# CS 161 Winter 2020 Section 8

March 5-6, 2020

## Mice to Holes

There are n mice and n holes along a line. Each hole can accommodate only 1 mouse. A mouse can stay at its position, move one step right from $x$ to $x + 1$, or move one step left from $x$ to $x - 1$. Any of these moves consumes 1 minute. Mice can move simultaneously. Assign mice to holes such that the time it takes for the last mouse to get to a hole is minimized, and return the amount of time it takes for that last mouse to get to its hole.

**Example:**
Mice positions: 4, -4, 2
Hole positions: 4, 0, 5
Best case: The last mouse gets to its hole in 4 minutes ($\{4 \rightarrow 4, -4 \rightarrow 0, 2 \rightarrow 5\}$ and $\{4 \rightarrow 5, -4 \rightarrow 0, 2 \rightarrow 4\}$ are both possible solutions)

## MST With Leaf Requirements

We are given an undirected weighted graph $G = (V, E)$ and a set $U \subset V$. Describe an algorithm to find a minimum spanning tree such that all nodes in $U$ are leaf nodes. (The result may not be an MST of the original graph $G$.)

## Roads and Airports

Given a set of $n$ cities, we would like to build a transportation system such that there is some path from any city $i$ to any other city $j$. There are two ways to travel: by driving or by flying. Initially all of the cities are disconnected. It costs $r_{ij}$ build a road between city $i$ and city $j$. It costs $a_i$ to build an airport in city $i$. For any two cities $i$ and $j$, we can fly directly from $i$ to $j$ if there is an airport in both cities. Give an efficient algorithm for determining which roads and airports to build to minimize the cost of connecting the cities.

## Telephone Line

A hot new Silicon Valley start-up, Blockain.AI.ML.IoT (BAMI), is moving into its new offices, which consists of two buildings. The head of HR, Jacques, needs to decide which engineers to place in each building. The buildings are very old and made of concrete, so there is no cell reception, and there is only one desk in each building that has a landline phone. In order for two friends to be able to talk to each other, they either (1) both have to be in the same building or, (2) both have to be sitting at the two desks with phones so they can call each other.

Jacques must find a way to assign engineers to buildings while still ensuring every engineer can talk to all their friends. Unfortunately, it is not possible to seat all engineers with their friends in the same building, so one pair of friends will need to talk to each other through the phone. Jacques' goal is to find a pair of friends that can sit at the desks with phones so they can still talk with each other, while also ensuring that all other engineers are in the same building as their friends.

Give a short but clear English description of your algorithm.

**Data Format**
**Input**
n: the number of engineers.
F: a mapping from an engineer to their list of friends (e.g. $F_{John}$ would refer to the friend list of John).
**Output**
($E_1$,$E_2$): a tuple containing the names of the two friends that will sit at the desks with phones.
**Assumptions**:
Your input will always lead to a valid solution (no need to handle the case where there is no solution), and the solution will always require the use of the phone. Every engineer has $\theta(\sqrt{n})$ friends.

**Example**
**Input:**
n=5
F = Alice: [Bob, John], Bob: [Alice, John], Jane: [Kathy, Susan], John: [Alice, Bob, Susan], Kathy: [Jane, Susan], Susan: [Jane, John, Kathy]
**Output**: (John, Susan)
**Explanation**: If John and Susan are at the desks with phones, it directly implies that Alice, Bob, and John are in one building, and Susan, Kathy, and Jane are in the other building. All friends would be able to talk with each other as the problem requires.