

Manger un max de pizza



Énoncé

Alice et Bob se partagent une pizza, selon le « protocole de politesse de la pizza ». Bob découpe d'abord la pizza en n parts, de tailles arbitraires et non nécessairement égales. Alice commence en choisissant la part de son choix. Ensuite, à tour de rôle, chaque joueur retire une part, mais par politesse, il ne peut prendre qu'une part située à l'une des deux extrémités accessibles, c'est-à-dire adjacente à une part déjà retirée. Ainsi, à chaque coup (sauf au tout premier et au tout dernier), un joueur dispose exactement de deux parts possibles. Le but pour Alice est de manger le plus de pizza possible.

Questions :

1. Montrez qu'Alice peut manger au moins la moitié de la pizza avec un nombre pair de parts.
2. Montrez qu'Alice peut manger au moins le tiers de la pizza en suivant une stratégie où, après avoir choisi sa première part, elle prend systématiquement la part révélée par Bob.
3. Montrez que c'est le mieux qu'Alice puisse faire si, après avoir choisi sa première part, elle prend systématiquement la part révélée par Bob.
4. Montrez que Bob peut choisir une découpe de la pizza de manière à s'assurer de manger au moins $\frac{5}{9}$ de celle-ci.
5. Montrez qu'Alice peut manger au moins $\frac{4}{9}$ de la pizza.

Solution

1. Colorons les parts alternativement en rouge et vert. Comme n est pair, il y a autant de parts rouges que vertes, et l'une des deux couleurs couvre au moins la moitié de la pizza. Supposons que ce soit le rouge. Alice commence par une part rouge, et comme à chaque tour Bob ne peut choisir qu'une part de couleur opposée, une nouvelle part rouge est toujours libérée pour Alice. Ainsi, Alice peut manger toutes les parts rouges, donc au moins la moitié de la pizza.
2. Soit une pizza découpée en n parts (pour n impair, le cas pair ayant été traité à la question précédente) numérotées $1, \dots, n$, de poids (w_1, \dots, w_n) , avec $\sum_{i=1}^n w_i = 1$. Pour un ensemble S de parts, on note son poids $|S| \triangleq \sum_{i \in S} w_i$. On appelle coupe C un point de séparation entre deux parts consécutives de la pizza. Il existe donc exactement n coupes : entre 1 et 2, entre 2 et 3, ..., entre $n-1$ et n , et enfin entre n et 1. Étant données deux coupes C_i et C_j , elles délimitent deux intervalles de la pizza : l'un contenant un nombre impair de parts, noté $[C_i, C_j]_1$, et l'autre contenant un nombre pair de parts, noté $[C_i, C_j]_0$. Comme la pizza comporte un nombre impair de parts, cette distinction est toujours possible et unique. À chaque intervalle I (impair ou pair) reliant C_i et C_j on associe une coloration canonique de ses parts : en partant de C_i , on colore la première part en rouge, puis les suivantes alternativement en rouge et en vert jusqu'à C_j . Dans le cas d'un intervalle pair, l'ordre des coupes est essentiel : en effet, les parts rouges de

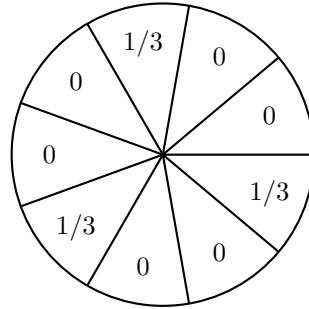
$[C_i, C_j]_0$ correspondent exactement aux parts vertes de $[C_j, C_i]_0$, et réciproquement. On note $R(I)$ (respectivement $V(I)$) l'ensemble des parts rouges (respectivement vertes) de I . Puisque la pizza comporte un nombre impair de parts, elle peut être considérée comme un intervalle impair à elle seule. En effet, il existe plusieurs intervalles impairs représentant la pizza entière, chacun de la forme $[C, C]_1$, où C est une coupe unique.

Définition. On dit que la stratégie d'Alice consiste à suivre-Bob, notée sB , si, après avoir choisi sa première part, elle prend systématiquement la part révélée par Bob.

