# Affectation de maisons

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve : 3 heures 30 minutes

Juin 2024

**ATTENTION !**

N'oubliez en aucun cas de recopier votre $u_0$
à l'emplacement prévu sur votre fiche réponse

**Important.**

Il vous a été donné un numéro $u_0$ qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion !). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de $u_0$. Vous indiquerez vos réponses (correspondant à votre $u_0$) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures !

Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre $n$, on demande l'ordre de grandeur en fonction du paramètre, par exemple : $O(n^2)$, $O(n \log n)$,...

Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

# 1 Préliminaires

Nous allons étudier le problème d'affectation de maisons. Étant donné $n$ personnes (numérotées avec $i \in [\![0, n-1]\!]$) et $n$ maisons (numérotées avec $j \in [\![0, n-1]\!]$), l'objectif est d'affecter exactement une maison à chaque personne, en tenant compte des préférences de chaque personne.

La partie 1 est nécessaire pour pouvoir traiter le reste du sujet. Les parties 2 et 3 sont indépendantes, et proposent deux algorithmes qui tiennent compte des ordres de préférences de chaque personne. Les parties 4 et 5 sont indépendantes, et présentent deux algorithmes tenant également compte des ordres de priorité de chaque maison. Enfin, les parties 6 et 7 sont indépendantes et étudient les propriétés des affectations calculées.

Dans la plupart des questions d'implémentation, calculer la réponse pour les dernières valeurs de $n$ nécessite une implémentation efficace. Nous vous suggérons de réfléchir à la complexité en temps avant de commencer l'implémentation. Si votre algorithme n'est pas assez efficace pour calculer certaines réponses, nous vous conseillons d'aborder les questions suivantes, avant de revenir optimiser votre implémentation.

## 1.1 Notations

Dans le sujet, $n$ est un entier fixé strictement positif. Chaque question d'implémentation vous fournira plusieurs valeurs de $n$ sur lesquelles tester vos algorithmes.

Dans ce sujet, vous serez amenés à manipuler des tableaux. Un **tableau** $t$ de taille $n$ associe à chaque indice $\ell \in [\![0, n-1]\!]$ une valeur $t[\ell]$. On écrira $i \in t$ s'il existe $\ell$ tel que $t[\ell] = i$, et $i \notin t$ sinon. Enfin, nous utiliserons la notation $[v_0, v_1, \ldots, v_{n-1}]$ pour désigner le tableau $t$ tel que $t[i] = v_i$ pour tout $i \in [\![0, n-1]\!]$.

Une **permutation** de taille $n$ est un tableau qui contient tous les entiers entre 0 et $n-1$ exactement une fois. Le **rang** d'une option $i \in [\![0, n-1]\!]$ dans la permutation est l'indice auquel $i$ apparaît.

## 1.2 Génération de nombres pseudo-aléatoires

Étant donné $u_0$, on considère la suite $u$ générée par la relation de récurrence suivante :

$$u_{k+1} = (1\,103\,515\,245 \times u_k + 12\,345) \mod 2^{31}$$

L'entier $u_0$ vous est donné, et doit être recopié sur votre fiche réponse avec vos résultats. Une fiche réponse type vous est donnée en exemple, et contient tous les résultats attendus pour une valeur de $u_0$ différente de la vôtre (notée $\widetilde{u_0}$). Il vous est conseillé de tester vos algorithmes avec cet $\widetilde{u_0}$ et de comparer avec la fiche de résultats fournie. Pour chaque calcul demandé, avec le bon choix d'algorithme le calcul ne devrait demander qu'au plus de l'ordre de quelques secondes, jamais plus d'une minute.

Lors du calcul de la suite $u_k$, nous vous suggérons de faire attention au problème de dépassement d'entier. On aura souvent besoin de nombreuses valeurs consécutives de la suite $u_k$. Il est donc conseillé que votre implémentation calcule le tableau de tous les $u_k$ jusqu'à un certain rang choisi soigneusement.

