Code Jam 2009 - Qualification Round

Analysis: Watersheds

For each cell, we need to determine its eventual sink. Then, to each group of cells that share the same sink, we need to assign a unique label.

The inputs to this problem are small enough for a simple brute-force simulation algorithm. Start with a cell and trace the path that water would take by applying the water flow rules repeatedly. Here is one possible solution in Python.

```
import sys

def ReadInts():
  return list(map(int, sys.stdin.readline().strip().split(" ")))

def Cross(a, b):
  for i in a:
    for j in b:
      yield (i, j)

def Neighbours(ui, uj, m, n):
  if ui - 1 >= 0: yield (ui - 1, uj)
  if uj - 1 >= 0: yield (ui, uj - 1)
  if uj + 1 < n: yield (ui, uj + 1)
  if ui + 1 < m: yield (ui + 1, uj)

N = ReadInts()[0]
for prob in xrange(1, N + 1):
  # Read the map
  (m, n) = ReadInts()
  maze = [ReadInts() for _ in xrange(m)]
  answer = [["" for _ in xrange(n)] for _ in xrange(m)]

  # The map from sinks to labels.
  label = {}
  next_label = 'a'

  # Brute force each cell.
  for (ui, uj) in Cross(xrange(m), xrange(n)):
    (i, j) = (-1, -1)
    (nexti, nextj) = (ui, uj)
    while (i, j) != (nexti, nextj):
      (i, j) = (nexti, nextj)
      for (vi, vj) in Neighbours(i, j, m, n):
        if maze[vi][vj] < maze[nexti][nextj]:
          (nexti, nextj) = (vi, vj)

    # Cell (ui, uj) drains to (i, j).
    if (i, j) not in label:
      label[(i, j)] = next_label
      next_label = chr(ord(next_label) + 1)
    answer[ui][uj] = label[(i, j)]

  # Output the labels.
```

```
print "Case #%d:" % prob
for i in xrange(m):
  print " ".join(answer[i])
```