Lemme. Alice peut manger au moins $\frac{1}{3}$ de la pizza avec une stratégie sB .

Démonstration. On choisit une coupe C minimisant $|R([C, C]_1)|$. En adoptant n'importe quelle stratégie sB , Alice finit par manger au moins $|R([C, C]_1)|$. En effet, à l'issue d'une telle stratégie, il existe une coupe C' telle que les parts mangées par Alice correspondent exactement à $R([C', C']_1)$. On suppose donc que $|R([C, C]_1)| < \frac{1}{3}$, et donc $|V([C, C]_1)| > \frac{2}{3}$. Soit p la part verte « centrale » de $V([C, C]_1)$, c'est-à-dire telle que le poids des parts vertes en partant de p (inclus) vers la coupe C dans l'une ou l'autre direction soit au moins $\frac{1}{2}|V([C, C]_1)|$. Alice commence par choisir p et applique la stratégie sB . Ainsi, Alice mange toutes les parts vertes de p jusqu'à C dans au moins une des deux directions, donc elle mange au moins $\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$ de la pizza. \square

3. On considère la pizza suivante :



Supposons qu'Alice commence par prendre une part de poids 0. Alors Bob mange immédiatement le $1/3$ adjacent. En continuant à suivre la stratégie consistant à prendre systématiquement la part révélée par Alice, Bob finit (après avoir concédé un $1/3$ à Alice) par obtenir un deuxième $1/3$. Inversement, si Alice commence par une part de poids $1/3$, alors Bob peut s'assurer de manger les deux autres $1/3$ en suivant la règle suivante :

- si un $1/3$ est disponible, Bob le prend ;
- si un 0 voisin d'un $1/3$ est disponible, Bob choisit l'autre.

4. Voici le code Python utilisé pour déterminer la pizza de Bob. Comme le dénominateur du ratio dans la question vaut 9, on peut raisonnablement supposer que le poids total de la pizza à trouver (avant normalisation) est de 9. Dans le code ci-dessous, la recherche est donc restreinte aux pizzas dont le poids total est 9, avec des parts entières. On utilise un algorithme de programmation dynamique. Sur un ordinateur portable classique, l'exécution de l'algorithme prend environ 50 secondes à trouver une solution. Enfin, on remarque qu'une démonstration (sans recours au calcul informatique) est possible, mais elle suppose de connaître préalablement la solution afin de pouvoir l'analyser.

```
python Télécharger le code

import time, functools

def canonical_rotation(pizza):
    n = len(pizza)
    if n == 0:
        return pizza
    candidates = [pizza[i:] + pizza[:i] for i in range(n)]
    return min(candidates)
```

```

def odd_pizzas(w):
    n = 1
    seen = set()
    while True:
        if n % 2 == 0:
            n += 1
            continue
        def backtrack(current, total):
            if len(current) == n:
                if total == w:
                    cano = tuple(canonical_rotation(current))
                    if cano not in seen:
                        seen.add(cano)
                        yield list(cano)
            return
        for v in range(w - total + 1):
            current.append(v)
            yield from backtrack(current, total + v)
            current.pop()
        yield from backtrack([], 0)
    n += 1

def bob_dp(pizza):
    n = len(pizza)

    def best_for_bob_after_alice_first(L):
        m = len(L)

        @functools.lru_cache(None)
        def dp(l, r, turn):
            if l > r:
                return 0
            if turn == "Bob":
                if l == r:
                    return L[l]
                return max(
                    L[l] + dp(l+1, r, "Alice"),
                    L[r] + dp(l, r-1, "Alice")
                )
            else: # Alice
                if l == r:
                    return 0
                return min(
                    dp(l+1, r, "Bob"),
                    dp(l, r-1, "Bob")
                )

        return dp(0, m-1, "Bob")

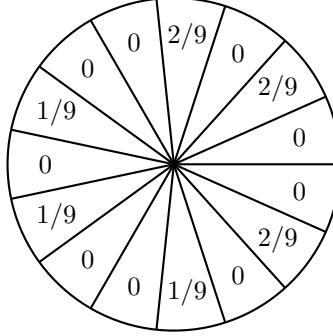
    best_bob = float("inf")
    for i in range(n):
        L = pizza[i+1:] + pizza[:i]
        score_bob = best_for_bob_after_alice_first(tuple(L))
        best_bob = min(best_bob, score_bob)

    return best_bob

start_time = time.time()
for pizza in odd_pizzas(9):
    if bob_dp(pizza) >= 5:
        print("Solution :", pizza, f"({time.time()-start_time} secondes)")
        break

```

La pizza solution est la suivante :



5. On regarde le cas n impair et on reprend les notations de la question 2.

Définition (Propriété des verts lourds).

