TP1

Daniel Francisco Texeira Andrade - A100057

Pedro André Ferreira Malainho - A100050

Problema 1

Enunciado

Pretende-se construir um horário semanal para o plano de reuniões de projeto de uma "StartUp" de acordo com as seguintes condições:

a. Cada reunião ocupa uma sala (enumeradas **1...S**) durante um "slot" (tempo, dia). Assume-se os dias enumerado *1..D* e, em cada dia, os tempos enumerados **1..T**.

b. Cada reunião tem associado um projeto (enumerados **1..P**) e um conjunto de participantes. Os diferentes colaboradores são enumerados **1..C**.

c. Cada projeto tem associado um conjunto de colaboradores, dos quais um é o líder. Cada projeto realiza um dado número de reuniões semanais. São "inputs" do problema o conjunto de colaboradores de cada projeto, o seu líder e o númer de reuniões semanais.

d. O líder do projeto participa em todas as reuniões do seu projeto; os restantes colaboradores podem ou não participar consoante a sua disponibilidade, num mínimo ("quorum") de 50% do total de colaboradores do projeto. A disponibilidade de cada participante, incluindo o líder, é um conjunto de "slots" ("inputs" do problema).

Imports

In [2]:
```
!pip install tabulate
```

```
Requirement already satisfied: tabulate in c:\python312\lib\site-packages (0.9.0)
```

In [3]:
```
!pip install ortools
```

```
Requirement already satisfied: ortools in c:\python312\lib\site-packages (9.10.40
67)
Requirement already satisfied: absl-py>=2.0.0 in c:\python312\lib\site-packages
(from ortools) (2.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\python312\lib\site-packages (f
rom ortools) (2.1.1)
Requirement already satisfied: pandas>=2.0.0 in c:\python312\lib\site-packages (f
rom ortools) (2.2.2)
Requirement already satisfied: protobuf>=5.26.1 in c:\python312\lib\site-packages
(from ortools) (5.28.0)
Requirement already satisfied: immutabledict>=3.0.0 in c:\python312\lib\site-pack
ages (from ortools) (4.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python312\lib\site-pa
ckages (from pandas>=2.0.0->ortools) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (fr
om pandas>=2.0.0->ortools) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages
(from pandas>=2.0.0->ortools) (2024.1)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from p
ython-dateutil>=2.8.2->pandas>=2.0.0->ortools) (1.16.0)
```

In [4]:
```python
from random import randint
from tabulate import tabulate
from ortools.linear_solver import pywraplp
```

Variáveis do Programa

- `num_rooms` - Number of Meeting Rooms (NR)
- `time_slots` - List of Slots $\rightarrow$ Pair ou 2-Tuple (day, time) (SL)
- `num_projects` - Number of Projects (NP)
- `num_collaborators` - Number of Collaborators (NC)
- `project_collaborators` - Dictionary where keys are project IDs and values are lists of collaborator IDs associated with those projects (PC)
- `project_leaders` - Dictionary where keys are project IDs and values are the IDs of the leaders of those projects (PL)
- `collaborator_projects` - Dictionary where keys are collaborator IDs and values are lists of project IDs to which the collaborator is associated
- `min_weekly_meetings` - Minimum number of weekly meetings required for each project

Implementação

Generate_Slots function

Generate a list of time slots for a given number of days and time range.

This function creates a list of tuples representing time slots. Each tuple consists of a day index and an hour of the day.

Parameters:

- `days` - The number of days to generate time slots for. Must be a non-negative integer.
- `h_min` - The minimum hour for the time slots. Must be a non-negative integer.
- `h_max` - The maximum hour for the time slots. Must be greater than `h_min`.

Raises ValueError:

- If `days` is negative, `h_min` is negative, `h_max` is less than or equal to `h_min`, or `h_max` is negative.

Returns:

- A list of tuples, where each tuple represents a time slot.

In [5]:
```python
def generate_slots(days, h_min, h_max):
    if days < 0:
        raise ValueError("The number of days must be a non-negative integer.")
    if h_min < 0:
        raise ValueError("The minimum hour (h_min) must be a non-negative intege
    if h_max <= h_min:
        raise ValueError("The maximum hour (h_max) must be greater than the mini

    slots = []
    for day in range(days):
        for hour in range(h_min, h_max):
            slots.append((day,hour))

    return slots
```

Generate_Availability function

Generate a random availability dictionary for a given number of collaborators and time slots.

