B.11:

Sei $G = (V, E)$ ein Graph. Mit Hilfe der Breitensuche können wir die Distanz aller Knoten zum Startknoten s und jeweils *einen* kürzesten Pfad (bzgl. Kantenanzahl) über den BFS-Baum bestimmen. Für $u \in V$ sei $S(u)$ die *Anzahl* der kürzesten Pfade von s zu Knoten u.

1. Überlegen Sie, wie $S(u)$ induktiv über $j \geq 0$ bestimmt werden kann, und zwar unter Verwendung des Ergebnisses einer vorhergehenden Breitensuche $L_0, \ldots, L_j, \ldots$.

(IA) $j = 0$

Es gilt $L_j = L_0 =_{def} \{s\}$. Die Breitensuche liefert nur das Startelement, was bedeutet, dass kein Pfad von s aus im Graphen existiert. Somit ist kein $u \in V$ von s aus erreichbar und es gilt für alle $u \in V$:

$$S(u) = 0$$

(IS) $j \to j + 1$

Sei U_{j+1} die Menge aller Knoten, die in Schicht $j + 1$ des BFS-Baumes liegen. All diese Knoten sind über eine Kante mit mindestens einem Knoten aus Schicht j des BFS-Baumes verbunden.

Da nach (IV) $S(u)$ für die Knoten aller Schichten bis Schicht j bereits definiert ist, definieren wir $S(u_{j+1})$ mit $u_{j+1} \in U_{j+1}$ folgendermaßen:

1. Bestimme alle Knoten u_j aus Schicht j des BFS-Baumes die direkt über eine Kante mit $S(u_{j+1})$ verbunden sind
2. Addiere die Anzahl der kürzesten Pfade dieser Knoten $S(u_j)$

2. Geben Sie eine nicht-rekursive Funktion `bfs_numpaths(G,s)` an, die die Zahlen $S(u)$ für alle $u \in V$ bestimmt und als Liste `S` zurückliefert. Führen Sie eine Laufzeitanalyse durch. Können Sie $O(m + k)$ erreichen?

```python
def bfs_numpaths(G,s):
    m = len(G)
    distance = [None]*m
    S_u = [None]*m

    result = bfs(G,s)       # Ergebnis der Breitensuche
    L_j = result[0]         # Liste L_j der Schichten
    for v in range(m):                          # Bestimme S_u für alle Knoten v
        j = 0
        for x in range(len(L_j)):               # Finde Nummer j der Schicht in der enthalten ist
            if v in L_j[x]:
                j = x
                break

        if(j != 0):             # Schicht 0 wird nicht betrachtet
            distance[v] = j
            count = 0           # Anzahl der Pfade mit kürzester Länge (j)

            nodes_to_check = set({v})               # Hilfsvariablen
            parent_nodes_upper_layer = set()

                                # Durchlaufe alle Schichten ab j nach oben um zu überprüfen, von wie vielen Knoten
            while j > 0:    # aus der momentan betrachtete Knoten v erreichbar ist -> entspricht count
                for node in L_j[j - 1]:
                    for n in G[node]:
                        if n in nodes_to_check:
                            parent_nodes_upper_layer.add(n)
                            count += 1

                nodes_to_check = parent_nodes_upper_layer   # alle Knoten von denen v aus erreichbar sind nach oben verfolgen
                parent_nodes_upper_layer = set()
                j -= 1

            S_u[v] = count

    print("DISTANCE: ", distance)
    return S_u
```