B.7:

Schreiben Sie eine PYTHON-Funktion `to_matrix(G)`, die für einen Graphen G in Standardrepräsentation dessen Adjazenzmatrix A in $O(m^2)$ zurückliefert. Schreiben Sie ebenso eine Funktionen `from_matrix(A)`, die für eine Adjazenzmatrix A den zugehörigen Graphen G in Standardrepräsentation in $O(m^2)$ zurückliefert.

Lösungshinweis: Repräsentieren Sie A als Liste von Listen mit Einträgen aus $\{0, 1\}$.

```python
# Liefert für Graphen G in Standardrepräsentation dessen Adjazenzmatrix A zurück
def to_matrix(G):
    # Anzahl der Knoten
    m = len(G)
    # m x m - Matrix gefüllt mit Nullen
    A = [[0 for _ in range(m)] for _ in range(m)]

    # Setze A[i][j] = 1, falls eine Kante zwischen i und j existiert
    for i in range(m):
        for x in G[i]:
            A[i][x] = 1
    return A
```

```python
# Liefert für eine Adjazenzmatrix A den zugehörigen Graphen G in Standardrepräsentation
def from_matrix(A):
    # Anzahl der Knoten
    m = len(A)

    # Alle Mengen leer anlegen
    G = [set() for _ in range(m)]

    # Füge {j} zur Menge G[i] hinzu, falls eine Kante zwischen i und j existiert
    # (also wenn A[i][j] gleich 1)
    for i in range(m):
        for j in range(m):
            if(A[i][j] == 1):
                G[i] |= {j}

    return G
```

```python
# Test mit Beispiel-Graphen aus Vorlesung
def define_G1():
    m = 6
    G = [set() for _ in range(m)]   # alle Mengen leer anlegen
    G[0] |= {1,2}
    G[1] |= {0,2,5}
    G[2] |= {0,1,3,5}
    G[3] |= {2,5}
    # G[4] ist leer
    G[5] |= {1,2,3}
    return G

G = define_G1()
A = to_matrix(G)

print(G == from_matrix(A))  # -> True

```