WORKSHEET

A4.2 Adjacency matrix and adjacency list

Trees and graphs · A level · OCR H446 1.4.2, AQA 7517 4.2.4.1, Eduqas A500QS 1.1 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Two ways to store a graph, and choosing between them for dense and sparse graphs.

- The adjacency matrix
- The adjacency list
- Which to use

Questions

6 marks in all

1. An adjacency matrix is symmetric about its leading diagonal. What does that tell you? [1 mark]

- ☐ **A** The graph is undirected, or every edge has a matching edge the other way
- ☐ **B** The graph is weighted
- ☐ **C** The graph has no cycles
- ☐ **D** Every vertex has the same degree

2. How many cells does the adjacency matrix of a graph with 50 vertices have? [1 mark]

3. A road map has 10,000 junctions, and each junction joins about 3 roads. Which representation suits it, and why? [1 mark]

- ☐ **A** An adjacency list, because the graph is sparse
- ☐ **B** An adjacency matrix, because the graph is sparse
- ☐ **C** An adjacency matrix, because it uses less memory
- ☐ **D** An adjacency list, because it tests for an edge in one look

4. What is an advantage of an adjacency matrix over an adjacency list? [1 mark]

- ☐ **A** Checking whether two particular vertices are joined takes one look
- ☐ **B** It uses less memory for a sparse graph
- ☐ **C** Adding a new vertex needs no extra space
- ☐ **D** Listing a vertex's neighbours never looks at more cells than it has neighbours

5. What does this program print? [1 mark]

```
VERTICES = ["A", "B", "C"]
matrix = [[0, 1, 1],
          [0, 0, 1],
          [1, 0, 0]]
for i in range(3):
    out = []
    for j in range(3):
        if matrix[i][j] == 1:
            out.append(VERTICES[j])
    print(VERTICES[i], out)
```

6. What word describes a graph that has only a small number of edges compared with the number it could have? [1 mark]

The task: matrix and list

The graph in `EDGES` is **directed** and weighted, as drawn below: `("A", "B", 4)` is an edge from A to B of weight 4, and there is no edge from B to A unless one is listed. `VERTICES` gives the order of the matrix's rows and columns. Build an adjacency matrix called `matrix`, a list of lists with `matrix[i][j]` holding the weight of the edge from `VERTICES[i]` to `VERTICES[j]` and 0 where there is no edge, and print it one row per line: the vertex, a colon, then the row's values separated by single spaces, such as `A: 0 4 0 7`. Then build an adjacency list as a dictionary, with each vertex's neighbours in the order their edges appear in `EDGES`, and print it one vertex per line in `VERTICES` order, in the form `A -> B(4) D(7)`. Build both from `EDGES`: do not type the rows.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

VERTICES = ["A", "B", "C", "D"]
EDGES = [("A", "B", 4), ("A", "D", 7), ("B", "C", 3), ("C", "A", 2), ("D", "C", 5), ("B",
"D", 1)]

n = len(VERTICES)
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a4-2-adjacency-matrix-and-list/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Print the matrix with every arrow reversed. What does row A tell you now?
2. Write `in_degree(v)` using the matrix and `out_degree(v)` using the list. Which representation makes each one easier, and why?
3. A graph has 1,000 vertices and each has about 5 neighbours. How many cells does its adjacency matrix have? About how many entries does its adjacency list hold if the edges are directed?