```
from typing import List, Set

class ActivityM2:
    @staticmethod
    def find_edges(grid: List[List[str]]) -> List[Set[int]]:
        if not grid or not grid[0]:
            return []

        m, n = len(grid), len(grid[0])
        visited = set()
        edges = []

        def dfs(i: int, j: int, edge: Set[int]) -> None:
            if (i, j) in visited:
                return
            visited.add((i, j))
            directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
            is_edge = False
            for dx, dy in directions:
                x, y = i + dx, j + dy
                if x < 0 or x >= m or y < 0 or y >= n or grid[x][y] == '0':
                    is_edge = True
                elif grid[x][y] == '1':
                    dfs(x, y, edge)
            if is_edge:
                edge.add(i * n + j)

        for i in range(m):
            for j in range(n):
                if grid[i][j] == '1' and (i, j) not in visited:
                    edge = set()
                    dfs(i, j, edge)
                    if edge:
                        edges.append(edge)

        return edges

if __name__ == "__main__":
    example = [
        ['1', '1', '0', '0', '0'],
        ['1', '1', '0', '0', '0'],
        ['0', '0', '1', '0', '0'],
        ['0', '0', '0', '1', '1']
    ]
```

```
edges = ActivityM2.find_edges(example)
print(edges)
```