This function creates a dictionary where each key represents a collaborator, and each value is another dictionary that maps each time slot to a randomly assigned availability value (0 or 1). A value of 1 indicates availability and 0 indicates unavailability.

Parameters:

- `C` - The number of collaborators. Must be a non-negative integer.
- `SL` - A list of time slots, where each slot is represented as a tuple containing a day and an hour. Where `day` is a non-negative integer representing the day index and `hour` is a non-negative integer representing the hour of the day.

Raises ValueError:

- If `C` is negative or if `SL` is not a list of tuples with exactly two elements each.

Returns:

- A dictionary where each key is an integer representing a collaborator index, and each value is a dictionary mapping time slot tuples to availability values (0 or 1).

In [6]:
```python
def generate_availability(C, SL):
    if C < 0:
        raise ValueError("Number of collaborators (c) must be a non-negative int

    if not isinstance(SL, list) or not all(isinstance(slot, tuple) and len(slot)
        raise ValueError(
            "Slots (sl) must be a list of tuples, each containing exactly two el

    availability = {}

    for it in range(C):
        availability[it] = {}
        for slot in SL:
            availability[it][slot] = randint(0, 1)  # Generates 0 or 1

    return availability
```

Is_Collaborator_In_Project function

Determine the involvement of collaborators in various projects.

This function creates a dictionary that maps each collaborator to their involvement in various projects. Each key in the dictionary represents a collaborator, and the associated value is another dictionary that indicates whether the collaborator is involved in each project.

Parameters:

- `C` - The number of collaborators. Must be a non-negative integer. Collaborators are indexed from 0 to C-1.
- `PC` - A dictionary where the keys are project identifiers and the values are lists of collaborator indices involved in those projects. Project identifiers can be of any hashable type (e.g., strings, integers).

Raises ValueError:

- If `C` is negative.

Returns:

- A dictionary where each key is a collaborator index, and each value is another dictionary with project identifiers as keys and binary values (1 or 0) indicating the collaborator's involvement in that project. A value of 1 indicates involvement, and 0 indicates no involvement.

In [7]:
```
def is_collaborator_in_project(PC, C):
    if C < 0:
        raise ValueError("Number of collaborators (c) must be a non-negative int

    collaborators_in_projects = {}

    for collaborator_id in range(C):
        collaborators_in_projects[collaborator_id] = {}
        for project_id, collaborators in PC.items():
            collaborators_in_projects[collaborator_id][project_id] = 1 if collab

    return collaborators_in_projects
```

Print_Table function

This function is designed to create and display a tabular representation of certain scheduling or assignment data. It uses the tabulate module to format and print the table.

Parameters:

- `h_Min` - The minimum time slot or index for hours.
- `h_Max` - The maximum time slot or index for hours.
- `days` - Maximum of days
- `A` - The matrix representing the allocation of meetings.
- `M` - The matrix representing the allocation of collaborators to meetings.

In [8]:
```
def print_table(h_min, h_max, d_max, A, R, P, S, C):
    if h_min < 0:
        raise ValueError("Minimum hour must be a non-negative integer.")
    if h_max <= h_min:
        raise ValueError("Maximum hour must be greater than minimum hour.")
    if d_max < 0:
        raise ValueError("Number of days must be non-negative.")

    headers = ["Slots"] + [f"Day {day}" for day in range(d_max)]
    table = []

    for hour in range(h_min, h_max):
        row = [f"{hour:02d}:00"]
        for day in range(d_max):
            cell_content = ""
            for proj in range(P):
                for slot in range(S):
                    if int(A[proj][slot][(day, hour)].solution_value()) == 1:
                        proj_slot_info = f"Proj {proj} - S {slot}: "
                        col_info = " ".join(str(c) for c in range(C) if int(R[c]
                        cell_content += proj_slot_info + col_info + "\n"
            row.append(cell_content.strip() if cell_content else "")
        table.append(row)

    print(tabulate(table, headers=headers, tablefmt='rounded_grid'))
```

Restrictions

- The **matrix A** is used to allocate project meetings *p* in rooms *nr* in slot *sl*, we then have:

  $$\forall_{p<NP}\ \forall_{nr<NR}\ \forall_{sl<SL}\ A_{p,nr,sl} = 1$$

  only if there is a meeting *p* in room *nr* in slot *sl*.

