« Amida-Kuji»

Année 2018-2019

Banhegyi Kilian, Choulet Rachel et Raquin Virgile, élèves de 1èreS.

Encadrés par Mme Martinelli Bousquet et Mme Gotte.

Lycée Jean Puy à Roanne (42300)

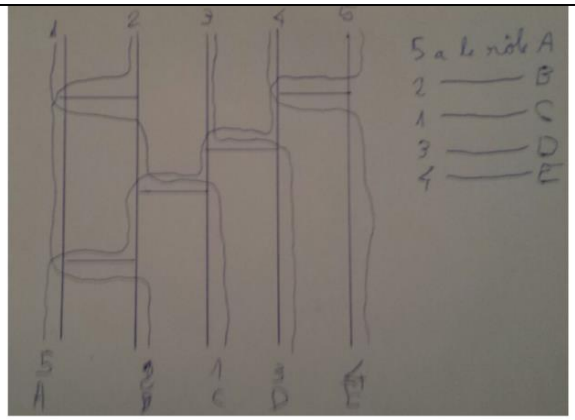
Nom du chercheur : M. Chardard (Jean Monnet à St Etienne)

1. Présentation du sujet

Voici le sujet proposé par notre chercheur :

Pour tirer au sort une répartition des rôles, les japonais utiliseraient le système suivant :

- On tire autant de traits verticaux que de participants.
- On écrit en-haut des traits le nom des participants et en-bas les rôles à attribuer.
- Chaque participant trace un ou plusieurs traits horizontaux reliant deux traits verticaux.
- Pour chaque participant, on trace un chemin qui en part. On suit le trait vertical. Dès que l'on rencontre un trait horizontal, on le suit jusqu'à arriver sur l'autre trait vertical que l'on descend jusqu'à arriver à un autre trait horizontal, ou à arriver au rôle attribué à ce participant.



Toutes les répartitions de rôles sont-elles possibles avec ce système ?
Si oui, combien de traits horizontaux sont nécessaires ?

2. Annonce des conjectures et résultats obtenus :

On considère un nombre donné n de joueurs. On a conjecturé que lorsque le nombre b de barrettes (traits verticaux) utilisées (où $b \geq n$) est pair, les différents positionnements des b barrettes permettraient d'obtenir la moitié des distributions de rôles possibles et lorsque b est impair, les différents positionnements permettraient d'obtenir l'autre moitié des distributions de rôles possibles.

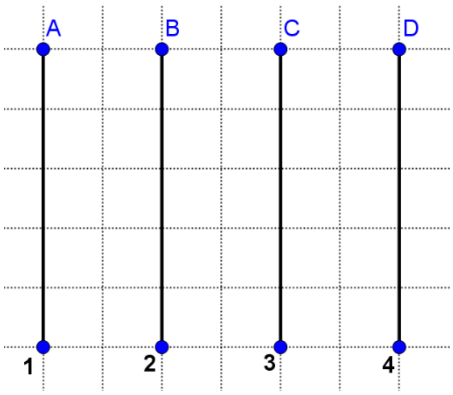
3. Texte de l'article

Partie 1 : Recherches préliminaires

Etape 1 : Découverte du jeu.

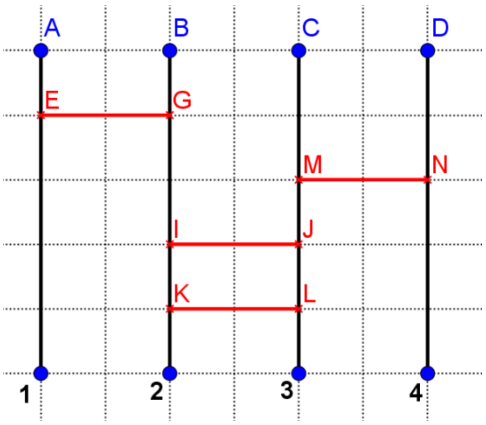
Les Amida-kuji (弥陀 , «loterie Amida ») est une méthode de loterie conçue pour créer des paires aléatoires entre deux ensembles, tant que le nombre d'éléments dans chaque ensemble est le même. Ce système est souvent utilisé pour répartir des choses entre les gens, par exemple des tâches qui sont alors attribuées de manière juste et aléatoire de cette manière.

Un amida-kuji est constitué de lignes verticales dont le nombre est égal au nombre de personnes qui jouent. Dans l'exemple ci-contre, on considère quatre joueurs A, B, C et D.
Au bas de chaque ligne se trouve un élément - une chose qui doit être jumelée à un joueur. Dans l'exemple les éléments 1, 2, 3 et 4.



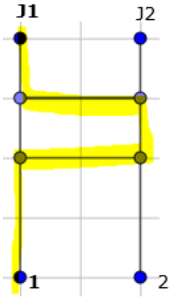
Chaque participant trace un ou plusieurs traits verticaux, que l'on appellera **des barrettes** reliant deux traits verticaux. Deux traits verticaux n'ont pas le droit d'être tracés à une même hauteur.

Ensuite pour chaque participant, saisissez sa ligne en haut et suivez cette ligne vers le bas. Si vous rencontrez une ligne horizontale, suivez-la pour atteindre une autre ligne verticale et continuez vers le bas. Répétez cette procédure jusqu'à la fin de la ligne verticale. Ensuite, le joueur reçoit la chose écrite au bas de la ligne.



