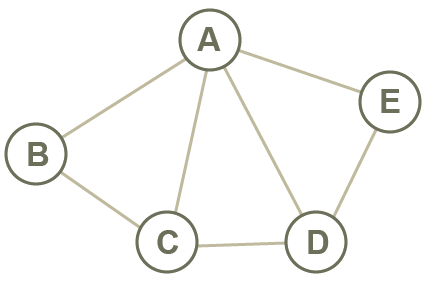
# Homework 5 Graphs Answers

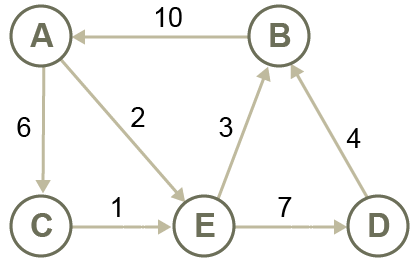
# 1. Draw the adjacency matrix for this graph. [5]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** |
| **A** |  | 1 | 1 | 1 | 1 |
| **B** | 1 |  | 1 |  |  |
| **C** | 1 | 1 |  | 1 |  |
| **D** | 1 |  | 1 |  | 1 |
| **E** | 1 |  |  | 1 |  |



# 2. Draw the adjacency list for this graph. [5]

|  |  |  |  |
| --- | --- | --- | --- |
| **A** |  |  | {C:6, E:2} |
| **B** |  |  | {A:10} |
| **C** |  |  | {E:1} |
| **D** |  |  | {B:4} |
| **E** |  |  | {B3:D:7} |



# 3. Contrast the use of an adjacency matrix with that of an adjacency list [5]

* An adjacency matrix is a 2-dimensional structure, with a **single cell for each edge**. It’s very **easy to add an edge** between two nodes by updating a cell. An edge can be deleted by clearing a cell. It’s quite easy to build a 2-dimensional array structure with a static length. Adding more nodes, after the initial build may involve reallocation and copying of the array. This is could be costly in both memory and processing time. It is most **applicable when the number of edges is high/the graph is highly connected**.
* An adjacency list, is a single array of nodes. It can be represented by a **dictionary, with the keys being the nodes and the values being lists of adjacent nodes** . This means that there is **no memory taken for edges that do not exist**. Adding edges and nodes is easily done by adding to the list or dictionary. It is most **applicable when the graph is sparsely connected.**

# C:\Users\Rob\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\1.png4. Draw the graph for this adjacency list. [5]

|  |  |  |  |
| --- | --- | --- | --- |
| **A** |  |  | [D, E] |
| **B** |  |  | [C, E] |
| **C** |  |  | [B, D] |
| **D** |  |  | [A, C] |
| **E** |  |  | [A, B] |

An alternative equivalent graph may be drawn with A, E, B, C, D in a circle, either clockwise or anti-clockwise.

[Total 20 Marks]