- The **matrix R** is used to allocate collaborators to meetings, so we have:

  $$\forall_{c<NC}\ \forall_{p<NP}\ \forall_{nr<NR}\ \forall_{sl<Sl}\ R_{x,p,nr,sl} = 1$$

  only if a collaborator has a project meeting *p* in room *nr* in slot *sl*.

---

1. Each project has X or more weekly meetings

$$\forall_{p<NP} \sum_{nr<NR,sl\in SL} A_{p,nr,sl}\ \geq\ X$$

- With this restriction we guarantee that a project *NP* has at least X weekly meetings.

2. A collaborator can only be assigned to a meeting if it has been scheduled

$$\forall_{sl<SL}\ \forall_{nr<NR}\ \forall_{p<NP}\ \forall_{c\in NC}\ R_{c,p,nr,sl}\ \leq\ A_{p,nr,sl}$$

- With this restriction we guarantee that a collaborator *NC* will only be assigned to a meeting in a slot *SL*, in a room *NR*, in a project *NP* if the meeting has been scheduled.

3. Collaborator availability

$$\forall_{c<NC}\ \forall_{p<NP}\ \forall_{nr<NR}\ \forall_{sl\in SL}\ R_{c,p,nr,sl}\ \leq\ disp_{c,sl}$$

- With this restriction we ensure that only one meeting is allocated to one collaborator *NC*, in one project *NP*, in a room *NR*, in a slot *SL* if the collaborator *NC* is available.

4. The collaborator can only go to the meeting if they have been allocated to the project

$$\forall_{p<NP}\ \forall_{c<NC}\ \forall_{nr<NR}\ \forall_{sl\in SL}\ R_{c,p,nr,sl}\ \geq\ A_{p,nr,sl}\ \times\ collaboratorProjects_{c,p}$$

- With this restriction, we guarantee that a collaborator *NC* will only be assigned to a meeting in a project *NP*, a room *NR* or a slot *SL* if they are assigned to the corresponding project.

5. There can be no more than one meeting in a room in a slot

$$\forall_{nr<NR}\ \forall_{sl\in SL} \sum_{p<NP} A_{p,nr,sl}\ <\ 1$$

- With this restriction we guarantee that in a room *NR*, in a slot *SL*, only one meeting is allocated or not.

6. A collaborator cannot attend two meetings at the same time

$$\forall_{c<NC, sl \in SL} \sum_{nr<NR, p<NP} R_{c,p,nr,sl} \leq 1$$

- With this restriction we guarantee that for any given collaborator *C*, in a given slot *SL*, they are not allocated to more than one room *NR*, in any given project *NP*.

7. Attendance at each meeting must be above 50%

$$\forall_{p<NP, nr<NR, sl \in SL} \frac{\sum_{c<NC} R_{c,p,nr,sl}}{len(PC_p)}$$

- PC gives us the list of project collaborators P
- With this restriction we guarantee that for any project *NP*, in a room *NR*, in a slot *SL*, attendance is at least 50% compared to the number of collaborators *NC* in the project.

8. The leader has to go to all the meetings of the project he or she is leading

$$\forall_{p<NP, nr<NR, sl \in SL} R_{PL_p, p, nr, sl} = A_{p,nr,sl}$$

- With this restriction we ensure that any project leader has to be present at all project *NP* meetings in one room *NR*, in one slot *SL*.