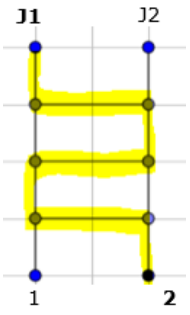
Exemple : Les quatre participants ont tracé une barrette chacun.
Pour connaître les éléments associés aux participants il faut suivre successivement les segments :
Pour A : [AE], [EG], [GI], [IJ], [JL], [LK] et [K2] A est associé à 2
Pour B : [BG], [GE] et [E1] B est associé à 1
Pour C : [CM], [MN] et [N4] C est associé à 4
Pour D : [DN], [NM], [MJ], [JI], [IK], [KL] et [L3] D est associé à 3.

Etape 2 : Premières recherches à la main



Au début de notre recherche, nous avons cherché à conjecturer pour un petit nombre de joueurs (n=2 ou n=3) si l'on pouvait obtenir toutes les répartitions de rôles possibles à l'aide de b barrettes (où $b \geq n$).

- On a commencé avec n=2 joueurs et b= 2 barrettes, il existe alors une seule façon de positionner les deux barrettes.
  On obtient la répartition J1- en position 1 et J2 en positon 2 notée $\begin{pmatrix} J1 & J2 \\ 1 & 2 \end{pmatrix}$



- Ensuite, avec n=2 joueurs et b= 3 barrettes, il existe également une seule façon de les disposer les trois barrettes.
  On obtient la répartition $\begin{pmatrix} J1 & J2 \\ 2 & 1 \end{pmatrix}$
- Ensuite, avec n=2 joueurs et b= 4 barrettes, ou b=6, ou b=8…, b pair, on conjecture que la distribution sera équivalente au cas b=2.
- Et avec n=2 joueurs et b= 5 barrettes, ou b=7, ou b=9…, b impair, on conjecture que la distribution sera équivalente au cas b=3.

On a alors conjecturé que la répartition pour n=2 joueurs dépendait du nombre b de barrettes, si le nombre de barrettes est pair on obtient la répartition $\begin{pmatrix} J1 & J2 \\ 1 & 2 \end{pmatrix}$ et si le nombre de barrettes est impair on obtient l'autre type répartition $\begin{pmatrix} J1 & J2 \\ 2 & 1 \end{pmatrix}$.

Ainsi pour le cas n=2 joueurs en considérant tous les nombres de barrettes b possibles (b ≥ n) il semble que l'on peut obtenir toutes les répartitions possibles de joueurs. Par contre avec un nombre imposé de barrettes, on ne peut pas obtenir toutes les distributions possibles. (1)

Après avoir étudié les répartitions possibles pour n=2 joueurs, nous avons voulu savoir si la règle concernant la parité de b restait valable pour n>2.
Les recherches faites à la main étaient trop longues et fastidieuses. Nous avons alors pensé à utiliser un algorithme.

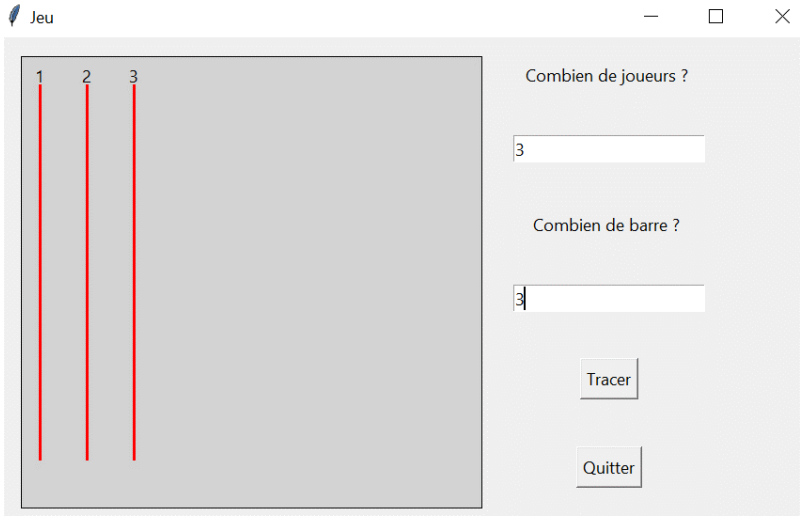
Etape 3 : Recherches à l'aide d'un logiciel

