Pseudocode

Input: A graph G and a vertex v of G.

Output: The closest vertex to v satisfying some conditions, or null if no such vertex exists.

```
from collections import deque

def BFS(G, v, target):
    queue = deque()
    queue.append(v)
    marked = set()

    while queue:
        w = queue.popleft()
        if w == target:
            return w

        for e in G.adjacentEdges(w):
            x = G.adjacentVertex(w, e)
            if x not in marked:
                marked.add(x)
                queue.append(x)

    return None

# Example usage
class Graph:
    def adjacentEdges(self, vertex):
        # Return a list of adjacent edges for the given vertex

    def adjacentVertex(self, vertex, edge):
```

```
        # Return the adjacent vertex for the given vertex and edge

# Create a graph
G = Graph()

# Choose a starting vertex and a target vertex
start_vertex = 0
target_vertex = 5

# Call BFS to search for the target vertex starting from the start vertex
result = BFS(G, start_vertex, target_vertex)

if result is not None:
    print("Target vertex found:", result)
else:
    print("Target vertex not found.")
```