- $[C_1, C_2]_0$ a la propriété des verts lourds si, pour tout $[C_1, C]_0 \subseteq [C_1, C_2]_0$, on a $|V([C_1, C]_0)| \geq |R([C_1, C]_0)|$.
- $[C_1, C_2]_1$ a la propriété des verts lourds si, pour tous $[C_1, C]_0, [C_2, C']_0 \subseteq [C_1, C_2]_1$, on a $|V([C_2, C']_0)| \geq |R([C_2, C']_0)|$ et $|V([C_1, C]_0)| \geq |R([C_1, C]_0)|$.

Définition (Meilleure réponse de Bob). Lorsque Alice joue la stratégie suivre-Bob après qu'un intervalle impair $[C_1, C_2]_1$ soit déjà mangé, le comportement de Bob revient à choisir une coupe C séparant $[C_1, C_2]_0$ en deux intervalles pairs $[C, C_1]_0$ et $[C, C_2]_0$. Dans ce cas, Alice obtient $R([C, C_1]_0) \cup R([C, C_2]_0)$, et Bob obtient $V([C, C_1]_0) \cup V([C, C_2]_0)$. On appelle meilleure réponse de Bob à $[C_1, C_2]_1$ toute coupe C qui minimise la quantité $|R([C, C_1]_0) \cup R([C, C_2]_0)|$, parmi toutes les coupes C telles que $[C, C_1]_0 \subseteq [C_1, C_2]_0$. Si $[C_1, C_2]_1$ ne contient qu'une seule part, on dit que C est une meilleure réponse à une part unique (mrpu). Dans ce cas, la stratégie sB d'Alice est dite associée à C .

Lemme (des verts lourds). Soit C telle que $[C, C_1]_0 \subseteq [C_1, C_2]_0$. Alors C est une meilleure réponse à $[C_1, C_2]_1$ si et seulement si $[C, C_1]_0$ et $[C, C_2]_0$ possèdent la propriété des verts lourds.

Démonstration. Une coupe C est une meilleure réponse à $[C_1, C_2]_1$ si et seulement si pour toute coupe \tilde{C} avec $[C, \tilde{C}]_0 \subseteq [C_1, C_2]_0$, on a $|R([C, C_1]_0) \cup R([C, C_2]_0)| \leq |R([\tilde{C}, C_1]_0) \cup R([\tilde{C}, C_2]_0)|$. Comme

$$R([C, \tilde{C}]_0) \subseteq R([C, C_1]_0) \cup R([C, C_2]_0), \quad V([C, \tilde{C}]_0) \subseteq R([\tilde{C}, C_1]_0) \cup R([\tilde{C}, C_2]_0), \\ (R([C, C_1]_0) \cup R([C, C_2]_0)) \setminus R([C, \tilde{C}]_0) = (R([\tilde{C}, C_1]_0) \cup R([\tilde{C}, C_2]_0)) \setminus V([C, \tilde{C}]_0),$$

on a que l'inégalité précédente est équivalente à $|V([C, \tilde{C}]_0)| \geq |R([C, \tilde{C}]_0)|$, ce qui correspond à la propriété des verts lourds pour $[C, C_1]_0$ si $[C, \tilde{C}]_0 \subseteq [C, C_1]_0$ et pour $[C, C_2]_0$ sinon. \square

Définition (Pizza facile et pizza dure). On dit qu'une pizza est facile si Alice dispose d'une stratégie sB lui garantissant au moins la moitié de la pizza. Sinon, la pizza est dite dure.