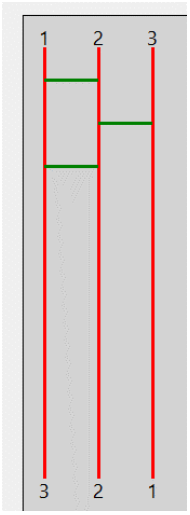
Nous avons souhaité étudier le cas de n=3 joueurs.
Grâce à un programme présenté plus loin, on a pu étudier l'ensemble des dispositions de barrettes possibles.
Nous avons écrit en langage naturel puis sous Python une partie du programme donné ci-dessous. Ce travail est expliqué dans la partie 2. Nos professeurs nous ont aidé à compléter et modifié ce programme pour qu'il soit plus efficace.

```
#création de la fenêtre
fenetre=Tk()
fenetre.title('Jeu')
fenetre.geometry('710x510+200+100')

#création du canevas
grille=Canvas(fenetre,width=550, height=500)
grille.grid(rowspan=6,column=0)
grille.create_rectangle(30,20,520,500,fill='lightgrey')

#initialisation
grille.cpt=0
grille.stock_numcol=[]
grille.stock_y=[0]

#tracer des lignes verticales
def tracer():
    if int(case_reponse.get())<=10:
        for i in range(1,int(case_reponse.get())+1):
            grille.create_text(i*50,40,text=str(i))
            grille.create_line(i*50,50,i*50,450, fill='Red', width=3)

#tracer des lignes horizontales
def horizontal(event):

    # calcul des coordonnées des extrémités du segments (multiple de 10 selon les y)
    y=10*(event.y//10)
    x1=50*(event.x//50)
    x2=x1+50
    #vérification avant le tracé qu'on est dans la zone, qu'il reste des barres à tracer et qu'elles sont bien les unes sous les autres (donc plus basse que le dernier élément)

    if 50<event.x<50*int(case_reponse.get()) and grille.cpt<int(case_reponse2.get()) and y>grille.stock_y[len(grille.stock_y)-1] :
        grille.create_line(x1,y,x2,y,fill='green', width=3)
        #stockage du numéro des colonnes entre lesquelles on a tracé
        grille.stock_numcol.append((x1//50,x1//50+1))
        #stockage de l'ordonnée
        grille.stock_y.append(y)

        grille.cpt+=1

#une fois les barettes placées
    if grille.cpt==int(case_reponse2.get()):
        #création de la liste d'arrivée
        arrivee=['' for i in range(int(case_reponse.get()))]

        #balayage des joueurs
        for i in range(1,int(case_reponse.get())+1):
            #mise en mémoire du joueur actuel
            rep=i
            #balayage des barettes
            for j in range(int(case_reponse2.get())):
                #si le numéro du joueur est dans la barette, on échange son numéro avec celui de l'autre colonne et on passe au suivant
                if i == grille.stock_numcol[j][0]:
                    i=grille.stock_numcol[j][1]
                elif i == grille.stock_numcol[j][1]:
                    i=grille.stock_numcol[j][0]

            #stockage de la position d'arrivée du joueur actuel
            arrivee[i-1]=rep
```

```
        #affichage de la liste d'arrivée
        for i in range(1,int(case_reponse.get())+1):
            grille.create_text(i*50,460,text=str(arrivee[i-1]))


#saisie du nombre de barres verticales
saisie=Label(fenetre,text='Combien de joueurs ? ')
saisie.grid(row=0,column=1)

case_reponse=Entry(fenetre)
case_reponse.grid (row=1, column=1)

#saisie du nombre de barres horizontales
saisie=Label(fenetre,text='Combien de barre ? ')
saisie.grid(row=2,column=1)

case_reponse2=Entry(fenetre)
case_reponse2.grid (row=3, column=1)

#tracer des barres verticales
bouton_valider=Button(fenetre, text='Tracer', command=tracer)
bouton_valider.grid(row=4, column=1)

#tracé des barres horizontales à l'emplacement de la souris
grille.bind('<Button-1>',horizontal)

#bouton quitter
bouton_quitter=Button(fenetre,text='Quitter',command=fenetre.destroy)
bouton_quitter.grid(row=5, column=1)
#raffraichissement
fenetre.mainloop()
```

Lorsqu'on exécute ce programme, une fenêtre s'ouvre. On tape le nombre de joueurs et de barrettes souhaités. Les traits verticaux apparaissent et avec la souris on clique aux endroits où l'on souhaite faire apparaître des barrettes





Le programme affiche en bas des traits verticaux à quel élément est associé chaque joueur. Dans l'exemple ci-dessus, ce positionnement de barrettes conduit à la répartition $\begin{pmatrix} J1 & J2 & J3 \\ 3 & 2 & 1 \end{pmatrix}$.

- Nous avons donc utilisé ce logiciel pour tester toutes les positions de **b= 3 barrettes** possibles.

Tous les cas ayant été traités, on a relevé les répartitions $\begin{pmatrix} J1 & J2 & J3 \\ 3 & 2 & 1 \end{pmatrix}$, $\begin{pmatrix} J1 & J2 & J3 \\ 1 & 3 & 2 \end{pmatrix}$ et $\begin{pmatrix} J1 & J2 & J3 \\ 2 & 1 & 3 \end{pmatrix}$.

- Ensuite nous avons utilisé le logiciel pour tester toutes les positions de **b=4 barrettes** possibles.

Tous les cas ayant été traités, on a relevé les répartitions $\begin{pmatrix} J1 & J2 & J3 \\ 1 & 2 & 3 \end{pmatrix}$, $\begin{pmatrix} J1 & J2 & J3 \\ 2 & 3 & 1 \end{pmatrix}$ et $\begin{pmatrix} J1 & J2 & J3 \\ 3 & 1 & 2 \end{pmatrix}$.

Lorsqu'on a poursuivi en étudiant le cas de n=3 joueurs avec n> 4 barrettes, on s'est rendu compte que l'on obtenait dans de nombreux cas des distributions que l'on avait déjà relevées.
On s'est donc intéressé à la « simplification » des Amida-kuji développée dans la partie 3 de l'article.

- Ainsi n=2 joueurs et b= 6 barrettes, ou b=8, ou b=10…, b pair, on conjecture que les répartitions possibles sont celles obtenues avec b=4.

- Et avec n=2 joueurs et b= 5 barrettes, ou b=7, ou b=9…, b impair, on conjecture que les répartitions possibles sont celles obtenues avec b=3.

Ainsi pour le cas n=3 joueurs en considérant tous les nombres de barrettes b possibles (b ≥ n) il semble que l'on peut obtenir toutes les répartitions possibles de joueurs. Par contre avec un nombre imposé de barrettes, on ne peut pas obtenir toutes les distributions possibles.