**Question 1** *Calculer les valeurs* $u_k \mod 100\,000$, *avec :*

***a)*** $k = 1$ ***b)*** $k = 73$ ***c)*** $k = 1337$ ***d)*** $k = 5\,318\,008$

## 1.3 Génération de permutations pseudo-aléatoires

Dans ce sujet, il vous sera demandé de générer des permutations pseudo-aléatoires. On notera $\texttt{perm}(n, d, k)$ la permutation pseudo-aléatoire de taille $n$, générée à l'aide des valeurs $u_k, \dots, u_{k+n-2}$ de la manière suivante :

**fonction** $\texttt{perm}(n, d, k)$
  initialiser un tableau $t \leftarrow [0, 1, 2, \dots, n-1]$ de taille $n$.
  **pour** $\ell$ *de* $0$ *à* $n-2$ **faire**
    définir $r \leftarrow \ell + (u_{k+\ell} \mod \min(d, n-\ell))$
    échanger les valeurs de $t[\ell]$ et $t[r]$
  **fin**
  **renvoyer** $t$
**fin**

**Question 2** *Calculer la permutation* $\texttt{perm}(n, 5, 73)$ *avec :*

***a)*** $n = 3$ ***b)*** $n = 5$ ***c)*** $n = 8$ ***d)*** $n = 10$

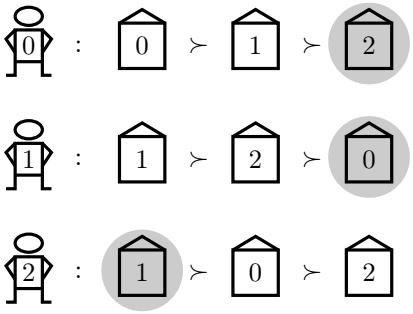
## 1.4 Ordres de préférences et affectation de maisons

Dans le problème d'affectation de maison, chaque personne a un **ordre de préférence**, qui est une permutation $t$ de taille $n$, où $t[0]$ est le premier choix, $t[1]$ est le second choix, ..., et $t[n-1]$ est le dernier choix. On note $\texttt{pref}_i$ l'ordre de préférence de la personne $i$. Dans l'intégralité du sujet, les ordres de préférences seront :

$$\forall i \in [\![0, n-1]\!], \qquad \texttt{pref}_i = \texttt{perm}(n, 50, n \cdot i).$$

Une **affectation** des maisons est une permutation $s$ de taille $n$, où $s[j]$ est le numéro de la personne qui possède la maison $j$. Afin de finir notre échauffement, nous allons étudier une **affectation aléatoire** des maisons aux personnes :

$$\texttt{affectAlea} = \texttt{perm}(n, n, n^2)$$




Les ordres de préférences sont :
- $\texttt{pref}_0 = [0, 1, 2]$
- $\texttt{pref}_1 = [1, 2, 0]$
- $\texttt{pref}_2 = [1, 0, 2]$

On a l'affectation
$\texttt{affectAlea} = [1, 2, 0]$
avec un rang total de $4 = 2 + 2 + 0$.

FIGURE 1 – Affectation aléatoire, avec $n = 3$ et $\widetilde{u_0} = 489$.

Enfin, pour mesurer l'efficacité d'une affectation, on regarde pour chaque personne le rang de sa maison dans son ordre de préférence (entre 0 et $n-1$). Le **rang total** d'une affectation est la

somme des rangs des maisons dans les ordres de leurs propriétaires. Ainsi, une affectation qui associerait à chaque personne sa maison préférée est de rang total 0. La figure 1 illustre sur un exemple le rang total de l'affectation `affectAlea`.

**Question 3** *Calculer le rang total de* `affectAlea` *avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 100$ ***d)*** $n = 1000$

## 2 Algorithme des dictateurs successifs

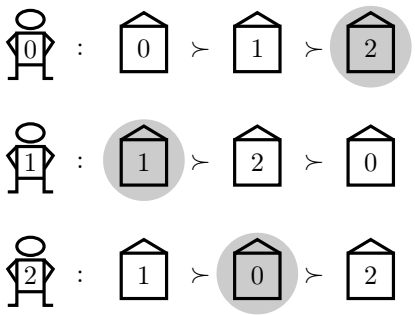
L'affectation aléatoire des maisons ne tient pas compte des préférences de chaque personne. Dans cette partie, nous allons étudier l'**algorithme des dictateurs successifs** : chaque personne choisit successivement sa maison préférée parmi celles qui sont toujours disponibles. Étant donnée une permutation $r$, on note $\texttt{affectDictSucc}(r)$ l'affectation $s$ obtenue par l'algorithme des dictateurs successifs, dans l'ordre $r$.