Minimize the average number of meetings per collaborator

- To minimize the average number of meetings per participant, we minimize the sum of the meetings allocated to collaborators in the R matrix.

$$\forall_{c<NC} \; minimize \sum_{p<NP, nr<NR, sl \in SL} R_{c,p,nr,sl}$$

Maximize the number of meetings held

- To maximize the number of meetings held, we maximize the sum of all the meetings allocated in the meeting allocation matrix.

$$maximize \sum_{p<NP, nr<NR, sl \in SL} A_{p,nr,sl}$$

In [9]:
```
def schedule(NR, NP, NC, SL, PC,PL, collaborator_projects, min_weekly_meetings,

    # Meeting Allocation matrix
    A = {}
    for p in range(NP):
        A[p] = {}
        for nr in range(NR):
            A[p][nr] = {}
            for t in SL:
                A[p][nr][t] = solver.BoolVar(f'A[{p}],[{nr}],[{t}]')


    # Employee allocation matrix for a meeting
    R = {}
    for c in range(NC):
        R[c] = {}
        for p in range(NP):
            R[c][p] = {}
            for nr in range(NR):
                R[c][p][nr] = {}
                for t in SL:
                    R[c][p][nr][t] = solver.BoolVar(f'R[{c},{p},{nr},{t}]')

    # 1 - Each project has X or more weekly meetings
    for p in range(NP):
        solver.Add( sum(A[p][nr][t] for nr in range(NR) for t in SL) >= min_week

    # 2 - A collaborator can only be assigned to a meeting if it has been schedu
    for t in SL:
        for nr in range(NR):
            for p in range(NP):
                for c in range(NC):
                    solver.Add( R[c][p][nr][t] <= A[p][nr][t])

    # 3 - Collaborator availability
    for c in range(NC):
        for p in range(NP):
            for nr in range(NR):
                for t in SL:
                    solver.Add( R[c][p][nr][t] <= disp[c][t])

    # 4 - The collaborator can only go to the meeting if they have been allocate
    for p in range(NP):
        for c in range(NC):
            for nr in range(NR):
                for t in SL:
                    solver.Add( R[c][p][nr][t] <= A[p][nr][t] * collaborator_pro

    # 5 - There can be no more than one meeting in a room in a slot
    for nr in range(NR):
        for t in SL:
            solver.Add( (sum(A[p][nr][t] for p in range(NP))) <= 1)

    # 6 - A collaborator cannot attend two meetings at the same time
    for c in range(NC):
        for t in SL:
            solver.Add( (sum(R[c][p][nr][t] for nr in range(NR) for p in range(N

    # 7 - Attendance at each meeting must be above 50%
    for p in range(NP):
        for nr in range(NR):
            for t in SL:
                solver.Add ( (sum(R[c][p][nr][t] for c in range(NC)) / len(PC[p]

    # 8 - The leader has to go to all the meetings of the project he or she is l
    for p in range(NP):
        for nr in range(NR):
            for t in SL:
                solver.Add( R[PL[p]][p][nr][t] == A[p][nr][t] )
    # Minimize the average number of meetings per collaborator
    for c in range(NC):
        solver.Minimize(sum(R[c][p][nr][t] for p in range(NP) for nr in range(NR

    # Maximize the number of meetings held
    solver.Maximize( sum(A[p][nr][t] for p in range(NP) for nr in range(NR) for

    status = solver.Solve()

    if status == pywraplp.Solver.OPTIMAL:
        print_table(SL[0][1], SL[-1][1] +1, 5, A, R, NP, NR, NC)
    else:
        print("No solution found")
```