Partie 2 : Ecriture d'un algorithme

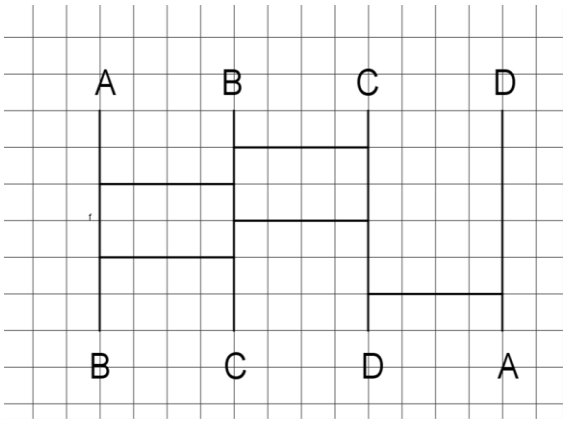
Etape 1 : En langage naturel

La première partie de nos recherches réalisée « à la main » était assez laborieuse et nous a conduits à écrire un algorithme en langage naturel qui permet de donner la répartition de 4 joueurs en suivant les barrettes positionnées par les joueurs.

Pour cela on traduit l'Amida-kuji et ses barrettes comme un tableau rempli de 0 ou de 1 selon la présence d'une barrette ou non.

Exemple : Il y a ci-dessous 5 barrettes qui définissent 5 « étages » dans l'Amida-Kuji

L'amida-kuji suivant :



se traduit par le tableau :

		A	**B**	**C**	**D**	
L i / k	L0	**L1**	**L2**	**L3**	**L4**	L5
1	0	**0**	**1**	**1**	**0**	0
2	0	**1**	**1**	**0**	**0**	0
3	0	**0**	**1**	**1**	**0**	0
4	0	**1**	**1**	**0**	**0**	0
5	0	**0**	**0**	**1**	**1**	0

Notations :
N le nombre de barrettes N≥4 i le numéro de liste k le rang (étape) J le numéro du joueur

L'algorithme suivant affiche la répartition associée au tableau qu'il faut préalablement entrer.

```
Début Algorithme
J←1
k←1
i←1
Tant que J≤4
|       Tant que k≤N
|       |       Si Li[k]=0 alors k←k+1
|       |       Sinon
|       |               Si Li-1[k]=1 alors i←i-1  et  k←k+1
|       |               Sinon i←i+1 et  k←k+1
|       Afficher J, i
Fin
Fin Algorithme
```

Etape 2 : En langage Python

Nous avons traduit notre algorithme en un programme écrit sous Python :

<tmp 1> amida kuji.py

```
J=1
k=1
i=1
n= eval(input("Combien de barrettes souhaitez-vous placer?"))
nb_joueurs= eval(input("Combien y a-t-il de joueurs?"))
while J<=nb_joueurs:
    while k<=n:
        if Li[k]==0 :
            k=k+1
        else :
            if Li-1[k]==1 :
                i=i-1
                k=k+1
            else :
                i=i+1
                k=k+1
print("Le joueur",J)
print("arrive dans la colone",i)
```

Par la suite, nos professeurs de mathématiques nous ont confié le programme plus complet décrit dans la Partie 1 de l'article.

Cet algorithme nous a permis de réaliser de nombreux tests rapidement et cela nous a fortement aidés à comprendre le fonctionnement de ce jeu. C'est notamment grâce à lui que les règles de simplification ont été trouvées.

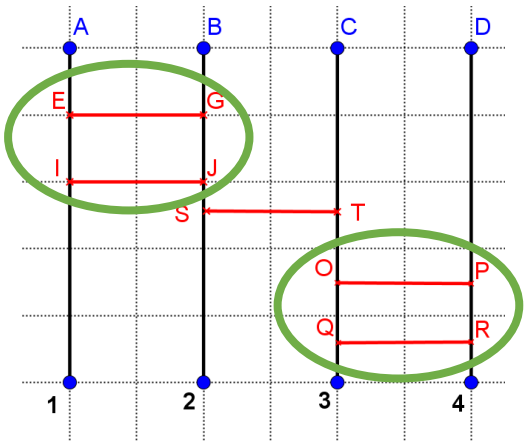
Partie 3 : Simplification d'un Amida-Kuji

Certains groupes de barrettes créent des répartitions équivalentes à d'autres groupes de barrettes. Et certains conduisent à associer les joueurs à l'élément placé sous eux verticalement. En ce sens, certains groupes de barrettes peuvent être « effacés » de l'Amida-Kuji alors allégé en barrettes. C'est **une simplification** de l'Amida-Kuji.

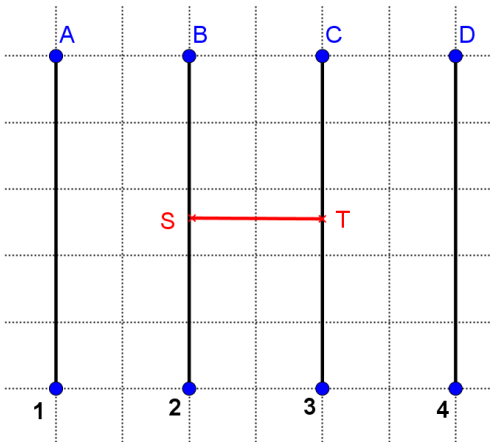
- **CAS 1**

Lorsque deux barrettes sont placées l'une sous l'autre (sans extrémité de barrettes entre elles). Ces barrettes peuvent être effacées de l'Amida-Kuji sans modifier la répartition associée. (2)

