# ECE374 SP23 HW7

## Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

## Problem 1

You are given a list $D[n]$ of $n$ words each of length $k$ over an alphabet $\Sigma$. The words are sorted in lexicographic order.

Describe an algorithm to efficiently identify the order of the symbols in $\Sigma$. Assume $D[n]$ always contains enough information to completely determine the order of the symbols.

## Solution

*Intuition.* We use a graph structure where each node represents a letter in $\Sigma$. For two neighboring words in $D[n]$, we add an edge between the first pair of letters *in the same indices* that is different between the two words. A topological sort of the graph will give us the order of symbols.

---

$\text{DetermineSymbolOrder}(D[n], n, k)$
    $G \leftarrow \text{emptyGraph}()$
    **for** $a \in \Sigma$
        $G.\text{addNode}(a)$
    **for** $i \leftarrow 0$ **to** $n - 2$ $\quad$ // each pair of words
        **for** $j \leftarrow 0$ **to** $k - 1$ $\quad$ // each pair of letters
            **if** $D[i][j] \neq D[i+1][j]$
                $G.\text{addEdge}(D[i][j], D[i+1][j])$
                **break**
    **return** $G.\text{topoSort}()$

---

*Runtime analysis.* The nested **for** loop takes $O(nk)$ time in the worst case. Topological sort takes $O(n+m)$. Total time complexity is $O(nk)$.