**fonction** $\texttt{affectDictSucc}(n, r)$
  initialiser un tableau $s \leftarrow [\bot, \dots, \bot]$ de taille $n$.
  **pour** *chaque personne* $i = r[0], r[1], ..., r[n-1]$ **faire**
    soit $j$ la maison préférée de $i$, telle que $s[j] = \bot$.
    affecter $s[j] \leftarrow i$.
  **fin**
  **renvoyer** $s$
**fin**

Pour que l'algorithme ne favorise pas les personnes en fonction de leur indice, nous allons utiliser un **ordre aléatoire** :

$$\texttt{ordreAlea} = \texttt{perm}(n, n, n^2)$$

La figure 2 illustre sur un exemple l'algorithme des dictateurs successifs, et donne le rang total de l'affectation calculée.




On a l'ordre aléatoire
$\texttt{ordreAlea} = [1, 2, 0]$.
Dans l'algorithme, successivement :
— 1 choisit la maison 1,
— 2 choisit la maison 0,
— 0 choisit la maison 2,
On obtient donc l'affectation
$\texttt{affectDictSucc}(n, \texttt{ordreAlea}) = [2, 1, 0]$
avec un rang total de $3 = 2 + 0 + 1$.

FIGURE 2 – Algorithme des dictateurs successifs, avec $n = 3$ et $\widetilde{u_0} = 489$.

**Question 4** *Calculer le rang total de* $\texttt{affectDictSucc}(n, \texttt{ordreAlea})$ *avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 100$ ***d)*** $n = 1000$

**Question à développer pendant l'oral 1** *Quelle est la complexité en temps de votre implémentation de l'algorithme* `affectDictSucc`*, dans le pire des cas, en fonction de* $n$ *?*

# 3 Algorithme des cycles d'échange

Dans la figure 1, les personnes 0 et 1 aimeraient échanger leurs maisons. Nous allons maintenant étudier un algorithme permettant d'améliorer cette affectation. Dans le problème de ré-affectation des maisons, chaque personne possède initialement une maison différente. Nous souhaitons trouver une affectation dans laquelle chaque personne reçoit une maison qu'elle préfère (ou qui est identique) à la maison qu'elle possédait initialement.

Informellement, l'**algorithme des cycles d'échange** procède de la manière suivante : chaque personne pointe du doigt la personne qui possède sa maison préférée, on calcule un cycle dans le graphe orienté ainsi obtenu, on donne à chaque personne du cycle sa maison préférée, et on recommence sans les maisons et personnes du cycle. Étant donnée une affectation initiale $r$, on note `affectCycle`$(n, r)$ la ré-affectation $s$ obtenue par l'algorithme des cycles d'échange décrit ci-dessous.

**fonction** `affectCycle`$(n, r)$
  initialiser un tableau $s \leftarrow [\bot, \dots, \bot]$ de taille $n$.
  **tant que** *une des cases de* $s$ *contient* $\bot$ **:**
    définir un graphe $G$, sur l'ensemble de nœuds $V = \{i : i \notin s\}$.
    **pour** *chaque personne* $i \in V$ **faire**
      soit $j_i$ la maison préférée de $i$, telle que $s[j_i] = \bot$.
      ajouter l'arc $i \to r[j_i]$ au graphe $G$.
    **fin**
    calculer un cycle $C$ dans le graphe $G$.
    **pour** *chaque personne* $i \in C$ **faire**
      affecter $s[j_i] \leftarrow i$.
    **fin**
  **fin**
  **renvoyer** $s$
**fin**

