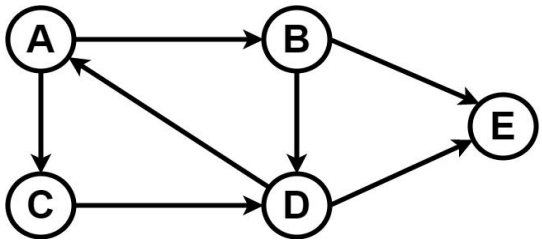
22BIO211: Intelligence of Biological Systems - 2

Lab Sheet 1

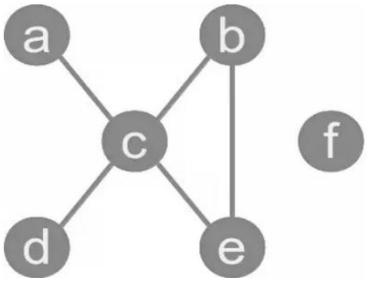
1. Answer the following.



a) Construct a Graph in Python for the given Structure (adjacency list).
b) Print the graph using simple print statement.
c) Print all vertices of the Graph using keys () function.
d) Print all edges of the Graph.

Sample Solution

a) The Python data structure 'Dictionary' is ideal for representing such graphs in Python, The keys of the dictionary are the nodes of graph, The corresponding values are list with the nodes, which are connected by an edge.

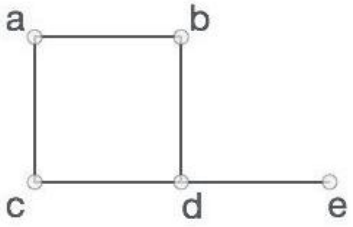



```
graph = { "a" : ["c“],
          "b" : ["c", "e“],
          "c" : ["a", "b", "d","e“],
         "d" : ["c“],
         "e" : ["c", "b“],
          "f" : []
}
```

b) # Print the graph
```
print(graph)
```
c) #Print all vertices of the Graph
```
print("The vertices of the graph are")
print(list(graph.keys()))
```
d) #Print all edges of the Graph
```
print("The edges of the graph are")
edges=[]
```

```
        # for each node(key) in graph
        for node in graph:
            # for each neighbour node of a single node
            for neighbour in graph[node]:
            # if edge exists then append
            edges.append((node, neighbour))
        print(edges)
```

Sample output




```
{'a': ['b', 'c'], 'b': ['a', 'd'], 'c': ['a', 'd'], 'd': ['e'], 'e': ['d']}
The vertices of the graph are
['a', 'b', 'c', 'd', 'e']
The edges of the graph are
[('a', 'b'), ('a', 'c'), ('b', 'a'), ('b', 'd'), ('c', 'a'), ('c', 'd'),
    ('d', 'e'), ('e', 'd')]
```

2. Construct the above graph in question 1 by reading user input. Add each edge of the graph by reading the vertex pair one by one from the user. Print the vertices and edges of the graph.

Sample Solution

a) Construct an empty dictionary
   a. graph = dict()

b) Add each edge

```
addEdge('A', 'B')
addEdge('A', 'C')
addEdge('B', 'D')
Etc….
```

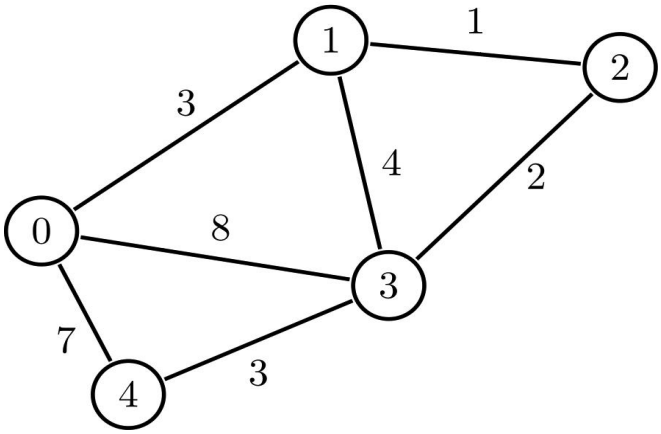
c) Define AddEdge method

```
def addEdge(node1, node2):
# create an empty list for a key node
        if node1 not in graph:
            graph[node1] = []
        if node2 not in graph:
            graph[node2] = []
# append the neighbor node to its corresponding key node
        graph[node1].append(node2)
```

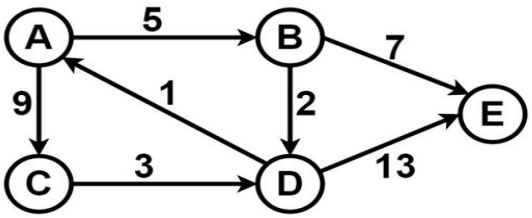
d) Print the vertices and edges of the graph

```
for key, val in graph.items():
  print(f"{key}-->{val}")
```

3. Construct the following graph in Python by reading user input: Read the full graph structure together. Print the vertices and edges of the graph.



Sample Solution : For the following graph



A : (B, 5), (C, 9)
B : (D, 2), (E, 7)
C : (D, 3)
D : (A, 1), (E, 13)
E :

First input the number of nodes and edges of the graph

5 7 - > nodes, edges = input().split()

Then, input each pair of nodes having an edge between them, and the associated cost

```
5 7
A B 5
A C 9
B D 2
B E 7
C D 3
D A 1
D E 13
```

```
for x in range(int(edges)):
    node1, node2,cost = input().split()
    addEdge(node1, node2,cost)
```

4. Construct the graph in question no. 3, by reading user input from a file. Print the vertices and edges of the graph.

Sample Solution:

For the given graph,
Create a file input.txt with the following content

```
5 7
A B 5
A C 9
B D 2
B E 7
C D 3
D A 1
D E 13
```

Write a program to read from the file

```
with open("input.txt") as f:
    lines = f.readlines()
nodes, edges = lines[0].split()
#first line: number of nodes and edges.
 # Pair of nodes starts from second line
 for i in range(1, len(lines)):
     node1, node2, cost = lines[i].split()
     g.addEdge(node1, node2, int(cost))
```

5. Visualize the above graph using 'networkx'. The vertex labels and edge weights should be shown.The graph should be drawn as a directed graph. Save the output as a .png file in your computer.

6. Generate a random DNA sequence of length 12 and create the K-mer composition of it where K = 3. Also reconstruct the DNA sequence back using the K-mers.

7. Using the K-mer composition of question 6, draw an overlap graph and visualize it using 'networkx'. The vertices should show the K-mers and the edges should include the overlap length.