Task 1

In [ ]:
```
import numpy as np
```

In [ ]:
```
h_0 = {
    'A': np.array([2, 1], dtype='float'),
    'B': np.array([0.03, 2], dtype='float'),
    'C': np.array([4, 1.75], dtype='float'),
    'D': np.array([-6.78, 1.02], dtype='float'),
    'E': np.array([-1, 1], dtype='float')
}

h_1 = {
}

w_0 = np.array([[1, 2.01], [2.3, 3.4], [1.7, -0.91]], dtype='float')
b_0 = np.array([[-1.13, 0.05], [3.46, 1.14], [-2.72, 0.002]], dtype='float')

def sigmoid(z):
    return np.exp(z) / (1 + np.exp(z))

def calculate_embedings(W, B, node, neighbours):
    ts = ()
    for x in neighbours:
        ts += (h_0[x],)
    sum = np.mean(np.vstack(ts), axis=0)
    h_1[node] = sigmoid(W@sum + B@h_0[node])

calculate_embedings(w_0, b_0, 'A', ['D', 'E'])
calculate_embedings(w_0, b_0, 'B', ['C'])
calculate_embedings(w_0, b_0, 'C', ['B'])
calculate_embedings(w_0, b_0, 'D', ['E', 'A'])
calculate_embedings(w_0, b_0, 'E', ['D', 'A'])

for k,v in h_1.items():
    print(k, v)
```

```
A [1.67922990e-02 9.27371713e-01 2.32892836e-06]
B [0.99949153 0.99999998 0.99411765]
C [4.05524069e-01 1.00000000e+00 3.22201627e-06]
D [9.99963654e-01 1.96340601e-08 9.99999990e-01]
E [0.69425757 0.01233382 0.09448181]
```

Task 2

In [ ]:

```
Hs = [
    {
        'A': np.array([2, 1], dtype='float'),
        'B': np.array([0.03, 2], dtype='float'),
        'C': np.array([4, 1.75], dtype='float'),
        'D': np.array([-6.78, 1.02], dtype='float'),
        'E': np.array([-1, 1], dtype='float')
    }
]

Ws = [
    np.array([[1, 2.01], [2.3, 3.4], [1.7, -0.91]], dtype='float')
]

Bs = [
    np.array([[-1.13, 0.05], [3.46, 1.14], [-2.72, 0.002]], dtype='float')
]

def sigmoid(z):
    return np.exp(z) / (1 + np.exp(z))

def calculate_embedings(W, B, node, neighbours, h):
    ts = ()
    for x in neighbours:
        ts += (h[x],)
    sum = np.sum(np.vstack(ts), axis=0)
    return sigmoid(W@sum + B@h[node])

neighbourhood_info = {
    'A': ['D', 'E'],
    'B': ['C'],
    'C': ['B'],
    'D': ['E', 'A'],
    'E': ['D', 'A']
```

```
}

for l in range(0, 6):
    if l == 1:
        Ws.append(np.array([[1, 1, 0], [0, 1, -1], [-1, 0, 1]], dtype='float'))
        Bs.append(np.array([[0, -1, 0], [1, 0, 1], [0, -1, 1]], dtype='float'))
    elif l>1:
        Ws.append(Ws[l-1]@Ws[l-1].T)
        Bs.append(Bs[l-1]@Bs[l-1].T)
    layer = {}
    for node in neighbourhood_info:
        layer[node] = calculate_embedings(Ws[l], Bs[l], node, neighbourhood_info[node], Hs[l])
    Hs.append(layer)

for l in range(1, 6):
    print('layer', l)
    for k,v in Hs[l].items():
        print(k, v)
```

```
layer 1
A [2.65196213e-03 4.89857317e-02 1.24740555e-09]
B [0.99949153 0.99999998 0.99411765]
C [4.05524069e-01 1.00000000e+00 3.22201627e-06]
D [9.99997046e-01 1.85801497e-06 9.99999989e-01]
E [0.61306162 0.00158434 0.00071522]
layer 2
A [0.8271639  0.26963464 0.34044109]
B [0.60001416 0.95228443 0.39857572]
C [0.73095859 0.60142583 0.2678868 ]
D [0.66067009 0.88593265 0.59507842]
E [0.74078493 0.41648115 0.49912047]
layer 3
A [0.98491315 0.97794241 0.77940626]
B [0.94235324 0.98145922 0.82532342]
C [0.94005903 0.97279429 0.75269   ]
D [0.98577182 0.98865494 0.89676798]
E [0.98829102 0.98516079 0.79949559]
layer 4
A [0.99999999 1.         0.87880212]
B [0.99999435 0.99999988 0.99871628]
C [0.99999032 0.99999977 0.99858664]
D [1.         1.         0.88937937]
E [0.99999999 1.         0.88826336]
layer 5
A [1.00000000e+00 1.00000000e+00 3.86289486e-26]
B [1. 1. 1.]
C [1. 1. 1.]
D [1.0000000e+00 1.0000000e+00 2.9653241e-26]
E [1.00000000e+00 1.00000000e+00 3.04922104e-26]
C:\Users\Amir Hossein\AppData\Local\Temp\ipykernel_9172\3086761627.py:20: RuntimeWarning: overflow encountered in exp
  return np.exp(z) / (1 + np.exp(z))
C:\Users\Amir Hossein\AppData\Local\Temp\ipykernel_9172\3086761627.py:20: RuntimeWarning: invalid value encountered i
n divide
  return np.exp(z) / (1 + np.exp(z))
```

In [ ]: