Inicialização das turmas: a primeira será a partir das disciplinas e ordem definidas no enunciado. As outras, são vazias, para serem preenchidas a partir da execução do algoritmo.
Todas elas, são armazenadas numa lista de turmas, para serem verificadas na restrição de choque de horários.

```
1   aulas = ["C1", "IP", "GA", "IP", "IP",
2            "IC", "C1", "LM", "LM", "GA"]
3
4   turma_a = [aulas[:5], aulas[5:]]
5   turma_b = [["" for _ in range(5)] for _ in range(2)]
6   turma_c = [["" for _ in range(5)] for _ in range(2)]
7
8   turmas = [turma_a, turma_b, turma_c]
9
```

Backtracking: percorre a matriz (turma) de cima para baixo, da esquerda para a direita, ou seja, o primeiro horário do primeiro dia, o segundo horário do primeiro dia, o primeiro horário do segundo dia e assim por diante. Dessa forma, há o controle das variáveis para não haver estouro de índices.

A inserção ocorre a partir de uma lista que é uma cópia da lista de aulas da primeira turma. Essa lista é manipulada para garantir que as turmas tenham as mesmas disciplinas e na quantidade original de cada uma.

Se a disciplina respeitar as restrições, poderá ser inserida. Assim, ela será removida da lista de aulas disponíveis e iremos para o próximo horário daquela turma até que esteja com a grade completa.

Caso contrário, a posição voltará a ficar livre e a aula retornará para a lista de aulas disponíveis (backtrack).

```
34  def preencher_turma(turma, horario, dia):
35      if horario >= 2:  # Todos os horários foram preenchidos
36          horario = 0
37          dia += 1
38
39      if dia >= 5:  # Todos os dias foram preenchidos
40          return True
41
42      for aula in aulas_disponiveis:
43          if not ha_choque(horario, dia, aula, turmas) and not aula_repetida(dia, aula, turma):
44              turma[horario][dia] = aula
45              aulas_disponiveis.remove(aula)
46              if preencher_turma(turma, horario + 1, dia):
47                  return True
48
49              # Backtrack
50              turma[horario][dia] = ""
51              aulas_disponiveis.append(aula)
52
53      return False
```

Com relação às restrições, são implementadas em duas funções.

A primeira verifica se a aula a ser inserida não está presente em outra turma no mesmo horário.

Já a segunda, verifica se a aula não está sendo repetida no mesmo dia naquela turma.

```python
# Verifica se há choque de horário em outras turmas
def ha_choque(horario, dia, aula, turmas):
    for turma in turmas:
        if turma[horario][dia] == aula:
            return True

    return False

# Verifica se a disciplina tem duas aulas no mesmo dia e na mesma turma
def aula_repetida(dia, aula, turma):
    '''
    Basta verificar se a aula é igual ao primeiro horário.
    Pois, nas novas turmas, a primeira iteração comparará a aula com string vazia.
    E na segunda, compara duas aulas preenchidas
    '''
    return turma[0][dia] == aula
```

Em seguida, basta inserir as aulas das outras turmas e imprimir o resultado:

```python
# Preenchendo turma_b e turma_c
aulas_disponiveis = aulas.copy() # Estrutura usada para manter controlar aulas inseridas
preencher_turma(turma_b, 0, 0)
aulas_disponiveis = aulas.copy() # Estrutura usada para manter controlar aulas inseridas
preencher_turma(turma_c, 0, 0)

imprimir_grades()
```

```python
def imprimir_grades():
    for i, turma in enumerate(turmas):
        print(f"Turma {chr(ord('A') + i)}:")
        for horario in turma:
            print(horario)
        print()
```

```
Turma A:
['C1', 'IP', 'GA', 'IP', 'IP']
['IC', 'C1', 'LM', 'LM', 'GA']

Turma B:
['IP', 'GA', 'IP', 'LM', 'LM']
['C1', 'IP', 'IC', 'GA', 'C1']

Turma C:
['GA', 'C1', 'LM', 'C1', 'IC']
['IP', 'GA', 'IP', 'IP', 'LM']
```