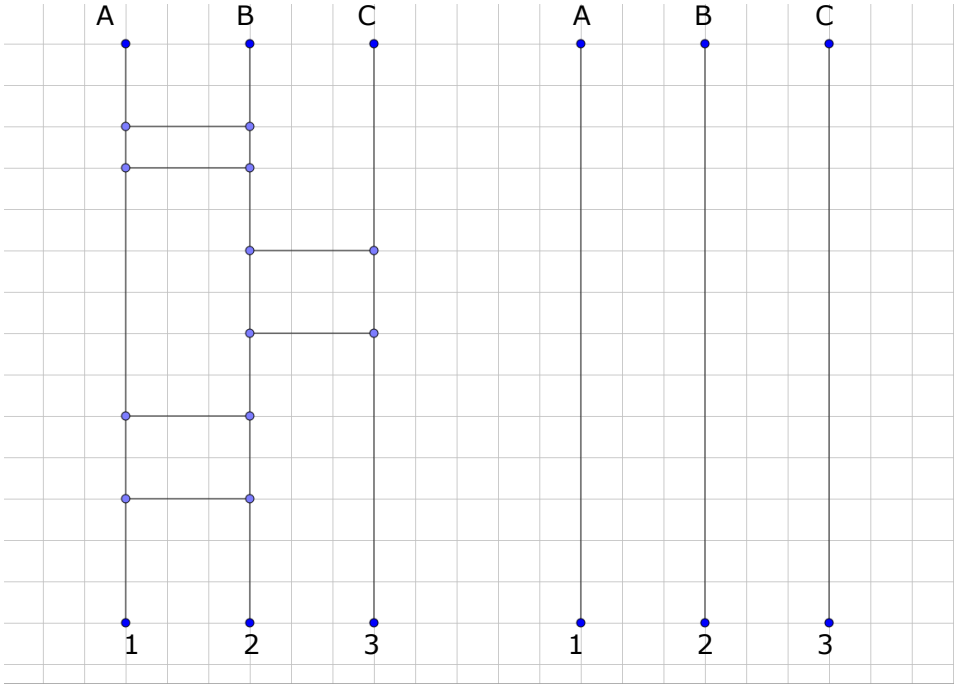
Exemple 1 : Ci-dessous les barrettes [EG] et [IJ] peuvent être enlevées ainsi que les barrettes [OP] et [QR]



On obtient comme Amida-kuji équivalent et simplifié :



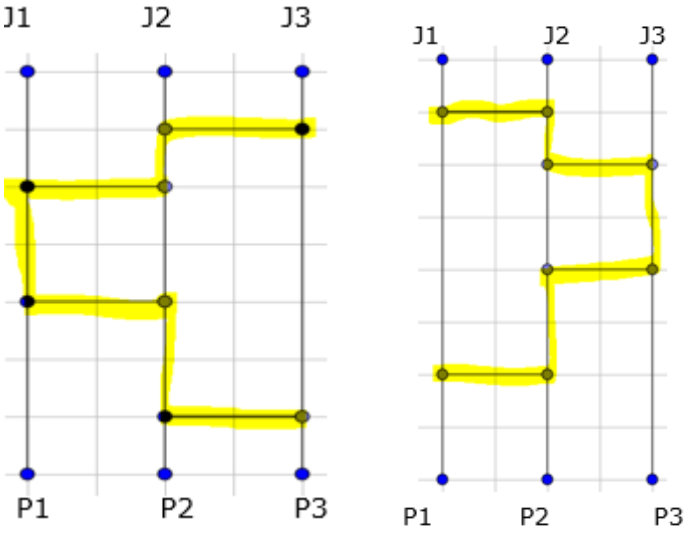
Exemple 2 : Les deux amida-kuji ci-dessous sont également équivalents :



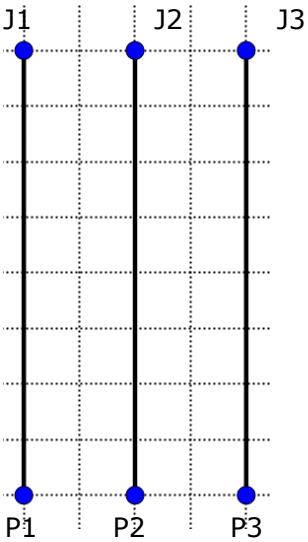
- **CAS 2**

Lorsque quatre barrettes sont placées sur trois traits verticaux adjacents de sorte à faire apparaître un polygone en « T » couché dans un sens ou un autre : ou , on peut supprimer les quatre barrettes horizontales sans modifier la répartition associée à l'Amida-Kuji. Pour cela il ne faut pas que l'extrémité d'une autre barrette appartienne au polygone.

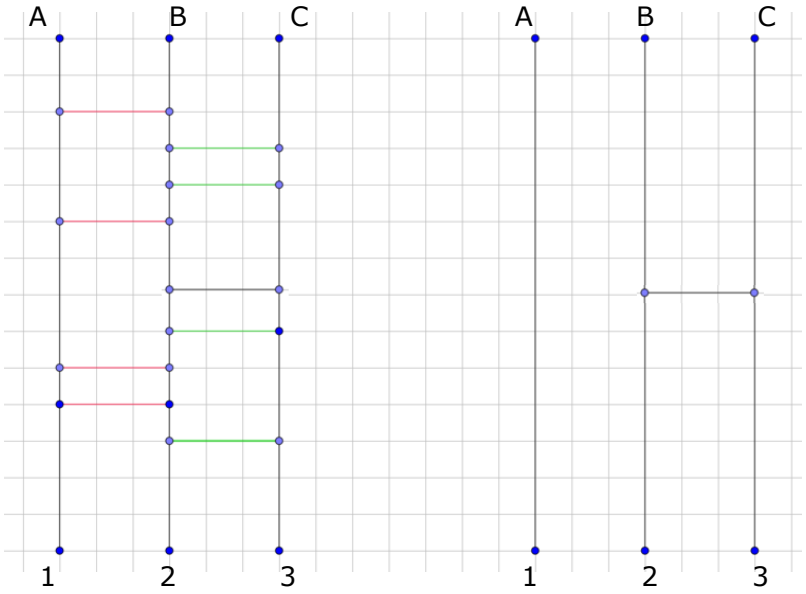
Exemple 3 : Ainsi les Amida-kuji suivants :



sont équivalents tous les deux à celui-ci :



