LAB DAY-10

1. Assembly Line Scheduling

```
def als(a, t, e, x):
    n = len(a[0])
    T1 = [0] * n
    T2 = [0] * n

    T1[0] = e[0] + a[0][0]
    T2[0] = e[1] + a[1][0]

    for i in range(1, n):
        T1[i] = min(T1[i-1] + a[0][i], T2[i-1] + t[1][i] + a[0][i])
        T2[i] = min(T2[i-1] + a[1][i], T1[i-1] + t[0][i] + a[1][i])

    return min(T1[n-1] + x[0], T2[n-1] + x[1])

a = [[4, 5, 3, 2], [2, 10, 1, 4]]
t = [[0, 7, 4, 5], [0, 9, 2, 8]]
e = [10, 12]
x = [18, 7]

print(als(a, t, e, x))
```

Output: 35

2. Knapsack Problem and Memory

```
def knapsack(weights, values, capacity):
    n = len(weights)
    dp = [[0 for _ in range(capacity + 1)] for _ in range(n + 1)]

    for i in range(1, n + 1):
        for w in range(capacity + 1):
            if weights[i-1] <= w:
                dp[i][w] = max(dp[i-1][w], dp[i-1][w-weights[i-1]] + values[i-1])
            else:
                dp[i][w] = dp[i-1][w]

    return dp[n][capacity]

weights = [1, 3, 4, 5]
values = [1, 4, 5, 7]
capacity = 7

print(knapsack(weights, values, capacity))
```

Output: 9

3. Bellman-Ford Algorithm

```
class Edge:
    def __init__(self, u, v, w):
        self.u = u
        self.v = v
        self.w = w

def bellmanford(edges, V, src):
    dist = [float('inf')] * V
    dist[src] = 0

    for _ in range(V - 1):
        for edge in edges:
            if dist[edge.u] + edge.w < dist[edge.v]:
                dist[edge.v] = dist[edge.u] + edge.w
ege
    for edge in edges:
        if dist[edge.u] + edge.w < dist[edge.v]:
            print("Graph contains a negative weight cycle")
            return None

    return dist

edges = [Edge(0, 1, -1), Edge(0, 2, 4), Edge(1, 2, 3), Edge(1, 3, 2), Edge(1, 4, 2),
         Edge(3, 2, 5), Edge(3, 1, 1), Edge(4, 3, -3)]
V = 5
src = 0

print(bellmanford(edges, V, src))
```

Output: [0, -1, 2, -2, 1]

4. Floyd-Warshall Algorithm:

```
def floydwarshall(graph):
    V = len(graph)
    dist = [[graph[i][j] for j in range(V)] for i in range(V)]

    for k in range(V):
        for i in range(V):
            for j in range(V):
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

    return dist

inf = float('inf')
graph = [[0, 3, inf, 5],
         [2, 0, inf, 4],
         [inf, 1, 0, inf],
         [inf, inf, 2, 0]]

print(floydwarshall(graph))
```

Output: [[0, 3, 7, 5], [2, 0, 6, 4], [3, 1, 0, 5], [5, 3, 2, 0]]