โปรแกรมนี้เป็นโปรแกรมที่ใช้สำหรับ**กำหนดกิจกรรมให้พนักงานในแต่ละส่วนงาน** โดยคำนึงถึงสัดส่วนการทำงานของพนักงานแต่ละคนและภาพรวมของส่วนงาน ด้วยการใช้เทคนิค **Linear Programming** (ผ่านไลบรารี PuLP) และมี **Greedy Algorithm** เป็นวิธีสำรองในกรณีที่ Linear Programming ไม่สามารถหาคำตอบที่เหมาะสมได้ โปรแกรมนี้เขียนด้วย Python และใช้ไลบรารีหลัก เช่น pandas, PuLP, และ numpy ในการประมวลผลข้อมูลและแก้ปัญหาการมอบหมาย

ต่อไปนี้เป็นคำอธิบายการทำงานของโปรแกรมโดยแบ่งตามส่วนหลัก ๆ:

---

1. วัตถุประสงค์ของโปรแกรม

โปรแกรมนี้มีเป้าหมายเพื่อ:

- **มอบหมายกิจกรรมให้พนักงาน**ในแต่ละส่วนงาน โดยพิจารณาความเหมาะสมจากสองปัจจัยหลัก:
  - **สัดส่วนการทำงานของพนักงานแต่ละคน** (Individual Weight, ค่าเริ่มต้น 0.7) เช่น พนักงานคนนี้เคยทำกิจกรรมนี้มากน้อยแค่ไหน
  - **สัดส่วนภาพรวมของส่วนงาน** (Section Weight, ค่าเริ่มต้น 0.3) เช่น ส่วนงานนี้โดยรวมต้องการพนักงานในกิจกรรมนี้มากน้อยแค่ไหน
- **สร้างผลลัพธ์ที่เหมาะสม** โดยพยายามให้จำนวนพนักงานที่ได้รับมอบหมายในแต่ละกิจกรรมตรงกับเป้าหมายที่คำนวณจากสัดส่วนภาพรวม
- **บันทึกผลลัพธ์** ในรูปแบบไฟล์ CSV สำหรับแต่ละส่วนงานและสรุปรวมทั้งหมด

---

2. โครงสร้างของโปรแกรม

โปรแกรมประกอบด้วยฟังก์ชันหลัก ๆ ดังนี้:

2.1 `assign_activities_by_section`

- **หน้าที่**: ฟังก์ชันหลักที่จัดการการมอบหมายกิจกรรมโดยแยกตามส่วนงาน
- **ขั้นตอนการทำงาน**:
  1. อ่านข้อมูลจากไฟล์ CSV (เช่น `employee_activities.csv`) ซึ่งประกอบด้วยข้อมูลพนักงาน เช่น รหัสพนักงาน (EMPLOYEE_ID), ชื่อ (NAME), ส่วนงาน (SECTION_NAME), กิจกรรม (Activity), และสัดส่วนการทำงาน (Values)
  2. สร้างโฟลเดอร์สำหรับเก็บผลลัพธ์ถ้ายังไม่มี
  3. แยกข้อมูลตามส่วนงาน (SECTION_NAME) และประมวลผลแต่ละส่วนงานแยกกัน
  4. เรียกใช้ฟังก์ชัน `assign_single_activity_from_df` เพื่อคำนวณการมอบหมายสำหรับแต่ละส่วนงาน
  5. สร้างสรุปผลรวมทั้งหมดโดยเรียกใช้ `create_overall_summary`
- **ผลลัพธ์**: สร้างไฟล์ CSV สำหรับแต่ละส่วนงานและไฟล์สรุปรวมทั้งหมด

2.2 `assign_single_activity_from_df`

- **หน้าที่**: คำนวณการมอบหมายกิจกรรมสำหรับส่วนงานเดียว
- **ขั้นตอนการทำงาน**:
  1. **เตรียมข้อมูล**:
     - ดึงรายชื่อพนักงานและกิจกรรมจาก DataFrame
     - สร้าง dictionary เพื่อเก็บสัดส่วนการทำงานของพนักงานแต่ละคนในแต่ละกิจกรรม
     - คำนวณสัดส่วนภาพรวมของส่วนงาน (activity_ratios) โดยรวมสัดส่วนของทุกพนักงาน
  2. **กำหนดเป้าหมาย**:
     - คำนวณจำนวนพนักงานที่ควรได้รับมอบหมายในแต่ละกิจกรรมตามสัดส่วนภาพรวม
     - ปรับเป้าหมายหากจำนวนพนักงานไม่ตรงกับผลรวมของเป้าหมาย (เพิ่มหรือลดจากกิจกรรมที่มีสัดส่วนมาก/น้อย)
  3. **จัดการเงื่อนครั้งพิเศษ**:
     - ระบุพนักงานที่ทำกิจกรรมเดียว 100% และมอบหมายกิจกรรมนั้นให้ทันที
  4. **คำนวณคะแนน**:
     - คำนวณคะแนนสำหรับการมอบหมายแต่ละครั้ง โดยใช้สูตร:

$$\text{Score} = (\text{Individual Score} \times \text{Individual Weight}) + (\text{Section Score} \times \text{Section Weight}) \quad (1)$$

       - Individual Score: สัดส่วนการทำงานของพนักงานในกิจกรรมนั้น
       - Section Score: สัดส่วนภาพรวมของส่วนงานในกิจกรรมนั้น
  5. **สร้างโมเดล Linear Programming**:
     - ใช้ PuLP สร้างโมเดลที่:
       - **วัตถุประสงค์**: เพิ่มคะแนนรวมของการมอบหมายให้สูงสุด
       - **ข้อจำกัด**:

        - พนักงานแต่ละคนได้รับมอบหมายเพียง 1 กิจกรรม
        - จำนวนพนักงานในแต่ละกิจกรรมต้องเท่ากับเป้าหมาย
        - พนักงานที่มีเงื่อนไขพิเศษต้องได้รับมอบหมายกิจกรรมตามที่กำหนด
    6. **แก้ปัญหา**:
        - หากโมเดลหาคำตอบได้ (Optimal) จะรวบรวมผลลัพธ์
        - หากโมเดลไม่สามารถหาคำตอบได้ จะใช้ `use_greedy_algorithm` แทน
    7. **สร้างผลลัพธ์**:
        - สร้าง DataFrame 3 ตัว:
            - `result_df`: รายละเอียดการมอบหมาย (รหัสพนักงาน, ชื่อ, กิจกรรมที่ได้รับมอบหมาย, คะแนน)
            - `activity_df`: รายละเอียดกิจกรรมเดิมของพนักงานและกิจกรรมที่ได้รับมอบหมาย
            - `summary_df`: สรุปจำนวนพนักงานที่ได้รับมอบหมายในแต่ละกิจกรรม เทียบกับเป้าหมาย
        - บันทึกผลลัพธ์ลงไฟล์ CSV 3 ไฟล์ (assignments, details, summary)
- **ผลลัพธ์**: คืนค่า tuple ของ DataFrame 3 ตัว (`result_df`, `activity_df`, `summary_df`)

2.3 `use_greedy_algorithm`

- **หน้าที่**: ใช้วิธี Greedy Algorithm ในกรณีที่ Linear Programming ล้มเหลว
- **ขั้นตอนการทำงาน**:
    1. คำนวณคะแนนสำหรับการมอบหมายแต่ละครั้ง (เหมือนใน `assign_single_activity_from_df`)
    2. มอบหมายกิจกรรมให้พนักงานที่มีเงื่อนไขพิเศษก่อน
    3. สร้างรายการของการมอบหมายที่เป็นไปได้ทั้งหมดและเรียงตามคะแนน (จากมากไปน้อย)
    4. มอบหมายกิจกรรมให้พนักงานที่ยังไม่ได้รับมอบหมาย โดยเลือกตามคะแนนสูงสุดและจำนวนพนักงานต้องไม่เกินเป้าหมาย
    5. หากยังมีพนักงานที่ไม่ได้รับมอบหมาย ให้มอบหมายกิจกรรมที่มีคะแนนสูงสุดสำหรับพนักงานคนนั้น
    6. สร้าง DataFrame และบันทึกผลลัพธ์ลงไฟล์ CSV (เหมือนใน `assign_single_activity_from_df`)
- **ผลลัพธ์**: คืนค่า tuple ของ DataFrame 3 ตัว (`result_df`, `activity_df`, `summary_df`)

2.4 `create_overall_summary`

- **หน้าที่**: รวมผลลัพธ์จากทุกส่วนงานและสร้างไฟล์สรุปรวม
- **ขั้นตอนการทำงาน**:
    1. รวม DataFrame จากทุกส่วนงาน (`result_df`, `activity_df`, `summary_df`)
    2. บันทึกผลลัพธ์ลงไฟล์ CSV 3 ไฟล์:
        - `all_assignments.csv`: การมอบหมายทั้งหมด
        - `all_details.csv`: รายละเอียดกิจกรรมเดิมและที่ได้รับมอบหมาย
        - `all_summary.csv`: สรุปจำนวนพนักงานในแต่ละกิจกรรมของทุกส่วนงาน
- **ผลลัพธ์**: ไฟล์ CSV สรุปรวมทั้งหมด

---

3. การทำงานโดยรวม

1. **อินพุต**:
    - ไฟล์ CSV (`employee_activities.csv`) ที่มีข้อมูล:
        - COST_CENTER: รหัสศูนย์ต้นทุน
        - SECTION_NAME: ชื่อส่วนงาน
        - EMPLOYEE_ID: รหัสพนักงาน
        - NAME: ชื่อพนักงาน
        - Activity: ชื่อกิจกรรม
        - Values: สัดส่วนการทำงาน (0.0 ถึง 1.0)
    - ตัวอย่างข้อมูล:

```
COST_CENTER,SECTION_NAME,EMPLOYEE_ID,NAME,Activity,Values
CC1,Section A,001,John,Task1,0.8
CC1,Section A,001,John,Task2,0.2
CC1,Section A,002,Jane,Task1,0.3
CC1,Section A,002,Jane,Task2,0.7
```

2. **กระบวนการ**:
   - อ่านข้อมูลและแยกตามส่วนงาน
   - สำหรับแต่ละส่วนงาน:
     - คำนวณสัดส่วนภาพรวมของกิจกรรม
     - กำหนดเป้าหมายจำนวนพนักงานในแต่ละกิจกรรม
     - คำนวณคะแนนสำหรับการมอบหมายแต่ละครั้ง
     - ใช้ Linear Programming หรือ Greedy Algorithm เพื่อมอบหมายกิจกรรม
     - บันทึกผลลัพธ์ลงไฟล์ CSV
   - รวมผลลัพธ์ทั้งหมดเป็นสรุปรวม
3. **เอาต์พุต**:
   - ไฟล์ CSV สำหรับแต่ละส่วนงาน:
     - `{section_name}_assignments.csv`: รายละเอียดการมอบหมาย
     - `{section_name}_details.csv`: รายละเอียดกิจกรรมเดิมและที่ได้รับมอบหมาย
     - `{section_name}_summary.csv`: สรุปจำนวนพนักงานในแต่ละกิจกรรม
   - ไฟล์สรุปรวม:
     - `all_assignments.csv`
     - `all_details.csv`
     - `all_summary.csv`
   - ตัวอย่างผลลัพธ์ใน `assignments.csv`:

```
COST_CENTER,SECTION_NAME,EMPLOYEE_ID,NAME,ASSIGNED_ACTIVITY,ORIGINAL_VALUE,SCORE
CC1,Section A,001,John,Task1,0.8,0.85
CC1,Section A,002,Jane,Task2,0.7,0.75
```

---

4. คุณสมบัติเด่น

- **ความยืดหยุ่น**: รองรับการกำหนดน้ำหนักของสัดส่วนพนักงานและส่วนงาน (individual_weight, section_weight)
- **การจัดการข้อจำกัดพิเศษ**: พนักงานที่ทำกิจกรรมเดียว 100% จะได้รับมอบหมายกิจกรรมนั้นแน่นอน
- **วิธีสำรอง**: หาก Linear Programming ล้มเหลว จะใช้ Greedy Algorithm
- **ผลลัพธ์ที่ครอบคลุม**: สร้างรายงานทั้งในระดับส่วนงานและภาพรวม
- **การตรวจสอบความถูกต้อง**: มีการตรวจสอบและปรับเป้าหมายหากจำนวนพนักงานไม่ตรงกับเป้าหมาย

---

5. การใช้งาน

1. เตรียมไฟล์ CSV (`employee_activities.csv`) ตามโครงสร้างที่กำหนด
2. รันโปรแกรมโดยเรียกใช้:

```python
assign_activities_by_section(
    input_file="employee_activities.csv",
    output_dir="results",
    individual_weight=0.7,
    section_weight=0.3
)
```

3. ผลลัพธ์จะถูกบันทึกในโฟลเดอร์ `results`

---

6. ข้อจำกัด

- ต้องมีไฟล์ CSV ที่มีโครงสร้างตามที่กำหนด

- หากข้อมูลมีข้อผิดพลาด (เช่น สัดส่วนรวมไม่เท่ากับ 1) อาจต้องทำความสะอาดข้อมูลก่อน
- Greedy Algorithm อาจให้ผลลัพธ์ที่ไม่เหมาะสมเท่า Linear Programming ในบางกรณี
- การคำนวณขึ้นอยู่กับความถูกต้องของสัดส่วนในข้อมูลนำเข้า

---

7. ตัวอย่างการนำไปใช้

สมมติว่าบริษัทมีส่วนงาน "Section A" ที่มีพนักงาน 4 คน ทำกิจกรรม Task1 และ Task2:

- John: Task1 (80%), Task2 (20%)
- Jane: Task1 (30%), Task2 (70%)
- Bob: Task1 (100%)
- Alice: Task2 (100%)

ผลลัพธ์ที่คาดหวัง:

- Bob และ Alice จะได้รับมอบหมาย Task1 และ Task2 ตามลำดับ (เงื่อนไขพิเศษ)
- John น่าจะได้รับ Task1 และ Jane น่าจะได้รับ Task2 (ตามสัดส่วนและคะแนน)
- ไฟล์ CSV จะแสดงการมอบหมาย สัดส่วนเดิม และคะแนน

ต่อไปนี้เป็นการอธิบาย**สมการ**และ**ตัวแปรที่ใช้**ในสองวิธีที่โปรแกรมนี้ใช้ในการมอบหมายกิจกรรมให้พนักงาน ได้แก่ **Linear Programming** (ใช้ในฟังก์ชัน `assign_single_activity_from_df`) และ **Greedy Algorithm** (ใช้ในฟังก์ชัน `use_greedy_algorithm`) โดยจะแยกอธิบายสำหรับแต่ละวิธีอย่างชัดเจน

---

1. Linear Programming (ใช้ PuLP)

ตัวแปร

- **ตัวแปรตัดสินใจ (Decision Variables)**:
  - x_i, j: ตัวแปรไบนารี (0 หรือ 1) ที่ระบุว่าพนักงาน i (EMPLOYEE_ID) ได้รับมอบหมายให้ทำกิจกรรม j (Activity) หรือไม่
    - $x\_i, j = 1$: พนักงาน i ได้รับมอบหมายกิจกรรม j
    - $x\_i, j = 0$: พนักงาน i ไม่ได้รับมอบหมายกิจกรรม j
  - $i \in E$: E คือเซตของพนักงานทั้งหมดในส่วนงาน ($E = \{\text{EMPLOYEE\_ID\_1}, \text{EMPLOYEE\_ID\_2}, \ldots\}$)
  - $j \in A$: A คือเซตของกิจกรรมทั้งหมดในส่วนงาน ($A = \{\text{Activity\_1}, \text{Activity\_2}, \ldots\}$)
- **ตัวแปรข้อมูลนำเข้า (Input Parameters)**:
  - v_i, j: สัดส่วนการทำงานของพนักงาน i ในกิจกรรม j (จากคอลัมน์ `Values` ใน DataFrame, ค่าอยู่ในช่วง [0, 1])
  - r_j: สัดส่วนภาพรวมของส่วนงานสำหรับกิจกรรม j (คำนวณจาก $r\_j = \frac{\sum\_i \in E v\_i, j}{\sum\_j \in A \sum\_i \in E v\_i, j}$)
  - w_ind: น้ำหนักของสัดส่วนงานของพนักงาน (individual_weight, ค่าเริ่มต้น 0.7)
  - w_sec: น้ำหนักของสัดส่วนภาพรวมของส่วนงาน (section_weight, ค่าเริ่มต้น 0.3)
  - t_j: เป้าหมายจำนวนพนักงานที่ควรได้รับมอบหมายในกิจกรรม j (คำนวณจาก $t\_j = \text{round}(|E| \cdot r\_j)$, โดย $|E|$ คือจำนวนพนักงานทั้งหมด)
  - s_i, j: คะแนนสำหรับการมอบหมายพนักงาน i ให้ทำกิจกรรม j
    - คำนวณจาก:

