מטלה 9 - שאלה 3

אלגוריתמים כלכליים
יעקב חודורקובסקי

* שאלה 3: תיכנות - תקציב פריק

כתבו פונקציה, המקבלת תקציב כלשהו, ובודקת אם הוא פריק, ואם כן - מחזירה פירוק אחד שלו.
כותרת הפונקציה:

```
def find_decomposition(
    budget: list[float],   # ההקצאה לכל נושא לפי הסדר (0, 1, ...)
    preferences:list[set[int]]   # הנושאים שכל אזרח תומך בהם
    )
```

דוגמה לקלט:

```
budget = [400, 50, 50, 0]    # נושא 0 קיבל 300, נושא 1 קיבל 100, וכו'
preferences = [ {0,1}, {0,2}, {0,3}, {1,2}, {0} ]
    # שחקן א תומך בנושאים 0,1; שחקן ב תומך בנושאים 0,2; וכו'.
```

במקרה זה התקציב פריק. דוגמה לפירוק: שחקנים א, ב, ג, ה תורמים 100 לנושא 0; שחקן ד תורם 50 לנושא 1 ו-50 לנושא 2.

הקלט:

- כסף בקופה: C.
- נושאים: m,...,1 (עמותות, ארגונים...).
- אזרחים: n,...,1.
- התועלת של אזרח i לנושא j היא: $u_{i,j}$.
- הנחה: התועלות *בינאריות* – 0 או 1.

הפלט:

- וקטור **d** המייצג תקציב: $d_1,\ldots,d_m$.
- $d_1 + \ldots + d_m = C$.
- התועלת של אזרח i מהתקציב d היא:
- $u_i(d) = \text{Sum}[j=1,\ldots,m]\ u_{i,j} * d_j$

תקציב פריק

הגדרה: תקציב $d_1,\ldots,d_m$ נקרא פָּרִיק (decomposable) אם קיימים סכומים $d_{i,j}$ לכל אזרח i ולכל נושא j כך ש:

- $\text{Sum}[i]\ d_{i,j} = d_j$; לכל נושא -
- $\text{Sum}[j]\ d_{i,j} = C/n$; לכל אזרח –
- $d_{i,j} > 0$ only if $u_{i,j} > 0$.

משמעות: אפשר לממש את התקציב באופן הבא: נותנים לכל אזרח את החלק היחסי שלו בתקציב C/n, ואומרים לו לפזר את התקציב בין הנושאים שהוא תומך בהם, לפי $d_{i,1},\ldots,d_{i,m}$.

1. **הגדרות:**
   - `budget`: רשימה של סכומים עבור חלקי התקציב.
   - `preferences`: רשימה של קבוצות - העדפות לכל שחקן.
2. **חישוב נתח הוגן לכל שחקן.**
3. **בניית המערך לאיסוף הנתונים.**
4. **לולאה על כל שחקן:**
   - חלוקת הנתח ההוגן לפי העדפות.
   - עדכון התקציב והנתח שנותר.
5. **בדיקת תקינות הפירוק.**

```python
def find_decomposition(budget: List[float], preferences: List[Set[int]]) -> Optional[List[List[float]]]:
    # *** General Case Handling ***
    def allocate_budget(budget, preferences, order):
        n = len(preferences)
        m = len(budget)
        share = sum(budget) / n
        decomposition = [[0 for _ in range(m)] for _ in range(n)]

        for i in order:
            remaining_share = share
            for j in range(m):
                if j in preferences[i] and budget[j] > 0:
                    allocation = min(budget[j], remaining_share)
                    decomposition[i][j] = allocation
                    budget[j] -= allocation
                    remaining_share -= allocation

        for j in range(m):
            if budget[j] != 0:
                return None

        return decomposition

    n = len(preferences)
    for order in permutations(range(n)):
        budget_copy = budget.copy()
        decomposition = allocate_budget(budget_copy, preferences, order)
        if decomposition is not None:
            return decomposition
```

https://colab.research.google.com/drive/1SZds_audT4yATsONSgJs3iFZZW-AsTGk?usp=sharing

```python
decomposition = find_decomposition(budget, preferences)
print(f"budget : {budget}")
print(f"preferences : {preferences}")
if decomposition is not None:
    print("Budget is decomposable.")
    for i, d in enumerate(decomposition):
        print(f"Player {i}: {d}")
else:
    print("Budget is not decomposable.")
```

דוגמאות מהפורום :

```
budget : [400, 50, 50, 0]
preferences : [{0, 1}, {0, 2}, {0, 3}, {1, 2}, {0}]
Budget is decomposable.
Player 0: [100.0, 0.0, 0, 0]
Player 1: [100.0, 0, 0.0, 0]
Player 2: [100.0, 0, 0, 0]
Player 3: [0, 50.0, 50.0, 0]
Player 4: [100.0, 0, 0, 0]
```

```
budget : [300, 200, 100]
preferences : [{0, 1}, {1, 2}, {0, 2}]
Budget is decomposable.
Player 0: [200.0, 0.0, 0]
Player 1: [0, 200.0, 0.0]
Player 2: [100.0, 0, 100.0]
```

```
budget : [100, 100]
preferences : [{0, 1}, {0}]
Budget is decomposable.
Player 0: [0, 100]
Player 1: [100, 0]
```

דוגמא לתקציב לא פריק

```
budget : [300, 150, 50]
preferences : [{0, 1}, {1, 2}, {0, 2}]
Budget is not decomposable.
```