Lemme (des pizzas faciles). Si deux coupes voisines C et C' sont chacune une mrpu, alors la pizza est facile.

Démonstration. Alice peut garantir $\max(|R([C, C_1]_1)|, |R([C', C'_1]_1)|)$ en choisissant la meilleure des deux stratégies sB associées à C et C' . Soit p la part comprise entre C et C' . Comme $R([C', C'_1]_1) = V([C, C_1]_1) \cup \{p\}$, on en déduit que $R([C, C_1]_1) \cup R([C', C'_1]_1)$ recouvre toute la pizza. En particulier, l'un des deux recouvre la moitié de la pizza. \square

Théorème (des pizzas dures). Si une pizza est dure et que $C_1 \in \arg \min_C |R([C, C_1]_1)|$, alors il existe deux mrpu C_2 et C_3 qui partitionnent la pizza en trois intervalles impairs disjoints $[C_1, C_2]_1, [C_1, C_3]_1, [C_2, C_3]_1$, chacun possédant la propriété des verts lourds.

Démonstration. Pour une coupe C , on note $A(C)$ l'ensemble des parts uniques dont C est une meilleure réponse.

- On choisit $C_2 \in \arg \max_{\text{mrpu } C} |R([C, C_1])|$ de façon à maximiser $|A(C_2) \setminus A(C_1)|$.
- On choisit C_3 comme une meilleure réponse à la part $\hat{p} \in V([C_1, C_2]_1)$ la plus proche de C_2 . Les colorations canoniques de $[C_1, C_1]_1$ et $[C_2, C_2]_1$ coïncident sur $[C_1, C_2]_1$ et sont inversées sur $[C_2, C_1]_0$. Ainsi, si C_2 est une meilleure réponse à une part $p \in [C_1, C_2]_1$, alors C_1 l'est également. Autrement dit :

$$A(C_2) \setminus A(C_1) \subseteq R([C_2, C_1]_0).$$

Soit \tilde{p} la dernière part de $A(C_2) \setminus A(C_1)$ en parcourant $[C_2, C_1]_0$ depuis C_2 . Les parts \hat{p} et \tilde{p} , avec la coupe C_2 , divisent la pizza en trois intervalles, dont un se réduit à une seule part.

Affirmation. La coupe C_3 se situe dans l'intervalle entre C_2 et \tilde{p} ne contenant pas \hat{p} .

Preuve de l'affirmation. Si C_3 est dans l'intervalle entre \hat{p} et C_2 ne contenant pas \tilde{p} , alors C_3 et C_2 sont voisines, et la pizza est facile par le lemme des pizzas faciles, contradiction.

Supposons maintenant que C_3 soit dans l'intervalle entre \hat{p} et \tilde{p} ne contenant pas C_2 . Comme C_3 est une meilleure réponse à \hat{p} , on a $\hat{p} \in [C_3, C_2]_0$ (car $\hat{p} \in V([C_2, C_2]_1)$), donc $\tilde{p} \in [C_3, C_2]_1$ et on a

$$A(C_2) \setminus A(C_1) \subseteq R([C_3, C_2]_1) \subseteq R([C_3, C_3]_1).$$

Cela signifie que C_3 est une réponse possible de Bob à toutes les parts de $A(C_2) \setminus A(C_1)$. Comme C_2 est une meilleure réponse à toutes les parts de $A(C_2) \setminus A(C_1)$, on a $|R([C_2, C_2]_1)| \leq |R([C_3, C_3]_1)|$. Mais comme C_3 est aussi une mrpu, et que C_2 maximise $|R([C_2, C_2]_1)|$ sur les mrpu, on a égalité, et C_3 est une meilleure réponse à toutes les parts de $A(C_2) \setminus A(C_1)$ et aussi à $\hat{p} \notin A(C_2) \setminus A(C_1)$, contredisant la règle de choix de C_2 . \square

D'après l'affirmation, C_3 est situé dans $[C_2, C_1]_0$. Comme $\hat{p} \in R([C_3, C_3]_1)$, la coupe C_3 sépare $[C_1, C_2]_0$ en deux intervalles impairs, donnant une partition en trois intervalles impairs

$$[C_1, C_2]_1, \quad [C_1, C_3]_1, \quad [C_2, C_3]_1.$$

On considère à présent la coloration canonique des chacun de ces intervalles. On a que chaque C_i est une meilleure réponse à une part verte dans l'intervalle opposé :

- C_1 est une meilleure réponse à toutes les parts de $R([C_1, C_1]_1) \supseteq V([C_2, C_3]_1) \neq \emptyset$ car sinon la pizza est facile par le lemme des pizzas faciles,
- C_2 est une meilleure réponse à $\tilde{p} \in V([C_1, C_3]_1)$,
- C_3 est une meilleure réponse à $\hat{p} \in V([C_1, C_2]_1)$.

Par le lemme des verts lourds, chacun des intervalles vérifie la propriété des verts lourds. Pour chaque intervalle, on applique le lemme deux fois : une fois avec chacune de ses extrémités, celles-ci jouant le rôle de la coupe C , tandis que la part verte située dans l'intervalle opposé à C joue le rôle de l'intervalle impair dans le lemme. On obtient ainsi successivement les deux conditions nécessaires, ce qui établit la propriété des verts lourds pour l'intervalle impair considéré. \square

Notation (pizza partielle). On considère l'intervalle impair $[C_2, C_3]_1$ comme une pizza indépendante, obtenue en supprimant les deux autres intervalles et en recollant les coupes C_2 et C_3 . La pizza ainsi obtenue est appelée pizza partielle et la coupe issue du recollement est notée $C_{2,3}$. Comme $[C_2, C_3]_1$ vérifie la propriété des verts lourds, la coupe recollée $C_{2,3}$ de la pizza partielle appartient à $\arg \min_C |R([C, C]_1)|$ ($\arg \min$ sur les coupes de la pizza partielle).

Notation. Pour $i \in \{1, 2, 3\}$, on note R_i l'ensemble des parts rouges de l'intervalle impair opposé à C_i , et r_i leur poids total, par exemple $r_2 = |R_2| = |R([C_1, C_3]_1)|$. De même, on note V_i l'ensemble des parts vertes de l'intervalle impair opposé à C_i , et v_i leur poids total, par exemple $v_1 = |V_1| = |V([C_2, C_3]_1)|$. Supposons qu'Alice joue une stratégie sB associée à une coupe $C \in \{C_1, C_2, C_3\}$. Alors Alice obtient au moins $|R([C, C]_1)|$, ce qui peut s'exprimer en fonction des r_i et v_i comme dans le tableau suivant :

Coupe	Gain d'Alice	Gain de Bob
C_1	$v_1 + r_2 + r_3$	$r_1 + v_2 + v_3$
C_2	$r_1 + v_2 + r_3$	$v_1 + r_2 + v_3$
C_3	$r_1 + r_2 + v_3$	$v_1 + v_2 + r_3$

Par définition de C_1 (resp. C_2), le gain $v_1 + r_2 + r_3$ (resp. $r_1 + v_2 + r_3$) est le plus petit (resp. le plus grand) des trois gains possibles pour Alice. On a donc

$$v_1 + r_2 + r_3 \leq r_1 + r_2 + v_3 \leq r_1 + v_2 + r_3.$$

D'autre part, puisque la pizza est dure, chacun des gains d'Alice dans le tableau est strictement inférieur à $\frac{1}{2}$, et donc strictement inférieur au gain de Bob. En sommant chaque paire d'inégalités ainsi obtenues, on en déduit que, pour tout $i \in \{1, 2, 3\}$, on a $v_i > r_i$. On note $p_i \in V_i$ la part « centrale » telle que la somme des poids de toutes les parts vertes à partir de p_i (inclus) dans chaque direction jusqu'à rencontrer une coupe dans $\{C_1, C_2, C_3\}$ est au moins égale à $\frac{v_i}{2}$.

Définition (Stratégie sB modifiée, sBm). Pour $i \in \{1, 2, 3\}$, la i -ième stratégie sBm, notée sBm_i , est définie comme suit :

- (a) Alice commence en mangeant $p_i \in V_i$.
- (b) Tant que les coups de Bob révèlent des pièces dans V_i , Alice les prend également, c'est-à-dire qu'elle suit Bob.
- (c) Au moment où le coup de Bob révèle la première pièce rouge d'un autre des trois intervalles impairs, Alice effectue un seul coup où elle ne suit pas Bob, en prenant une pièce dans R_i .
- (d) Alice suit Bob à partir de ce moment jusqu'à la fin du jeu.

Ainsi, une stratégie sBm contient exactement un coup d'Alice où elle ne suit pas Bob. Après ce coup particulier, un intervalle impair $[C, C_j]_1 \subseteq [C_i, C_j]_1$ avec $C_i \neq C_j \in \{C_1, C_2, C_3\}$ est entièrement mangé. Alice suit alors Bob après que cet intervalle soit mangé. D'après le lemme suivant, soit C_i soit C_j est une meilleure réponse à tout intervalle impair $[C, C_j]_1 \subseteq [C_i, C_j]_1$. Dans le pire des cas, Alice obtient donc soit $r_i + v_j$, soit $v_i + r_j$ en dehors de $[C_i, C_j]_1$. La plus petite de ces deux quantités peut donc être considérée comme le gain garanti d'Alice. En combinant cela avec la définition de la pièce centrale p_i , on obtient donc les gains garantis suivants pour les stratégies sBm :

Stratégie sBm	Gain garanti pour Alice
sBm_1	$\frac{v_1}{2} + v_2 + r_3$
sBm_2	$v_1 + \frac{v_2}{2} + r_3$
sBm_3	$v_1 + r_2 + \frac{v_3}{2}$

Lemme (des sBm). Pour une pizza dure, soient $C_i, C_j \in \{C_1, C_2, C_3\}$ deux coupes distinctes et C telle que $[C, C_j]_1 \subseteq [C_i, C_j]_1$. Alors C_i ou C_j est une meilleure réponse à $[C, C_j]_1$.

Démonstration. Supposons qu'il existe une coupe $\tilde{C} \notin \{C_i, C_j\}$ qui soit une meilleure réponse à $[C, C_j]_1$. Comme on a $[\tilde{C}, C_j]_0 \subseteq [C, C_j]_0$, on obtient aussi $[\tilde{C}, C_i]_0 \subseteq [C, C_j]_0$. Ainsi, soit $[\tilde{C}, C_i]_0$, soit $[\tilde{C}, C_j]_0$ est entièrement contenu dans un des trois intervalles de la tripartition. D'après le Théorème des pizzas dures, chaque intervalle de la tripartition vérifie la propriété des verts lourds. On en déduit que, pour $k = i$ ou $k = j$, on a $|V([C_k, \tilde{C}]_0)| \geq |R([C_k, \tilde{C}]_0)|$. Comme \tilde{C} est une meilleure réponse, le lemme des verts lourds implique $|V([\tilde{C}, C_k]_0)| \geq |R([\tilde{C}, C_k]_0)|$, i.e., $|V([C_k, \tilde{C}]_0)| = |R([C_k, \tilde{C}]_0)|$. Ainsi, $|R([\tilde{C}, C]_0) \cup R([\tilde{C}, C_j]_0)| = |R([C_k, C]_0) \cup R([C_k, C_j]_0)|$. Cela montre que $C_k \in \{C_i, C_j\}$ est aussi une meilleure réponse à $[C, C_j]_1$. \square

Lemme (de la pizza partielle). Considérons une pizza avec un nombre impair de parts (pas nécessairement dure), et un état intermédiaire du jeu dans lequel l'intervalle pair $[C, C']_0$ est mangé, où $C \in \arg \min_C |R([C, C]_1)|$. Soient p et p' les parts adjacentes à C et C' , respectivement, mais non comprises dans $[C, C']_0$. Alors, si Alice choisit p' et suit ensuite Bob, elle obtient au moins autant que si elle choisit p et joue ensuite de façon optimale.

Démonstration. Supposons qu'Alice choisisse p' puis suive Bob. $C \in \arg \min_C |R([C, C]_1)|$, donc l'intervalle impair $[C, C]_1$ vérifie la propriété des verts lourds. Par le lemme des verts lourds, C est alors une meilleure réponse à l'intervalle $[C, C']_0 \cup \{p'\}$. Ainsi, dans le pire des cas, Alice obtient les parts rouges $R([C, C']_1)$ et Bob les parts vertes $V([C, C']_1)$. D'un autre côté, si Alice choisit p et joue ensuite au mieux, Bob peut forcer la même distribution en suivant simplement Alice jusqu'à ce que toute la pizza soit mangée. Par conséquent, choisir p' et suivre Bob ne peut jamais donner un résultat moins bon. \square

Définition (Stratégie bonne pour la pizza partielle). On dit qu'une stratégie pour la pizza partielle est bonne si, au moment où $C_{2,3}$ est révélé, elle prescrit déjà à Alice de suivre Bob (dans la pizza partielle) jusqu'à la fin du jeu (bien entendu, dans le contexte de la pizza entière, Alice ne peut pas appliquer directement cette instruction). Une stratégie bonne est dite connectée à la pizza entière si les conditions suivantes sont vérifiées :

- (a) Alice commence dans $[C_2, C_3]_1$ comme s'il s'agissait de la pizza partielle, et y applique la stratégie bonne tant que ni C_2 ni C_3 ne sont révélées.
- (b) Si Bob révèle C_2 ou C_3 (ce qui correspond à $C_{2,3}$ dans la pizza partielle), alors Alice joue selon la première variante du lemme de la pizza partielle plutôt que la seconde.
- (c) Chaque fois que Bob choisit une part en dehors de $[C_2, C_3]_1$, Alice suit Bob.

Lemme. Une bonne stratégie connectée à la pizza entière garantit à Alice au moins le gain garanti par cette stratégie dans la pizza partielle, plus $r_2 + v_3$.

Démonstration. Puisque la stratégie dans la pizza partielle est bonne, le lemme de la pizza partielle assure qu'à l'intérieur de $[C_2, C_3]_1$, Alice obtient au moins le gain garanti par cette stratégie pour la pizza partielle. En effet, tant que la coupe $C_{2,3}$ n'est pas révélée, la partie se déroule uniquement dans la pizza partielle et Alice suit alors sa stratégie. Lorsque $C_{2,3}$ est finalement révélé, elle applique la première variante du lemme, ce qui lui garantit un résultat au moins aussi bon que celui qu'elle aurait obtenu dans avec sa stratégie dans la pizza partielle. À l'extérieur de $[C_2, C_3]_1$, Alice suit Bob après qu'un intervalle impair $[C, C_2]_1 \subseteq [C_2, C_3]_1$ ou $[C, C_3]_1 \subseteq [C_2, C_3]_1$ a été mangé. Or, d'après le lemme des sBm, une meilleure réponse est alors donnée soit par C_2 , soit par C_3 . Ainsi, dans le pire des cas, Alice obtient soit $r_2 + v_3$, soit $v_2 + r_3$ en dehors de $[C_2, C_3]_1$. Or, on a $r_2 + v_3 \leq v_2 + r_3$. \square

Théorème. Alice peut manger au moins $4/9$ de n'importe quelle pizza.

Démonstration. Considérons d'abord la pizza partielle et distinguons deux cas.

Cas 1. La pizza partielle est facile.

Par définition, il existe une stratégie sB pour la pizza partielle assurant à Alice au moins la moitié de son poids. Chaque stratégie sB est une bonne stratégie et peut donc être connectée à la pizza entière. Considérons trois stratégies d'Alice pour la pizza entière :

- (a) La stratégie sB associée à C_2 , qui garantit au moins $r_1 + v_2 + r_3$,
- (b) La stratégie sBm₂, notée sBm₂, qui garantit au moins $v_1 + \frac{v_2}{2} + r_3$,
- (c) La stratégie sB connectée à la pizza entière, qui garantit au moins $\frac{v_1+r_1}{2} + r_2 + v_3$ (par le lemme précédent).

Le ratio de $\frac{4}{9}$ se démontre en calculant une moyenne appropriée de ces stratégies (au moins l'une de ces trois stratégies assure donc à Alice un gain moyen de $\frac{4}{9}$) :

$$\frac{3}{9}(r_1 + v_2 + r_3) + \frac{2}{9}\left(v_1 + \frac{v_2}{2} + r_3\right) + \frac{4}{9}\left(\frac{v_1+r_1}{2} + r_2 + v_3\right) \geq \frac{4}{9}(r_1 + v_1 + r_2 + v_2 + r_3 + v_3).$$

Cas 2. La pizza partielle est dure.

D'après le Théorème des pizzas dures, la pizza partielle peut être tripartitionnée par la coupe $C'_1 \triangleq C_{2,3}$ et deux autres meilleures réponses C'_2 et C'_3 en trois intervalles impairs disjoints possédant la propriété des verts lourds. On utilise la notation abrégée habituelle pour les poids totaux des parts rouges et vertes dans ces intervalles, par exemple $r'_1 = |R([C'_2, C'_3]_1)|$ et $v'_2 = |V([C'_1, C'_3]_1)|$. La stratégie sB pour la pizza partielle associée à C'_2 , ainsi que la stratégie sBm₁ pour la pizza partielle, sont des bonnes stratégies et peuvent être connectées à la pizza entière. Nous considérons quatre stratégies d'Alice pour la pizza entière :

- (a) La stratégie sB associée à C_2 , donnant au moins $r_1 + v_2 + r_3$,
- (b) La stratégie sBm₂, donnant au moins $v_1 + \frac{v_2}{2} + r_3$,
- (c) La stratégie sB associée à C'_2 connectée à la pizza entière, donnant au moins $r'_1 + v'_2 + r'_3 + r_2 + v_3$ (lemme précédent),
- (d) La stratégie sBm₁ connectée à la pizza entière, donnant au moins $\frac{v'_1}{2} + v'_2 + r'_3 + r_2 + v_3$ (lemme précédent).

On a donc

$$\begin{aligned}
& \frac{3}{9}(r_1 + v_2 + r_3) + \frac{2}{9}(v_1 + \frac{v_2}{2} + r_3) + \frac{2}{9}(r'_1 + v'_2 + r'_3 + r_2 + v_3) + \frac{2}{9}\left(\frac{v'_1}{2} + v'_2 + r'_3 + r_2 + v_3\right) \\
& \geq \frac{3}{9}(v_2 + r_3 + r_1) + \frac{2}{9}(\frac{v_2}{2} + r_3 + v_1) + \frac{2}{9}(r_2 + v_3 + v_1) + \frac{2}{9}(r_2 + v_3 + \frac{r_1}{2}) \\
& \geq \frac{4}{9}(r_1 + v_1 + r_2 + v_2 + r_3 + v_3).
\end{aligned}$$

en utilisant $v'_2 + r'_3 \geq r'_2 + v'_3$ et $r_1 = v'_1 + r'_2 + r'_3$ et $v_1 = r'_1 + v'_2 + v'_3$. \square

Notes et références

Le *problème de la pizza de Winkler* a été posé par Peter Winkler lors de la conférence *Building Bridges*, en l'honneur du 60^e anniversaire de László Lovász, à Budapest en 2008. La borne optimale connue de $\frac{4}{9}$ a été obtenue indépendamment dans deux travaux : [1] et [2]. La présente solution est basée sur ce dernier travail. Un problème voisin a été étudié dans un contexte plus général : deux joueurs se partagent un graphe connexe dont les sommets portent des poids non négatifs. À tour de rôle, ils choisissent un sommet et collectent son poids, en respectant la règle que la partie restante du graphe doit rester connexe après chaque coup. Ce jeu est connu sous le nom de *graph-grabbing game*, étudié par [3].

Sources

- [1] Josef Cibulka, Jan Kynčl, Viola Mészáros, Rudolf Stolař, and Pavel Valtr. Solution of peter winkler's pizza problem. In *International Workshop on Combinatorial Algorithms*, pages 356–367. Springer, 2009.
- [2] Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. How to eat 4/9 of a pizza. *Discrete Mathematics*, 311(16):1635–1645, 2011.
- [3] Piotr Micek and Bartosz Walczak. A graph-grabbing game. *Combinatorics, Probability and Computing*, 20(4):623–629, 2011.