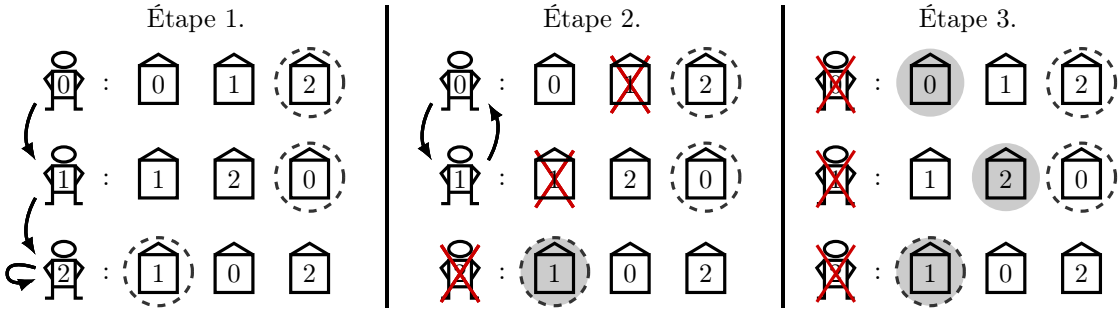
On remarque qu'à chaque itération, si une maison $j$ n'a pas encore été ré-affectée ($s[j] = \bot$), alors son propriétaire n'a pas encore reçu de maison ($r[j] \in V$). Par conséquent, le graphe $G$ est bien défini.

**Question à développer pendant l'oral 2** *Dans le graphe* $G$ *défini à chaque itération, chaque sommet* $i \in V$ *a exactement un arc sortant. Démontrer que le graphe* $G$ *possède au moins un cycle* $C$*, et proposez une procédure simple pour le calculer.*

On admettra que la ré-affectation calculée par l'algorithme des cycles d'échange ne dépend pas de l'ordre dans lequel les cycles sont choisis (en effet, si un cycle n'est pas choisi à une itération, il existera toujours à la suivante).

**Question 5** *Calculer le rang total de* `affectCycle`$(n,$ `affectAlea`$)$ *avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 100$ ***d)*** $n = 1000$




L'affectation initiale `affectAlea` $= [1, 2, 0]$ est représentée en pointillé. À la fin de l'algorithme, on obtient l'affectation `affectCycle`$(n,$ `affectAlea`$) = [0, 2, 1]$, avec un rang total de $1 = 0 + 1 + 0$.

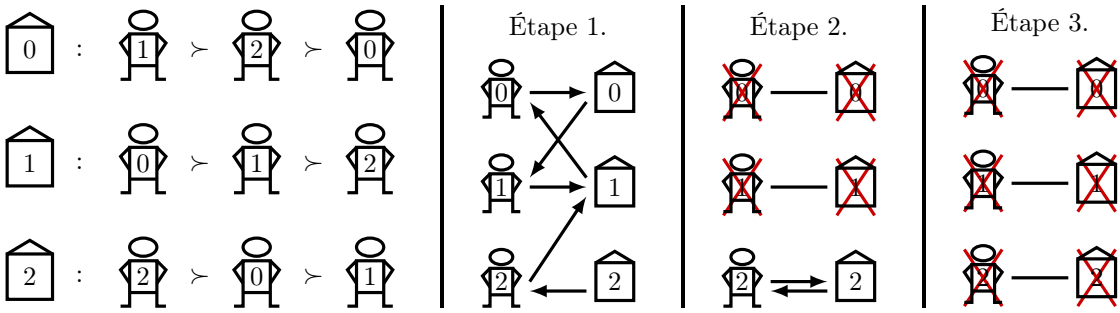
FIGURE 3 – Algorithme des cycles d'échange, avec $n = 3$ et $\widetilde{u_0} = 489$.

**Question à développer pendant l'oral 3** *Quelle est la complexité en temps de votre implémentation de l'algorithme* `affectCycle`*, dans le pire des cas, en fonction de* $n$ *?*

## 4 Algorithme des cycles d'échange généralisé

Dans cette quatrième partie, nous allons étudier le problème d'affectation lorsque chaque maison a un ordre de priorité. Plus précisément, chaque maison $j$ possède une permutation $p_j$, où $p_j[0]$ est la personne qui a la plus haute priorité, $p_j[1]$ a la seconde priorité, ..., et $p_j[n-1]$ a la plus faible priorité.