Examples

Example 1

In [10]:
```
solver = pywraplp.Solver.CreateSolver('SCIP')

# Number of rooms
NR = 2

# Number of projects
NP = 2

# Number of collaborators
NC = 4

# Dictionary mapping project IDs to lists of collaborator IDs
project_collaborators = {
    0: [0,1,3],
    1: [0,2,3]
}

# Dictionary mapping project IDs to the ID of the project leader
project_leaders = {
      0: 1,
      1: 2
}

# Generate time slots (days, hours)
SL = generate_slots(5, 9, 17)

# Generate availability for each collaborator
availability = generate_availability(NC,SL)

# Generate a dictionary mapping collaborator IDs to lists of project IDs they ar
collaborator_projects = is_collaborator_in_project(project_collaborators, NC)

# Minimum number of weekly meetings required for each project
min_weekly_meetings = 5

# Scheduling function call
schedule(NR, NP, NC, SL, project_collaborators,project_leaders, collaborator_pro
┌─────────┬───────────────────┬───────────────────┬───────────────────┬─────────
─────────┬───────────────────┐
│ Slots   │ Day 0             │ Day 1             │ Day 2             │ Day 3
│ Day 4             │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 09:00   │                   │ Proj 0 - S 1: 1 3 │ Proj 0 - S 1: 1 3 │ Proj 1 -
S 0: 2 3 │ Proj 0 - S 0: 0 1 │
│         │                   │ Proj 1 - S 0: 0 2 │                   │
│                   │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 10:00   │ Proj 1 - S 0: 2 3 │ Proj 0 - S 1: 1 3 │ Proj 0 - S 0: 1 3 │
│ Proj 1 - S 0: 2 3 │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 11:00   │                   │ Proj 0 - S 0: 1 3 │ Proj 0 - S 0: 1 3 │
│                   │
│         │                   │                   │ Proj 1 - S 1: 0 2 │
│                   │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 12:00   │                   │ Proj 0 - S 0: 0 1 │                   │
│ Proj 0 - S 0: 0 1 │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 13:00   │                   │ Proj 1 - S 0: 0 2 │                   │
│ Proj 1 - S 0: 2 3 │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 14:00   │                   │ Proj 1 - S 0: 2 3 │ Proj 1 - S 0: 2 3 │ Proj 0 -
S 0: 0 1 │ Proj 0 - S 1: 1 3 │
│         │                   │                   │                   │
│ Proj 1 - S 0: 0 2 │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 15:00   │                   │ Proj 1 - S 0: 0 2 │                   │
│                   │
├─────────┼───────────────────┼───────────────────┼───────────────────┼─────────
─────────┼───────────────────┤
│ 16:00   │ Proj 0 - S 0: 0 1 │ Proj 0 - S 0: 0 1 │ Proj 1 - S 0: 0 2 │ Proj 0 -
S 0: 1 3 │                   │
└─────────┴───────────────────┴───────────────────┴───────────────────┴─────────
─────────┴───────────────────┘
```

