```
from collections import deque

def solve(N, edges):
    adj = [[] for _ in range(N + 1)]
    rev_adj = [[] for _ in range(N + 1)]
    for u, v, w in edges:
        adj[u].append((v, w))
        rev_adj[v].append((u, w))

    # step 1: reverse BFS to find valid pathes
    can_reach = [False] * (N + 1)
    can_reach[N] = True
    q = deque([N])
    while q:
        curr = q.popleft()
        for prev, w in rev_adj[curr]:
            if not can_reach[prev]:
                can_reach[prev] = True
                q.append(prev)

    if not can_reach[1]:
        return "No Path"

    # step 2: zero BFS to find initial zeros
    visited = [False] * (N + 1)
    visited[1] = True
    q = deque([1])
    layer = [] # starting nodes

    while q:
        u = q.popleft()
        if can_reach[u]:
            layer.append(u)
        for v, w in adj[u]:
            if w == 0 and not visited[v]:
                visited[v] = True
                q.append(v)

    if visited[N]:
        return "0"
```

```
    # step 3: greedy BFS
    ans = []
    while layer and not visited[N]:
        min_digit = 10

        for u in layer:
            for v, w in adj[u]:
                if can_reach[v] and not visited[v]:
                    min_digit = min(min_digit, w)

        ans.append(str(min_digit))

        # create next layer with min_digit edges
        next_layer = []
        for u in layer:
            for v, w in adj[u]:
                if w == min_digit and can_reach[v] and not
visited[v]:
                    visited[v] = True
                    next_layer.append(v)

        layer = next_layer

    return "".join(ans)

N1 = 4
edges1 = [
    (1, 2, 2),
    (1, 3, 2),
    (2, 4, 5),
    (3, 4, 3)
]
print(solve(N1, edges1))
print()

N2 = 4
edges2 = [
    (1, 2, 0),
    (2, 3, 0),
    (3, 4, 4),
    (1, 4, 5)
```

```
]
print(solve(N2, edges2))
print()

N3 = 10
edges3 = [
    (1, 2, 0),
    (1, 5, 0),
    (5, 6, 0),
    (2, 8, 0),
    (8, 9, 1),
    (2, 4, 3),
    (4, 10, 5),
    (6, 7, 3),
    (7, 10, 1)
]
print(solve(N3, edges3))
print()
```