REMINDEERS

• Assignment 09 due Friday April 5\textsuperscript{th} at 6:00 PM
• Final Exam is on Monday, April 15\textsuperscript{th}
• Office hours and review sessions are available.
• Piazza is still open for questions and announcements

Check PIAZZA!!!
REVIEW: UNDIRECTED GRAPHS

• An undirected graph $G$ is a set $V$, of vertices, and a collection $E$, of unordered pairs from $V$, called edges. We write $G = (V, E)$.

• (4, 6) in Graph 1 is an edge
  – 4, 6 are neighbours
  – 4, 6 are adjacent

• Degree is number of neighbours of a vertex
  – Degree of 5 in Graph 1 is 3

• $G_1$ is connected and $G_2$ is unconnected
QUESTION 1: REPRESENTING GRAPHS IN DIFFERENT WAYS

Question:
Show how to represent the graph using:
• **Vertex and Edge list** representation
• **Adjacency list** representation
• **Adjacency matrix** representation
  (call 'A' vertex 0, 'B' vertex 1, etc)
 Question 1 Ans:
Vertex and Edge List Representation

\[ V = \{A', B', C', D', E', F', G'\} \]


- Each pair only appears once.
ADJACENCY LIST

- Key: label of the vertex
- Value: List of adjacent vertices (neighbours)

\[
\begin{aligned}
\{'A'\} & : [\{'B\', \{'C\'}\}], \\
\{'B\'} & : [\{'A\', \{'C\', \{'D\'}\}], \\
\{'C\'} & : [\{'A\', \{'B\', \{'E\'}\}], \\
\{'D\'} & : [\{'B\'}], \\
\{'E\'} & : [\{'C\', \{'F\', \{'G\'}\}], \\
\{'F\'} & : [\{'E\'}], \\
\{'G\'} & : [\{'E\'}] \\
\end{aligned}
\]
QUESTION 1 ANS:
ADJACENCY MATRIX REPRESENTATION

```
A B C D E F G
[ [0,1,1,0,0,0,0], #A
  [1,0,1,1,0,0,0], #B
  [1,1,0,0,1,0,0], #C
  [0,1,0,0,0,0,0], #D
  [0,0,1,0,0,1,1], #E
  [0,0,0,0,1,0,0], #F
  [0,0,0,0,1,0,0]] #G
```
QUESTION 2A: DRAW THE GRAPH

Draw the graph corresponding to the following adjacency list.

```
{'A': ['B', 'C'],
'B': ['A', 'D', 'E', 'F'],
'C': ['A'],
'D': ['B', 'E'],
'E': ['B', 'D', 'F'],
'F': ['B', 'E']}
```
QUESTION 2A ANS: ADJACENCY LIST
QUESTION 2B: DRAW THE GRAPH

Draw the graph corresponding to the following adjacency matrix.

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
QUESTION 2B ANS: ADJACENCY MATRIX
• Traversals:
  – Finding all connected vertices, starting from A

• Two approaches
  – **Breadth-first search** (from starting vertex, then neighbours, then neighbours of neighbours, etc.)
  – **Depth-first search** (from starting vertex follow path as far as possible, back up to closest unvisited neighbour, repeatedly)
QUESTION 3 (A)

• Perform bfs and dfs traversals for the following graphs.
• Starting from A and E

Graph 1

Graph 2
Recursive implementation of dfs traversal

```python
def dfs(graph, v):
    visited = []
    return visit(graph, v, visited)

def visit(g, v, all):
    all.append(v)
    for neighbour in g[v]:
        if neighbour not in all:
            visit(g, neighbour, all)
    return all
```
Write the function `edges_count` that consumes a nonempty graph \( G \) (stored as an adjacency list) and returns the number of edges in \( G \).

Examples:

\[
G1 = \{1:[2, 5], 2:[1, 3], 3:[2], 4:[5], 5:[1, 4]\}
\]

\[
G2 = \{1:[2, 4], 2:[1, 3, 4, 5], 3:[2, 5], 4:[1, 2], 5:[2, 3], 6:\[]\}
\]

`edges_count(G1) => 4`

`edges_count(G2) => 6`
QUESTION 5: DEGREE__ADJ__MAT

Write the function `degree_adj_mat` that consumes a nonempty graph $G$ (stored as an adjacency matrix) and a vertex number $v$, and returns the degree of vertex $v$ in $G$.

Note that for adjacency matrix:
• the vertices are numbered $0, 1, \ldots, n-1$.
• $G$ has $n$ lists, and each list has a length $n$ as well.

Challenge Question:
On your own, implement the functions `degree_adj_list` and `degree_edges`, to determine the degree of a vertex using the other representations.
Write the function `list_dictionary` that consumes a graph $G$ (stored as an adjacency list), and returns a list of edges. Edge should be in the format of `[a, b]` where $a < b$.

Examples:

$G1 = \{1: [2, 5], 2: [1, 3, 5], 3: [2, 4], 4: [3, 5, 6],
5: [1, 2, 4], 6: [4]\}$

`list_dictionary(G1) => [[1, 2], [1, 5], [2, 3],
[2, 5], [3, 4], [4, 5], [4, 6]]`
**QUESTION 6B: ADMATRIX_DICTIONARY**

Write the function `admatrix_dictionary` that consumes a graph $G$ (stored as an adjacency list), and returns the graph which is stored as an adjacency matrix.

Examples:

$G1 = \{1: [2, 5], 2: [1, 3, 5], 3: [2, 4], 4: [3, 5, 6], 5: [1, 2, 4], 6: [4]\}$

`admatrix_dictionary(G1) =>
[[0, 1, 0, 0, 1, 0], [1, 0, 1, 0, 1, 0], [0, 1, 0, 1, 0, 0],
[0, 0, 1, 0, 1, 1], [1, 1, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0]]`

**Note:** row 0 is for vertex 1
QUESTION 6C: ADD_EDGE

Write the function add_edge that consumes a graph G (stored as an adjacency list), and add an edge , [a, b], where each of a, b can be new node or an existed one. You may assume that a and b are not neighbors.

Examples:
G1 = {1: [2, 5], 2: [1, 3, 5], 3: [2, 4], 4: [3, 5, 6], 5: [1, 2, 4], 6: [4]}
a = 7, b = 1
add_edge(G, a, b) => None
G1 = {1: [2, 5, 7], 2: [1, 3, 5], 3: [2, 4], 4: [3, 5, 6], 5: [1, 2, 4], 6: [4], 7: [1]}