$$s\_i, j = (v\_i, j \cdot w\_\text{ind}) + (r\_j \cdot w\_\text{sec}) \tag{2}$$

  - S_i: เซตของพนักงานที่มีเงื่อนไขพิเศษ (พนักงานที่ทำกิจกรรม j เดียวด้วยสัดส่วน 100%, คือ $v\_i, j = 1.0$) พร้อมกิจกรรมที่ต้องมอบหมายให้ (j)

สมการ

1. **ฟังก์ชันวัตถุประสงค์ (Objective Function)**:
   - เป้าหมายคือ**เพิ่มคะแนนรวมของการมอบหมายให้สูงสุด**:

$$\text{Maximize} \quad \sum\_i \in E \sum\_j \in A s\_i, j \cdot x\_i, j \tag{3}$$

   - อธิบาย: ต้องการให้การมอบหมายมีคะแนนรวมสูงสุด โดยคะแนน s_i, j คำนวณจากความเหมาะสมของพนักงานในกิจกรรมนั้น (จากสัดส่วนส่วนตัวและภาพรวมของส่วนงาน)
2. **ข้อจำกัด (Constraints)**:
   - **พนักงานแต่ละคนต้องได้รับมอบหมายเพียง 1 กิจกรรม**:

$$\sum\_j \in Ax\_i, j = 1 \quad \forall i \in E \tag{4}$$

- อธิบาย: ทุกพนักงานต้องได้รับมอบหมายกิจกรรมหนึ่งและเพียงหนึ่งกิจกรรมเท่านั้น

- **จำนวนพนักงานที่ได้รับมอบหมายในแต่ละกิจกรรมต้องเท่ากับเป้าหมาย**:

$$\sum\_i \in Ex\_i, j = t\_j \quad \forall j \in A \text{ where } t\_j > 0 \tag{5}$$

- อธิบาย: จำนวนพนักงานที่ได้รับมอบหมายในกิจกรรม j ต้องตรงกับเป้าหมาย t_j (เฉพาะกิจกรรมที่มีเป้าหมายมากกว่า 0)

- **เงื่อนไขพิเศษสำหรับพนักงานที่ทำกิจกรรมเดียว 100%**:

$$x\_i, j = 1 \quad \forall (i, j) \in S\_i \tag{6}$$

- อธิบาย: พนักงานที่มีเงื่อนไขพิเศษ (ทำกิจกรรม j ด้วยสัดส่วน 100%) ต้องได้รับมอบหมายกิจกรรม j นั้น

- **ตัวแปรเป็นไบนารี**:

$$x\_i, j \in \{0, 1\} \quad \forall i \in E, j \in A \tag{7}$$

- อธิบาย: x_i, j ต้องเป็น 0 หรือ 1 เท่านั้น

3. **การปรับเป้าหมาย (ถ้าจำเป็น)**:
   - หากผลรวมของ t_j ไม่เท่ากับจำนวนพนักงานทั้งหมด ($\sum\_j \in At\_j \neq |E|$) จะมีการปรับ t_j:
     - ถ้า $\sum\_j \in At\_j < |E|$: เพิ่ม t_j ของกิจกรรมที่มี r_j สูงสุด
     - ถ้า $\sum\_j \in At\_j > |E|$: ลด t_j ของกิจกรรมที่มี r_j ต่ำสุด
   - สูตรการปรับ:

$$\text{Difference} = |E| - \sum\_j \in At\_j \tag{8}$$

     - ถ้า $\text{Difference} > 0$: $t\_j\_\max += \text{Difference}$
     - ถ้า $\text{Difference} < 0$: ลด t_j จาก j ที่มี r_j ต่ำสุดจนครบ Difference