Nous **généralisons l'algorithme des cycles d'échange** de la manière suivante : chaque personne pointe du doigt sa maison préférée, chaque maison pointe du doigt sa personne préférée, on calcule un cycle dans le graphe orienté ainsi obtenu, on donne à chaque personne du cycle sa maison préférée, et on recommence sans les maisons et personnes du cycle. On note `affectCyclePrio`$(n, p_0, \ldots, p_{n-1})$ l'affectation obtenue par l'algorithme des cycles d'échange généralisé.




On obtient `affectCyclePrio`$(3, p_0, p_1, p_2) = [0, 1, 2]$, avec un rang total de $2 = 0 + 0 + 2$.

FIGURE 4 – Algorithme des cycles d'échange généralisé, avec $n = 3$ et $\widetilde{u_0} = 489$.

**Question à développer pendant l'oral 4** *Quelle affectation obtient-on si toutes les maisons ont le même ordre de priorité ? Et si toutes les maisons ont des premiers choix distincts ?*

À nouveau, on admettra que la ré-affectation calculée par l'algorithme des cycles d'échange généralisé ne dépend pas de l'ordre dans lequel les cycles sont choisis.

**Question 6** *Pour chaque maison $j$, on construit l'ordre de priorité $p_j = \texttt{perm}(n, n, n^2 + n \cdot j)$. Calculer le rang total de* $\texttt{affectCyclePrio}(n, p_0, \ldots, p_{n-1})$ *avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 100$ ***d)*** $n = 1000$

# 5 Algorithme d'acceptation différée

Dans la figure 4, la personne 2 aurait préféré avoir la maison 0, dans laquelle elle a une priorité plus élevée que la personne 0. Nous allons essayer de résoudre ce problème avec l'algorithme d'acceptation différée.

Informellement, l'**algorithme d'affectation différée** procède de la manière suivante. À chaque itération, on choisit une personne qui n'a pas encore de maison. On la laisse choisir sa maison préférée parmi les maisons non-affectées, et les maisons où elle a une priorité plus élevée que le propriétaire actuel. L'algorithme se termine lorsque chaque personne a une maison. Plus formellement, on note $\texttt{affectAccDiff}(n, p_0, \ldots, p_{n-1})$ l'affectation $s$ obtenue par l'algorithme d'acceptation différée décrit ci-dessous.

**fonction** $\texttt{affectAccDiff}(n, p_0, \ldots, p_{n-1})$
  initialiser un tableau $s \leftarrow [\bot, \ldots, \bot]$ de taille $n$.
  **tant que** *il existe une personne* $i \notin s$ **:**
    soit $j$ la maison préférée de $i$, telle que :
      $s[j] = \bot$ ou $i$ est mieux classé que $s[j]$ dans $p_j$.
    affecter $s[j] \leftarrow i$.
  **fin**
  **renvoyer** $s$
**fin**

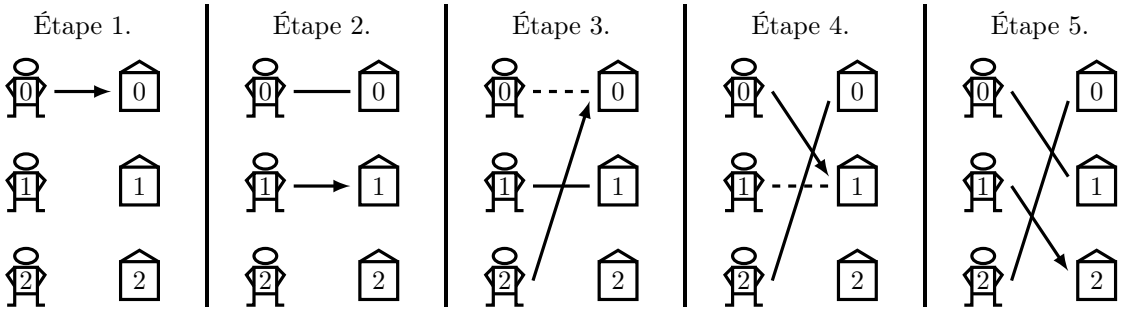
**Question à développer pendant l'oral 5** *Donner une borne supérieure sur le nombre d'itérations de la boucle principale.*

On admettra que l'affectation calculée par l'algorithme ne dépend pas de l'ordre dans lequel on choisit les personnes à affecter. Dans notre implémentation, nous choisirons à chaque itération la personne $i \notin s$ avec l'indice le plus petit.