Example 2

In [11]:
```python
solver = pywraplp.Solver.CreateSolver('SCIP')

# Number of rooms
NR = 4

# Number of projects
NP = 5

# Number of collaborators
NC = 10

# Dictionary mapping project IDs to lists of collaborator IDs
```

```
project_collaborators = {
    0: [0,1,2,3,6,8],
    1: [0,1,4,5,9],
    2: [2,3,7,8],
    3: [4,5,6,7,9],
    4: [1,3,4,5]
}

# Dictionary mapping project IDs to the ID of the project leader
project_leaders = {
      0: 1,
      1: 4,
      2: 2,
      3: 5,
      4: 3
}

# Generate time slots (days, hours)
SL = generate_slots(5, 9, 19)

# Generate availability for each collaborator
availability = generate_availability(NC,SL)

# Generate a dictionary mapping collaborator IDs to lists of project IDs they ar
collaborator_projects = is_collaborator_in_project(project_collaborators, NC)

# Minimum number of weekly meetings required for each project
min_weekly_meetings = 10

# Scheduling function call
schedule(NR, NP, NC, SL, project_collaborators,project_leaders, collaborator_pro
╭─────────┬─────────────────────┬─────────────────────┬─────────────────────┬────
─────────────────┬─────────────────────╮
│ Slots   │ Day 0               │ Day 1               │ Day 2               │ Day
3               │ Day 4               │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 09:00   │ Proj 3 - S 3: 4 5 7 │                     │ Proj 2 - S 1: 2 7   │
│ Proj 0 - S 2: 1 2 6 │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 10:00   │ Proj 2 - S 0: 2 8   │ Proj 1 - S 1: 0 1 4 │ Proj 2 - S 0: 2 3   │ Pro
j 0 - S 3: 0 1 3 │ Proj 1 - S 3: 0 4 9 │
│         │ Proj 4 - S 1: 1 3   │                     │ Proj 3 - S 1: 5 6 9 │
│ Proj 2 - S 1: 2 3   │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 11:00   │ Proj 2 - S 2: 2 8   │ Proj 2 - S 2: 2 3   │ Proj 0 - S 0: 1 6 8 │ Pro
j 1 - S 0: 1 4 5 │ Proj 2 - S 0: 2 3   │
│         │ Proj 4 - S 3: 1 3   │                     │ Proj 1 - S 1: 0 4 5 │ Pro
j 2 - S 2: 2 7   │                     │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 12:00   │ Proj 3 - S 0: 4 5 6 │ Proj 0 - S 1: 1 3 6 │ Proj 0 - S 0: 0 1 3 │
│                     │
│         │                     │                     │ Proj 2 - S 1: 2 7   │
│                     │
│         │                     │                     │ Proj 3 - S 2: 5 6 9 │
│                     │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 13:00   │ Proj 2 - S 3: 2 3 7 │                     │ Proj 3 - S 1: 5 6 9 │ Pro
j 2 - S 2: 2 8   │ Proj 2 - S 1: 2 8   │
│         │ Proj 3 - S 0: 5 6 9 │                     │                     │ Pro
j 3 - S 1: 4 5 7 │                     │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 14:00   │ Proj 4 - S 0: 3 5   │ Proj 1 - S 1: 1 4 5 │                     │ Pro
j 4 - S 0: 3 4   │ Proj 4 - S 0: 1 3   │
│         │                     │ Proj 2 - S 0: 2 3   │                     │
│                     │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 15:00   │                     │ Proj 0 - S 1: 0 1 8 │                     │
│ Proj 2 - S 0: 2 7   │
│         │                     │                     │                     │
│ Proj 4 - S 3: 1 3   │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 16:00   │ Proj 3 - S 3: 4 5 6 │ Proj 0 - S 0: 0 1 8 │ Proj 1 - S 2: 0 4 9 │ Pro
j 4 - S 1: 3 4   │ Proj 2 - S 3: 2 8   │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
│ 17:00   │ Proj 1 - S 0: 1 4 9 │ Proj 2 - S 0: 2 3   │ Proj 1 - S 2: 0 4 9 │ Pro
j 4 - S 0: 3 5   │ Proj 0 - S 3: 0 1 8 │
│         │ Proj 2 - S 2: 2 8   │ Proj 3 - S 3: 5 6 7 │ Proj 2 - S 3: 2 7   │
│                     │
│         │                     │                     │ Proj 4 - S 1: 3 5   │
│                     │
├─────────┼─────────────────────┼─────────────────────┼─────────────────────┼────
─────────────────┼─────────────────────┤
```

```
│ 18:00   │ Proj 4 - S 3: 1 3   │ Proj 1 - S 3: 0 4 5 │ Proj 0 - S 2: 1 6 8 │ Pro
j 1 - S 1: 0 1 4 │ Proj 0 - S 0: 1 2 6 │
│         │                     │                     │ Proj 1 - S 1: 0 4 9 │ Pro
j 2 - S 0: 2 8   │                     │
│         │                     │                     │                     │ Pro
j 3 - S 3: 5 6 7 │                     │
└─────────┴─────────────────────┴─────────────────────┴─────────────────────┴────
─────────────────┴─────────────────────┘
```

Example 3

In [12]:

```python
# Number of rooms
NR = 4

# Number of projects
NP = 7

# Number of collaborators
NC = 20

# Dictionary mapping project IDs to lists of collaborator IDs
project_collaborators = {
    0: [0,1,2,3,6,8,15,16],
    1: [0,1,4,5,9,15,17],
    2: [2,3,7,8,10,11,17,16],
    3: [4,5,6,7,9,18,19],
    4: [1,3,4,5,12,13,14],
    5: [1,2,8,12,13],
    6: [0,5,9,10,12,14]
}

# Dictionary mapping project IDs to the ID of the project leader
project_leaders = {
      0: 1,
      1: 4,
      2: 2,
      3: 5,
      4: 3,
      5: 12,
      6: 9
}

# Generate time slots (days, hours)
SL = generate_slots(5, 9, 19)

# Generate availability for each collaborator
availability = generate_availability(NC,SL)

# Generate a dictionary mapping collaborator IDs to lists of project IDs they ar
collaborator_projects = is_collaborator_in_project(project_collaborators, NC)

# Minimum number of weekly meetings required for each project
min_weekly_meetings = 40

# Scheduling function call
schedule(NR, NP, NC, SL, project_collaborators,project_leaders, collaborator_pro
```

```
No solution found
```