---

2. Greedy Algorithm

ตัวแปร

- **ตัวแปรข้อมูลนำเข้า (Input Parameters)**:
  - ตัวแปรส่วนใหญ่เหมือนกับ Linear Programming:
    - v_i, j: สัดส่วนการทำงานของพนักงาน i ในกิจกรรม j
    - r_j: สัดส่วนภาพรวมของส่วนงานสำหรับกิจกรรม j
    - w_ind: น้ำหนักของสัดส่วนงานของพนักงาน (0.7)
    - w_sec: น้ำหนักของสัดส่วนภาพรวมของส่วนงาน (0.3)
    - t_j: เป้าหมายจำนวนพนักงานในกิจกรรม j
    - s_i, j: คะแนนสำหรับการมอบหมายพนักงาน i ให้ทำกิจกรรม j

$$s\_i, j = (v\_i, j \cdot w\_\text{ind}) + (r\_j \cdot w\_\text{sec}) \tag{9}$$

    - S_i: เซตของพนักงานที่มีเงื่อนไขพิเศษและกิจกรรมที่ต้องมอบหมาย
  - ตัวแปรเพิ่มเติม:
    - AE: เซตของพนักงานที่ได้รับมอบหมายแล้ว ($AE \subseteq E$)
    - AA_j: จำนวนพนักงานที่ได้รับมอบหมายในกิจกรรม j (เริ่มต้นเป็น 0)
    - $Assignments$: Dictionary ที่เก็บผลการมอบหมาย ($\text{Assignments}[i] = j$ หมายถึงพนักงาน i ได้รับมอบหมายกิจกรรม j)
- **ตัวแปรชั่วคราว**:
  - Q: Priority Queue (หรือรายการที่เรียงลำดับ) ที่เก็บการมอบหมายที่เป็นไปได้ทั้งหมดในรูปแบบ:
    - $\{\text{emp\_id} : i, \text{activity} : j, \text{score} : s\_i, j\}$
    - เรียงตาม s_i, j จากมากไปน้อย

สมการ

1. **การคำนวณคะแนน**:
   - คะแนนสำหรับการมอบหมายพนักงาน i ให้ทำกิจกรรม j:

$$s\_i, j = (v\_i, j \cdot w\_\text{ind}) + (r\_j \cdot w\_\text{sec}) \tag{10}$$