**Question 7** *Pour chaque maison $j$, on construit l'ordre de priorité $p_j = \texttt{perm}(n, n, n^2 + n \cdot j)$. Calculer le rang total de* $\texttt{affectAccDiff}(n, p_0, \ldots, p_{n-1})$ *avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 100$ ***d)*** $n = 1000$

**Question à développer pendant l'oral 6** *Quelle est la complexité en temps de votre implémentation de l'algorithme* `affectAccDiff`*, dans le pire des cas, en fonction de $n$ ?*

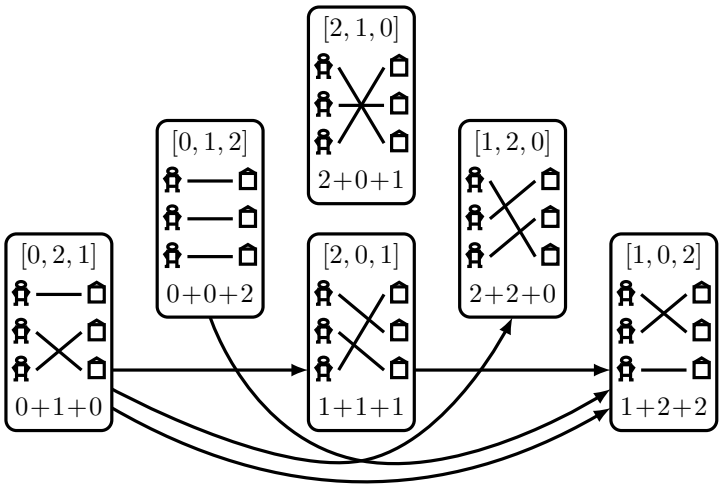



À chaque étape, la flèche représente le choix de $i$ et $j$, et une ligne pointillée représente l'ancienne affectation de la maison $j$. On obtient $\texttt{affectAccDiff}(3, p_0, p_1, p_2) = [2, 0, 1]$, avec un rang total de $3 = 1 + 1 + 1$.

FIGURE 5 – Algorithme d'acceptation différée, avec $n = 3$ et $\widetilde{u_0} = 489$.

## 6 Optimum de Pareto

On dit qu'une affectation $s'$ **améliore** une affectation $s$, et on notera $s' \succeq s$, si chaque personne reçoit dans $s'$ une maison qu'elle préfère (ou qui est identique) à celle qu'elle reçoit dans $s$. Étant donné une affectation $s$, on dira que $s$ est **améliorable** s'il existe $s' \neq s$ tel que $s' \succeq s$, et sinon on dira que $s$ est un **optimum** (de Pareto).




Sous chaque affectation est renseigné son rang total. Un arc entre deux affectations $s' \to s$ signifie que $s'$ améliore $s$. Il existe trois optima (affectations sans arc entrant).

FIGURE 6 – Toutes les affectation possibles, avec $n = 3$ et $\widetilde{u_0} = 489$.

Nous allons admettre que les propriétés suivantes sont vérifiées :

- l'algorithme des dictateurs successifs renvoie toujours un optimum ;
- tout optimum peut être obtenu avec l'algorithme des cycles d'échange (avec un choix astucieux d'affectation initiale) ;

— toute affectation obtenue par l'algorithme des cycles d'échange peut aussi être obtenue par l'algorithme des dictateurs successifs (avec un choix astucieux d'ordre).

Ces trois propriétés nous permettent de caractériser l'ensemble des optima. En utilisant cette caractérisation, nous souhaitons calculer le nombre d'optima.

