מגיש דולב דובלון

מטלה 8 קורס אלגוריתמים כלכליים

שאלה 1: שיטת פראגמן - תיכנות

כתבו קוד פייתון המבצע שלב אחד של שיטת פראגמן לחלוקת תקציב. כותרת הפונקציה:

```
def elect_next_budget_item(
    votes: list[set[str]]],    # רשימת ההצבעות של האזרחים. לכל אזרח מצביע לקבוצה של פריטים.
    balances: list[float],     # היתרה הוירטואלית של כל אחד מהאזרחים
    costs: dict[str,float]     # העלות של כל אחד מהפריטים
    )
```

הפונקציה צריכה לכתוב למסך כמה כסף וירטואלי כל אזרח יקבל, מה הפריט הבא שייבחר, ומה היתרות החדשות. למשל:

```
After adding 0.08 to each citizen, "Park in street X" is chosen.
Citizen 1 has 1.96 remaining balance.
Citizen 2 has 0 remaining balance.
```

תשובה:

מצורף קישור לקוד בקולאב
https://colab.research.google.com/drive/1KCmLHMrDdKEcHUmkkMS9PCZoSZtvlan3?authuser=0#scrollTo=4Oa9-_VeNkWU

בנוסף בעמוד הבא מוצג הקוד:

`calculate_initial_support`

מחשבת את התמיכה ההתחלתית שקיימת כרגע בפריט/ועדה/מפלגה

`determine_funded_item`

עוברת על המועמדים והפריטים, פריט ללא תמיכה לא רלוונטי
הפונקציה מחשבת עבור הפריט את הכמות כסף שצריך להוסיף לכל משתתף על מנת "לקנות" אותו, וגם במקביל לפי הלאגוריתם צריכה למצוא את הפריט הכי מינימאלי.

`update_balances`

עדכון היתרות של המשתתפים לאחר ביצוע הרכישה

`Announce_and_finalize`

הדפסה של התוצאה לפי הוראות השאלה

```python
def calculate_initial_support(votes, costs):
    """Calculate initial support for each item."""
    item_support = {item: 0 for item in costs}
    for vote in votes:
        for item in vote:
            item_support[item] += 1
    return item_support

def determine_funded_item(votes, balances, costs, item_support):
    """Determine the item to be funded and calculate the required contribution per
supporter."""
    selected_item = None
    min_required_funding_per_supporter = float('inf')

    for item, support in item_support.items():
        if support == 0:
            continue  # Skip items with no support

        total_cost = costs[item]
        current_funding = sum(balances[i] for i, vote in enumerate(votes) if item in
vote)
        required_funding_per_supporter = (total_cost - current_funding) / support

        if required_funding_per_supporter < min_required_funding_per_supporter:
            min_required_funding_per_supporter = required_funding_per_supporter
            selected_item = item

    return selected_item, min_required_funding_per_supporter

def update_balances(votes, balances, selected_item, min_required_funding_per_supporter):
    """Update balances for citizens who supported the selected item."""
    for i, vote in enumerate(votes):
            balances[i] += min_required_funding_per_supporter  # Add
min_required_funding_per_supporter

    for i, vote in enumerate(votes):
        if selected_item in vote:
            balances[i] = 0



def announce_and_finalize(selected_item, min_required_funding_per_supporter, balances):
    """Announce the chosen item and print new balances."""
    if selected_item is None or min_required_funding_per_supporter < 0:
        print("No item can be afforded with the current balances.")
        return

    print(f'After adding {min_required_funding_per_supporter:.2f} to each citizen,
"{selected_item}" is chosen.')
    for i, balance in enumerate(balances):
        print(f"Citizen {i + 1} has {balance:.2f} remaining balance.")

def elect_next_budget_item(votes, balances, costs):
    """Main function to elect the next budget item based on votes and balances."""
    balances = balances.copy()
    item_support = calculate_initial_support(votes, costs)
```

```
    selected_item, min_required_funding_per_supporter = determine_funded_item(votes,
balances, costs, item_support)
    update_balances(votes, balances, selected_item, min_required_funding_per_supporter)
    announce_and_finalize(selected_item, min_required_funding_per_supporter, balances)


# Test the refactored function
votes_test = [{"Candidate1"} for _ in range(51)] + [{"Candidate2"} for _ in range(49)]
balances_test = [0] * 51 + [1.96] * 49
costs_test = {"Candidate1": 100, "Candidate2": 100}

elect_next_budget_item(votes_test, balances_test, costs_test)
```