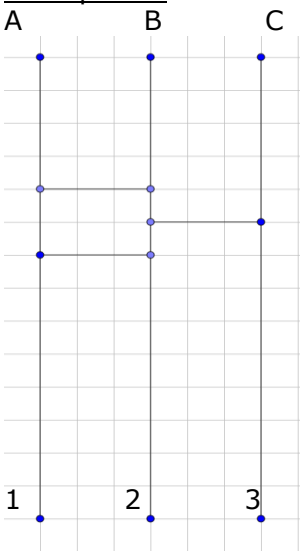
Exemple 3 : Ainsi les Amida-kuji suivants sont équivalents:



- **CAS 3**

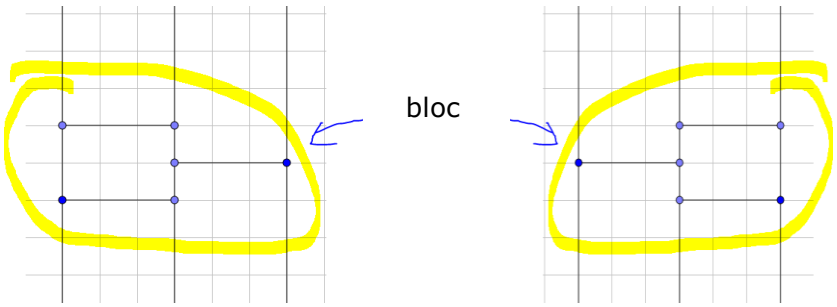
Lorsque trois barrettes sont placées sur trois traits verticaux adjacents de sorte à faire apparaître une forme comme suivant : ⊣ ou ⊢ , la répartition qui en découle associe le joueur du centre au rôle situé sous lui et les autres joueurs aux rôles opposés.

Exemple 4 :



Pour cet Amida-Kuji, la répartition est les suivante :
C va en position 1, B est inchangé (position 2) et A va en position 3.

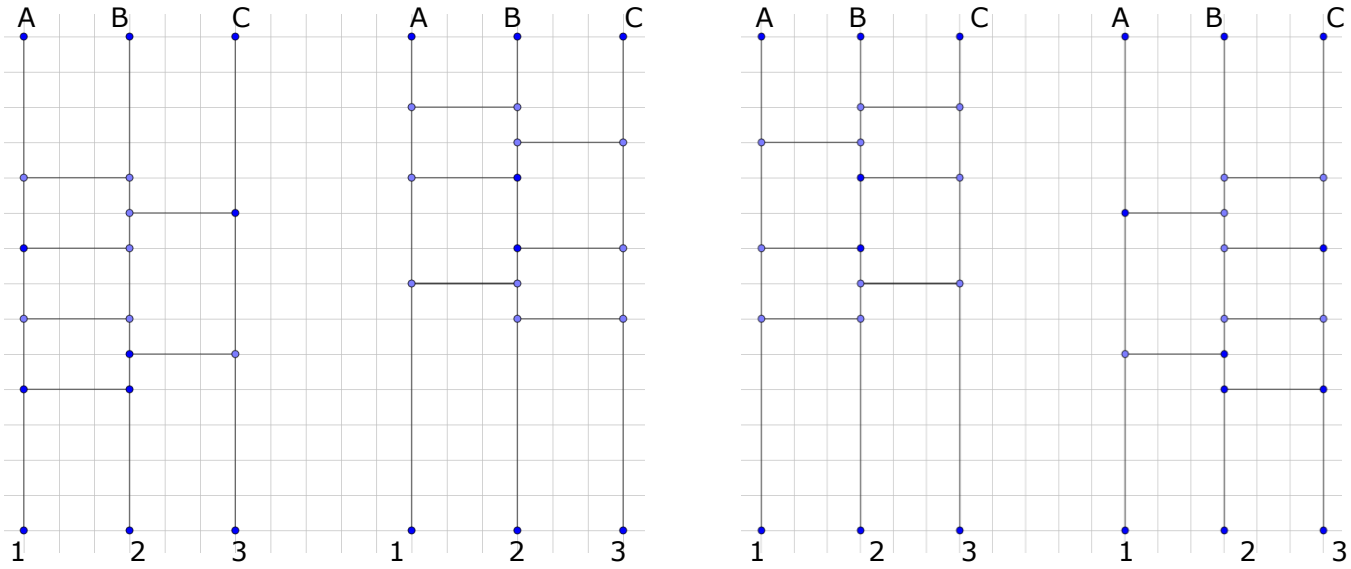
Pour la suite, nous désignons par « bloc » une forme comme décrite au début du paragraphe



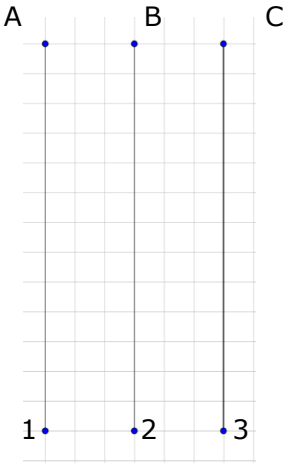
Nous avons conjecturé qu'en fonction du nombre de blocs, la répartition associée change (peu importe le sens):

-**si le nombre de blocs est pair** on peut supprimer les barrettes horizontales sans modifier la répartition associée à l'Amida-Kuji.