**Question 8** *Calculer le nombre d'optima différents, avec :*

***a)*** $n = 3$ ***b)*** $n = 5$ ***c)*** $n = 8$

***d)*** $n = 10$ ***e)*** $n = 12$ ***f)*** $n = 15$

**Question à développer pendant l'oral 7** *Expliquer votre algorithme pour compter le nombre d'optima, ainsi que les éventuelles optimisations améliorant le temps de calcul.*

# 7 Stabilité

Dans le problème d'affectation avec des priorités (sections 4 et 5), on dit qu'une personne $i$ et une maison $j$ bloquent une affectation $s$ si $i$ et $j$ se préfèrent mutuellement à leurs affectations respectives. On dit qu'une affectation est **stable** si elle n'a pas de couple $(i, j)$ bloquant.

Dans la figure 4, la personne $i = 2$ et la maison $j = 0$ bloquent l'affectation (car $i$ préfère $j$ à la maison 2, et $j$ préfère $i$ à la personne 0), qui n'est donc pas stable. Dans la figure 5, aucun couple ne bloque l'affectation, qui est donc stable. On admettra que l'algorithme d'affectation différée renvoie toujours une affectation stable.

Dans l'exemple avec $n = 3$ et $\widetilde{u_0} = 489$, il existe deux affectations stables : $[2, 0, 1]$ renvoyée par `affectAccDiff`, et $[1, 0, 2]$. Nous souhaitons calculer le nombre d'affectations stables.

**Question 9** *Pour chaque maison $j$, on construit l'ordre de priorité $p_j = \texttt{perm}(n, n, n^2 + n \cdot j)$. Calculer le nombre d'affectation stables, avec :*

***a)*** $n = 3$ ***b)*** $n = 10$ ***c)*** $n = 15$

***d)*** $n = 20$ ***e)*** $n = 30$ ***f)*** $n = 50$

**Question à développer pendant l'oral 8** *Expliquer votre algorithme pour compter le nombre d'affectations stables, ainsi que les éventuelles optimisations améliorant le temps de calcul.*

# Fiche réponse type : Affectation de maisons

$\widetilde{u_0}$ : 489

**Question 1**

| | |
|---|---|
| a) | 71 502 |
| b) | 88 966 |
| c) | 82 070 |
| d) | 18 545 |

**Question 2**

| | |
|---|---|
| a) | [0, 2, 1] |
| b) | [1, 4, 3, 0, 2] |
| c) | [1, 4, 3, 7, 6, 5, 0, 2] |
| d) | [1, 4, 3, 7, 6, 9, 0, 2, 8, 5] |

**Question 3**

| | |
|---|---|
| a) | 4 |
| b) | 51 |
| c) | 4537 |
| d) | 499 406 |

**Question 4**

| | |
|---|---|
| a) | 3 |
| b) | 17 |
| c) | 2878 |
| d) | 478 941 |

**Question 5**

| | |
|---|---|
| a) | 1 |
| b) | 19 |
| c) | 2922 |
| d) | 478 674 |

**Question 6**

| | |
|---|---|
| a) | 2 |
| b) | 16 |
| c) | 2902 |
| d) | 478 628 |

**Question 7**

**a)** 3

**b)** 19

**c)** 3368

**d)** 496 738

**Question 8**

**a)** 3

**b)** 9

**c)** 1034

**d)** 3325

**e)** 8434

**f)** 130 529

**Question 9**

**a)** 2

**b)** 2

**c)** 4

**d)** 11

**e)** 8

**f)** 20

# Fiche réponse : Affectation de maisons

**Nom, prénom, $u_0$ :** .....

**Question 1**

a)

b)

c)

d)

**Question 2**

a)

b)

c)

d)

**Question 3**

a)

b)

c)

d)

**Question 4**

a)

b)

c)

d)

**Question 5**

a)

b)

c)

d)

**Question 6**

a)

b)

c)

d)

**Question 7**

**a)**

**b)**

**c)**

**d)**

**Question 8**

**a)**

**b)**

**c)**

**d)**

**e)**

**f)**

**Question 9**

**a)**

**b)**

**c)**

**d)**

**e)**

**f)**