- อธิบาย: คะแนนนี้ใช้ในการตัดสินใจเลือกการมอบหมายที่ดีที่สุด

2. **การกำหนดเป้าหมายจำนวนพนักงาน**:
   - จำนวนพนักงานเป้าหมายสำหรับกิจกรรม j:
   $$t\_j = \text{round}(|E| \cdot r\_j) \tag{11}$$
   - อธิบาย: คำนวณจากสัดส่วนภาพรวมของส่วนงาน คล้ายกับ Linear Programming
3. **ขั้นตอนการมอบหมาย (Algorithmic Steps, ไม่ใช่สมการโดยตรง)**:
   - **ขั้นตอน 1: มอบหมายพนักงานที่มีเงื่อนไขพิเศษ**:
   $$\text{If } (i, j) \in S\_i \text{ then Assignments}[i] = j, \quad AE = AE \cup \{i\}, \quad AA\_j += 1 \tag{12}$$
   - **ขั้นตอน 2: สร้าง Priority Queue**:
   $$Q = [\{\text{emp\_id} : i, \text{activity} : j, \text{score} : s\_i, j\} \quad \forall i \in E \setminus AE, j \in A] \tag{13}$$
     - เรียง Q ตาม s_i, j จากมากไปน้อย
   - **ขั้นตอน 3: มอบหมายพนักงานที่เหลือตามคะแนน**:
   $$\text{For each } q \in Q: \tag{14}$$
   $$\text{If } q.\text{emp\_id} \notin AE \text{ and } AA\_q.\text{activity} < t\_q.\text{activity}: \tag{15}$$
   $$\text{Assignments}[q.\text{emp\_id}] = q.\text{activity}, \quad AE = AE \cup \{q.\text{emp\_id}\}, \quad AA\_q.\text{activity} += 1 \tag{16}$$
   - **ขั้นตอน 4: มอบหมายพนักงานที่ยังไม่ได้รับมอบหมาย**:
   $$\text{For each } i \in E \setminus AE: \tag{17}$$
   $$j^* = \arg\max\_j \in As\_i, j \tag{18}$$
   $$\text{Assignments}[i] = j^*, \quad AE = AE \cup \{i\}, \quad AA\_j^* += 1 \tag{19}$$
4. **การสร้างผลลัพธ์**:
   - สำหรับแต่ละการมอบหมาย $\text{Assignments}[i] = j$:
     - คำนวณข้อมูลผลลัพธ์:
     $$\text{Result} = \{\text{EMPLOYEE\_ID} : i, \text{NAME}, \text{ASSIGNED\_ACTIVITY} : j, \text{ORIGINAL\_VALUE} : v\_i, j, \text{SCORE} : s\_$$
   - สรุปจำนวนพนักงานในแต่ละกิจกรรม:
   $$\text{ASSIGNED\_COUNT}\_j = AA\_j \tag{21}$$
   $$\text{MATCHING}\_j = (\text{ASSIGNED\_COUNT}\_j == t\_j) \tag{22}$$

เปรียบเทียบสมการและตัวแปร

ด้าน	Linear Programming	Greedy Algorithm
ตัวแปรตัดสินใจ	x_i, j (ไบนารี)	ไม่มีตัวแปรตัดสินใจแบบชัดเจน ใช้ Assignments เพื่อเก็บผลการมอบหมาย
คะแนน	$s\_i, j = (v\_i, j \cdot w\_\text{ind}) + (r\_j \cdot w\_\text{sec})$	เหมือนกัน
เป้าหมาย	Maximize $\sum\_i \in E \sum\_j \in As\_i, j \cdot x\_i, j$	เลือกการมอบหมายที่มี s_i, j สูงสุดตามลำดับ
ข้อจำกัด	- พนักงาน 1 คนต่อ 1 กิจกรรม - จำนวนพนักงานในกิจกรรม = t_j - เงื่อนไขพิเศษ	- พิจารณาเงื่อนไขพิเศษก่อน - จำกัด $AA\_j \leq t\_j$ - มอบหมายทุกพนักงาน
วิธีการแก้ปัญหา	ใช้ PuLP เพื่อหาคำตอบที่เหมาะสมที่สุด	ใช้การเรียงลำดับคะแนนและเลือกตามลำดับ (Greedy)
ผลลัพธ์	$x\_i, j = 1$ ระบุการมอบหมาย	$\text{Assignments}[i] = j$ ระบุการมอบหมาย

หมายเหตุ

- **Linear Programming** เหมาะสำหรับการหาคำตอบที่เหมาะสมที่สุด (optimal) แต่ใช้เวลาและทรัพยากรการคำนวณมากกว่า และอาจล้มเหลวหากข้อจำกัดขัดแย้งกัน
- **Greedy Algorithm** เร็วกว่าและรับประกันว่าจะมอบหมายทุกพนักงาน แต่ผลลัพธ์อาจไม่เหมาะสมที่สุด (suboptimal)
- ทั้งสองวิธีใช้ **คะแนน** s_i, j และ **เป้าหมาย** t_j ที่คำนวณจากข้อมูลเดียวกัน เพื่อให้ผลลัพธ์สอดคล้องกับความต้องการของส่วนงาน

Ref. https://grok.com/share/bGVnYWN5_92ee1668-a129-4951-bdc1-d2fc92f21fb0