Exemple 5 :
Les quatre Amida-Kuji ci-dessous :

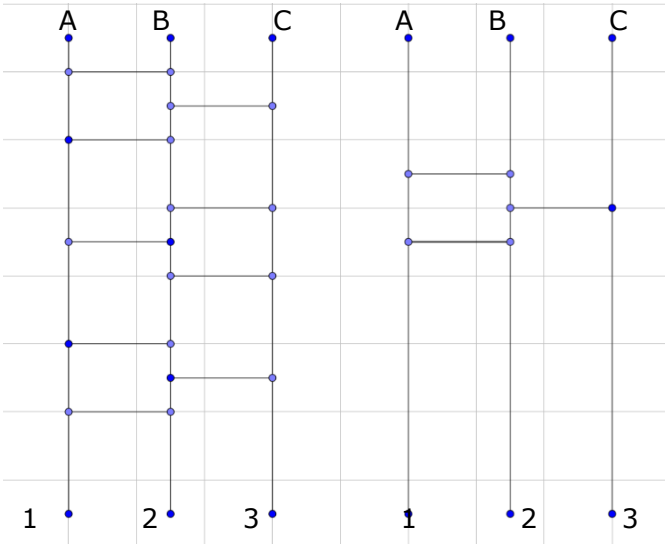


sont équivalents à celui-ci :



-**si le nombre de bloc est impair** on peut supprimer des barrettes horizontales et se ramener à un seul « bloc » sans modifier la répartition associée à l'Amida-Kuji.

Exemple 6 :
Les Amida-Kuji ci-dessous sont équivalents :



Partie 4 : Signature d'un Amida-Kuji

Lors de l'une de nos rencontres, M. Chardard nous a donné une technique pour savoir si une répartition précise peut s'obtenir à partir d'un nombre pair ou impair de barrettes.

Exemple 1:

Voici sur un exemple le principe de cette méthode. On considère la répartition suivante : $\begin{pmatrix} J1 & J2 & J3 & J4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$ et l'on souhaite savoir s'il faut un nombre pair ou impair de barrettes pour l'obtenir.

numéro du joueur : 1 2 3 4
place attribuée : 2 3 4 1

On se base d'abord sur le premier numéro de la place attribuée dans l'exemple '2'.

On doit réaliser le produit des différences entre ses voisins de droite et le nombre 2 :
On soustrait au nombre '3' le nombre '2' puis au nombre '4' le nombre '2' et au nombre '1' le nombre '2'

On recommence avec le deuxième nombre '3', le troisième '4' et le quatrième '1' et on obtient le produit suivant : P = (3-2)x(4-2)x(1-2)x(4-3)x(1-3)x(1-4)=-12

D'après M. Chardard, si le résultat trouvé est négatif, cela signifie qu'avec un nombre impair de barrettes on peut obtenir cette répartition et si le résultat trouvé est positif, cela signifie qu'on peut l'obtenir avec un nombre pair de barrettes.

Dans l'exemple -12<0 il faut donc utiliser un nombre impair de barrettes pour obtenir la répartition proposée en exemple.

Exemple 2:
On considère cette fois-ci la répartition suivante : $\begin{pmatrix} J1 & J2 & J3 & J4 \\ 3 & 1 & 2 & 4 \end{pmatrix}$ et l'on souhaite savoir s'il faut un nombre pair ou impair de barrettes pour l'obtenir.

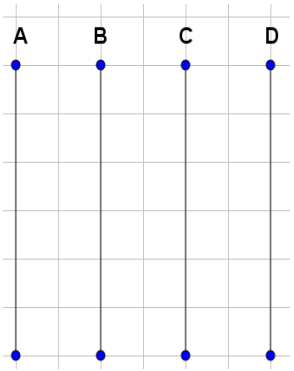
On calcule P = (1-3)x(2-3)x(4-3)x(2-1)x(4-1)x(4-2)=12.
Or 12>0, on en déduit qu'il faut un nombre pair de barrettes pour obtenir cette répartition.

Nous n'avons pas eu le temps de finir nos recherches concernant cette propriété, nous devions traduire les Amida-Kuji comme une composition de transpositions (les barrettes) et faire le lien entre leur nombre et le signe du produit P.

Partie 5 : Construire un Amida-Kuji associé à une répartition donnée

Finalement, parallèlement à la partie 4, il nous semble avoir déterminé un procédé qui permet d'associer à n'importe quelle répartition donnée un Amida-Kuji qui lui correspond. Voici le principe expliqué sur un exemple.

Exemple 1:



On considère la répartition suivante : $\begin{pmatrix} A & B & C & D \\ 3 & 1 & 2 & 4 \end{pmatrix}$

A partir de cette répartition, on veut trouver comment disposer des barrettes de façon à obtenir un Amida-Kuji conduisant à cette répartition.

On place les 4 joueurs et les traits verticaux sous les joueurs.

On observe ensuite la répartition donnée et à quel rôle sont associés les différents joueurs en partant de la droite.

- Le dernier joueur D est associé à la position 4. Dans ce cas, on n'a pas besoin d'installer de barrettes car dans la configuration actuelle (début) Le joueur D est bien associé à la position 4.
- Le joueur suivant C est associé à la position 2. On installe une barrette en partant du bas qui permet d'associer C à 2 (FIG1).
- Le joueur suivant B est associé à la position 1. En l'état actuel B est associé à 3. On installe une barrette située au-dessus des précédentes qui permet d'associer B à 1 (FIG2).
- Le joueur suivant A est associé à la position 3. En l'état actuel A est bien associé à 3. Il n'y aurait pas de modification à apporter. Néanmoins le nombre de barrettes final doit être supérieur ou égal à 4, on place donc deux barrettes « qui s'annulent » dans la configuration (FIG3).

