Practical=8

Name: shubham shivraj Suryawanshi
Reg. No:2020BIT004

Write a Program to implement,
Floyd-Warshall
Knapsack problem
Write a Algorithm with complete Simulation

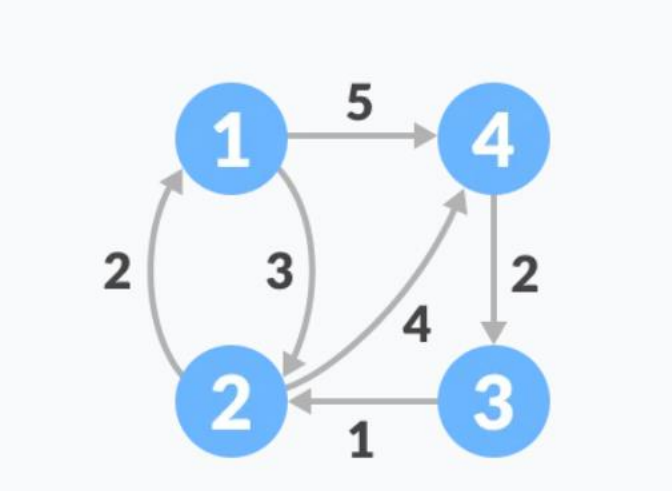
1) Floyd-Warshall

```
#include <stdio.h>

#define nV 4

#define INF 999

void printMatrix(int matrix[][nV]);


void floydWarshall(int graph[][nV]) {
  int matrix[nV][nV], i, j, k;

  for (i = 0; i < nV; i++)
    for (j = 0; j < nV; j++)
      matrix[i][j] = graph[i][j];



  for (k = 0; k < nV; k++) {
    for (i = 0; i < nV; i++) {
      for (j = 0; j < nV; j++) {
        if (matrix[i][k] + matrix[k][j] < matrix[i][j])
          matrix[i][j] = matrix[i][k] + matrix[k][j];
      }
    }
  }
  printMatrix(matrix);
}

void printMatrix(int matrix[][nV]) {
  for (int i = 0; i < nV; i++) {
    for (int j = 0; j < nV; j++) {
      if (matrix[i][j] == INF)
        printf("%4s", "INF");
      else
        printf("%4d", matrix[i][j]);
    }
    printf("\n");
  }
}

int main() {
  int graph[nV][nV] = {{0, 3, INF, 5},
          {2, 0, INF, 4},
          {INF, 1, 0, INF},
          {INF, INF, 2, 0}};
  floydWarshall(graph);
}}
```

Output:

```
   0   3   7   5
   2   0   6   4
   3   1   0   5
   5   3   2   0
PS D:\Assignments TY\DAA\Codes\output>
```

Simulation:



2) Knapsack problem

```
#include <stdio.h>

int max(int a, int b) { return (a > b) ? a : b; }

int knapSack(int W, int wt[], int val[], int n)
{

   if (n == 0 || W == 0)
      return 0;


   if (wt[n - 1] > W)
      return knapSack(W, wt, val, n - 1);

   else
      return max(
         val[n - 1]
            + knapSack(W - wt[n - 1], wt, val, n - 1),
         knapSack(W, wt, val, n - 1));
}

int main()
{
   int profit[] = { 60, 100, 120 };
   int weight[] = { 10, 20, 30 };
   int W = 50;
   int n = sizeof(profit) / sizeof(profit[0]);
   printf("%d", knapSack(W, weight, profit, n));
   return 0;
}Output:
```

```
PS D:\Assignments TY\DAA\Codes\output> & .\'Knapsack.exe'
220
PS D:\Assignments TY\DAA\Codes\output>
```

Compiled successfully!

Simulation: