# OSU Wireless

Suppose OSU Wireless will not available if we are outside of a building. Given the campus map, if we want to visit all the buildings on campus, how many times at least will we lose Wi-Fi connection?

# Easy version

None of the buildings is surrounded by any other building(s).
Which means you can reach any building from the outside.
(It is true according to campus map)

# Easy version

The optimal way to visit all the buildings:

- Starting from one of them
- Visiting them one by one
- N - 1 Wi-Fi loses for N buildings

# Solution

- Counting buildings
    - Connected components
    - DFS or BFS
- Simply output N - 1

```
................
.XXX..XXX...XXX.
.XXXX.XXXX..XXX.
.XXX...XXXXXXXX.
...X.XXX...XXXX.
.X.X.XXX...XXXX.
.X.X.XXX...XXX..
................
```

```
................
.111..222...222.
.1111.2222..222.
.111...22222222.
...1.222...2222.
.3.1.222...2222.
.3.1.222...222..
................
```

# Connected Components

- Loop through the map
    - Find an “X” not visited before
    - Find all connected “X” with it
    - Mark them as visited
    - Count as one building

# Connected Components

```
for i = 0, 1, ..., height - 1
    for j = 0, 1, ..., width - 1
        if map[i][j] == 'x'
            if not visited[i][j]
                dfs(i, j)
                count += 1
loses = max(count - 1, 0)
```

# Find all connected “X”

```
procedure dfs(i, j)
    if map[i][j] == 'X'
        visited[i][j] = true
        dfs(i, j - 1)
        dfs(i, j + 1)
        dfs(i - 1, j)
        dfs(i + 1, j)
```

# Hard version

No restriction

- A building can be in another building
- Buildings can be in other buildings
- Recursively

```
......XXXXXXXXX.XXXXXXXXX
......X.......X.X.......X
XXXXX.X.XXXXX.X.X.XXXXX.X
X...X.X.X...X.X.X.X...X.X
X.X.X.X.X.X.X.X.X.X.X.X.X
X...X.X.X...X.X.X.X...X.X
XXXXX.X.XXXXX.X.X.XXXXX.X
......X.......X.X.......X
......XXXXXXXXX.XXXXXXXXX
```

```
......XXXXXXXXX.XXXXXX...
......X.......X.X.....XX.
XXXXX.X.XXXXX.X.X.XXXX.XX
X...X.X.X...X.X.X.X...X.X
X.X.X.X.X.X.X.X.X.X.X.X.X
X...X.X.X...X.X.X.X...X.X
XXXXX.X.XXXXX.X.XX.XXXX.X
......X.......X..XX.....X
......XXXXXXXXX....XXXXXX
```

```
XXXXXXXXXXXXXXXXXXX.XXXXXXXXXXXXX
X.................X.X...........X
X.XXXXXXXXX.XXXXX.X.X.XXXX.XXXX.X
X.X...X...X.X...X.X.X.X..X.X..X.X
X.X.X.X.X.X.X.X.X.X.X.X.X.X.X.X.X
X.X...X...X.X...X.X.X.X..X.X..X.X
X.XXXXXXXXX.XXXXX.X.X.XXXX.XXXX.X
X.................X.X...........X
XXXXXXXXXXXXXXXXXXX.XXXXXXXXXXXXX
```

# Graph

We can view the map as a graph:

A vertex = A building itself

An edge = A pair of neighbor buildings

# Graph

We can view the map as a graph:

A vertex = A building itself

An edge = A pair of neighbor buildings

Adjacent to the same open space

# Hamiltonian Path

- Visiting all the vertices
- Finding the shortest one
  - Similar to TSP
    (Travelling Salesman Problem)
  - Actually shortest Hamiltonian Path
    - Repeat visitings are allowed

# Solution

- Creating the graph
- Finding the shortest path between each pair of vertices
- Finding the shortest Hamiltonian Path
- Answer = Length of the path

# Solution

- Creating the graph
  - Find buildings (vertices)
  - Find open spaces (edges)
    - DFS or BFS for both
  - Find all adjacency relationship
    - Between a vertex and an edge

# Solution

- Finding the shortest path between each pair of vertices
    - Needed by the next step
    - Floyd-Warshall
        - Do you remember it? :-)

# Solution

- Finding the shortest Hamiltonian Path
    - Dynamic Programming
    - Table-driven cache
        - Starting vertex, end vertex
        - Set of visited vertices

# Solution

- Finding the shortest Hamiltonian Path
    - Dynamic Programming
        - Initialize table[$v_i$][$v_i$][{$v_i$}]
        - Fill table[$v_i$][$v_j$][{$v_0$..$v_{n-1}$}]
    - Complexity $O(2^n * n^3)$

# Faster Algorithms?

This problem is Metric TSP (Delta TSP).
For the exact solution, it is still NP-hard.
But, several considerable fast approximation algorithms exist.

# Faster Algorithms?

Further restriction on the graph:

- 4-color for vertices + edges
- Plane (2D) graph

NP-Hard, NP, or even P?

It is still an open question.