21, 27, 29, 33, 34]`

E Partie III : calculer le vainqueur

Pour déterminer le vainqueur d'une élection au scrutin jugement majoritaire, il est nécessaire de déterminer le nombre clef lié au nombre d'électeurs (cf figure 1). Lorsque le nombre d'électeur est **pair**, le nombre clef correspond à la médiane, c'est à dire `n//2`. Lorsque le nombre d'électeur est **impair**, le nombre clef correspond à la médiane plus un, c'est à dire `n//2 + 1`.

Une fois le nombre clef déterminé, la mention majoritaire d'un candidat est obtenue en sélectionnant dans son profil la mention approuvée par un nombre d'électeurs égal au nombre clef.

Par exemple, pour 165 électeurs, le nombre clef est le 83 e. Supposons que le profil du candidat soit `[30, 26, 21, 30, 33, 25]`. Si on utilise une échelle numérique 0,1,2,3,4, et 5, la mention majoritaire du candidat est 3, car 83 électeurs ont approuvé **au moins** la mention 3 pour le candidat, c'est-à-dire qu'ils ont attribué les mentions 0, 1, 2 ou 3. Autrement dit, cela signifie qu'une majorité d'électeurs a estimé que ce candidat valait au moins la mention 3.

E1. Créer une fonction de prototype `nb_clef(n)` qui renvoie le nombre clef pour `n` électeurs.

Solution :

```
def nb_clef(n):
  if n % 2 == 0:
    return n // 2
  else:
    return n // 2 + 1
```

E2. Créer une fonction de prototype `mention_majoritaire(profil)` renvoyant la mention majoritaire associée au profil d'un candidat sous la forme d'un type `int`. L'utilisation de la fonction `sum` n'est pas autorisée.

Solution :

```
def mention_majoritaire(profil):
      n = 0
      for v in profil:
          n += v
      clef = nb_clef(n)
      cumul = 0
      for i in range(len(profil)):
          cumul += profil[i]
          if cumul >= clef:
              return i
      return None
```

Pour départager les candidats ayant obtenu des mentions majoritaires identiques, on calcule le nombre d'électeurs qui ont attribué une mention strictement supérieure et strictement inférieure à la mention majoritaire du candidat.

E3. Créer une fonction de prototype `dessus_dessous(profil, mention)` dont les paramètres sont de type `list` pour `profil` et `int` pour `mention`. Cette fonction renvoie un **tuple** dont les deux valeurs sont :

- le nombre d'électeurs ayant attribué une mention strictement plus favorable à `mention`
- **et** le nombre d'électeurs ayant attribué une mention strictement moins favorable à `mention` au candidat ayant pour profile `profil`.

Par exemple, pour un profil de candidat [30, 26, 21, 30, 33, 25] et la mention 2, la fonction renvoie (56,88).

Solution :

```
def dessus_dessous(profil, mention):
    dessus = 0
    for i in range(mention):
      dessus += profil[i]
    dessous = 0
    for i in range(mention + 1, len(profil)):
      dessous += profil[i]
    return dessus, dessous
```

E4. Créer une fonction de prototype `meilleurs_candidats(mentions_majoritaires)` dont le paramètre est la liste des mentions majoritaires obtenues par chaque candidat. Cette fonction renvoie la liste des indices des candidats vainqueurs ex aequo, c'est-à-dire dont la mention majoritaire est la meilleure (la plus faible numériquement). Il est autorisé d'utiliser la fonction `min` de Python. Par exemple, pour quatre candidats, la liste de mentions majoritaires `[1,4,1,5]` et l'échelle numérique 0>1>2>3>4>5, cette fonction renvoie la liste de candidats `[0,2]`.

Solution :

```
def meilleurs_candidats(mentions_majoritaires):
    min_mm = min(mentions_majoritaires)
    vainqueurs = []
    for c in range(len(majm)):
        if mentions_majoritaires[c] == min_mm:
            vainqueurs.append(c)
    return vainqueurs
```

On dispose maintenant de trois listes issues de nos calculs :

- la liste `dessus` qui contient, pour chaque candidat, le cumul des votes obtenus pour des mentions strictement supérieures à la mention majoritaire du candidat.
- la liste `dessous` qui contient, pour chaque candidat, le cumul des votes obtenus pour des mentions strictement inférieures à la mention majoritaire du candidat.
- la liste `vainqueurs` qui contient la liste des vainqueurs potentiels de l'élection qui ont même mention majoritaire (liste issue de la fonction précédente).

E5. Expliquer en détail le fonctionnement de la fonction suivante.

```
def vainqueur(vainqueurs, dessus, dessous):
  while len(vainqueurs) > 1:
    ma, ia = emax_imax_parmi(dessus, vainqueurs)
    mb, ib = emax_imax_parmi(dessous, vainqueurs)
```

```
      if mb >= ma:
        vainqueurs.remove(ib)
      else:
        return ia
    return vainqueurs[0]
```

Solution : Cette fonction départage les candidats ex aequo en utilisant le rapport de force entre partisans et opposants. Tant que la liste des vainqueurs potentiels possède plus d'un élément, la fonction évalue le maximum des partisans et le maximum des opposants. Puis, si le nombre d'opposants est le nombre le plus grand, la fonction élimine le candidat correspondant. Si le nombre de partisans est maximum, alors le vainqueur est le candidat correspondant.

E6. Créer une version récursive de l'algorithme précédent.

Solution :

```
def rec_vainqueur(vainqueurs, dessus, dessous):
  if len(vainqueurs) == 1:
    return vainqueurs[0]
  else:
    ma, ia = emax_imax_parmi(dessus, vainqueurs)
    mb, ib = emax_imax_parmi(dessous, vainqueurs)
    if mb >= ma:
      vainqueurs.remove(ib)
      return rec_vainqueur(vainqueurs, dessus, dessous)
    else:
      return ia
```

F Pour la culture citoyenne

Les avantages du scrutin à jugement majoritaire sont :

- la disparition des stratégies de vote (type dilemme «vote utile»),
- le gagnant ne change pas si un perdant est ajouté ou retiré du scrutin, ce qui encourage les candidatures sans pour autant engendrer la division des voix et l'élimination d'un candidat au sein d'un même camp,
- l'électeur peut exprimer son désaccord et même rejeter une candidature qu'il estime inacceptable, ce qui évite le recours à l'abstention ou au vote blanc,
- le vote exprimé par l'électeur est nuancé, autant que le permet l'échelle des mentions,
- le dépassement des limites des scrutins de Condorcet pour lesquels il n'y a parfois pas de candidat idéal majoritaire.