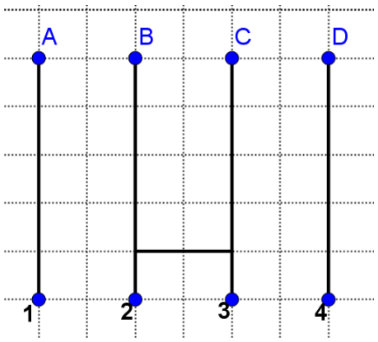


FIG1

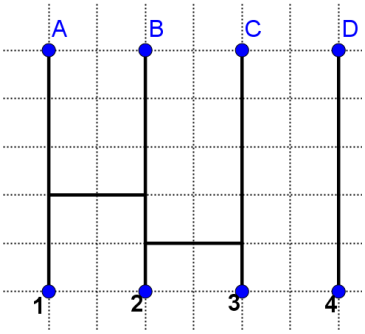


FIG 2

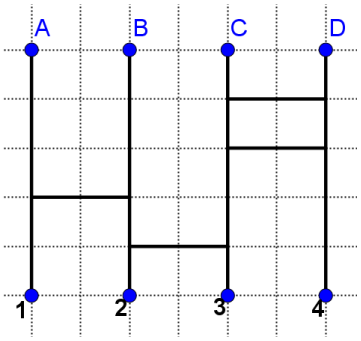
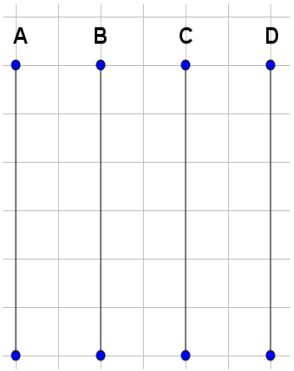


FIG 3

Exemple 2:



On considère la répartition suivante : $\begin{pmatrix} A & B & C & D \\ 4 & 1 & 3 & 2 \end{pmatrix}$

De nouveau, à partir de cette répartition, on veut trouver comment disposer des barrettes de façon à obtenir un Amida-Kuji conduisant à cette répartition.

On place les 4 joueurs et les traits verticaux sous les joueurs.

- Le dernier joueur D est associé à la position 2. On installe un ensemble de deux barrettes en bas de la construction qui permet d'associer D à 2 (FIG1).
- Le joueur suivant C est associé à la position 3. En l'état actuel C est associé à 4. On installe une barrette située au-dessus des précédentes qui permet d'associer C à 3 (FIG2).
- Le joueur suivant B est associé à la position 1. En l'état actuel B est associé à 4. On installe deux barrettes situées au-dessus des précédentes qui permettent d'associer B à 2 (FIG3).

- Le joueur suivant A est associé à la position 1. En l'état actuel A est bien associé à 1. Il n'y a pas de modification à apporter.

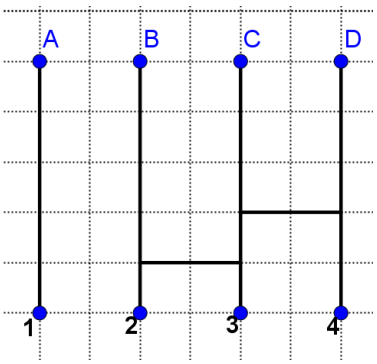


FIG1

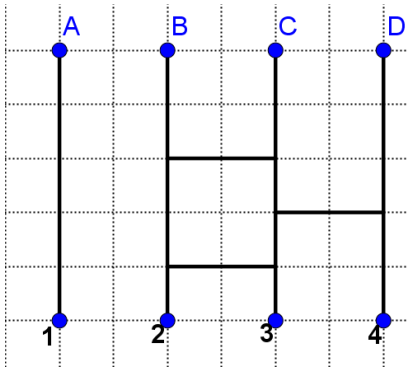


FIG 2

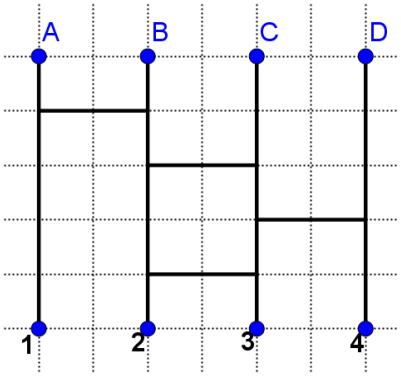


FIG 3

Il s'agit donc d'étudier successivement les associations souhaitées pour les joueurs en commençant par la droite. On installe ensuite si nécessaire un ensemble de barrettes en partant du bas et au-dessus du précédent à chaque fois, de sorte que le joueur étudié se voit associé à la bonne position.

4. Conclusion

Nous sommes satisfaits d'avoir pu mener des recherches dans le cadre de cet atelier cette année. Nous remercions nos professeurs et M. Chardard de nous avoir accompagnés et encouragés.

Nous n'avons pas terminé toutes nos recherches mais essaierons d'en faire aboutir quelques-unes de plus.

Notes d'édition

(1) Ce qui semble présenté ici comme une conjecture est en fait prouvé par les auteurs.

(2) Il faut faire attention à la cohérence de la définition d'un Amida-Kuji : chaque participant pose au moins une barrette.