## 16/04/2025 – LAB 4

1. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm

```
#include <stdio.h>
#include <limits.h>

#define MAX 20
#define INF 999

int main() {
   int i, j, n, v, u, min, k;
   int c[MAX][MAX], d[MAX], s[MAX];

   printf("Enter the number of vertices: ");
   scanf("%d", &n);

   printf("Enter the cost adjacency matrix (use 999 for no edge):\n");
   for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++) {
         scanf("%d", &c[i][j]);
      }
   }

   printf("Enter the source vertex (0 to %d): ", n - 1);
   scanf("%d", &v);

   for (i = 0; i < n; i++) {
      d[i] = c[v][i];
      s[i] = 0;
   }

   d[v] = 0;
   s[v] = 1;

   for (k = 1; k < n; k++) {
      min = INF;
      u = -1;

      for (i = 0; i < n; i++) {
         if (!s[i] && d[i] < min) {
            min = d[i];
            u = i;
         }
      }

      if (u == -1) break;
```

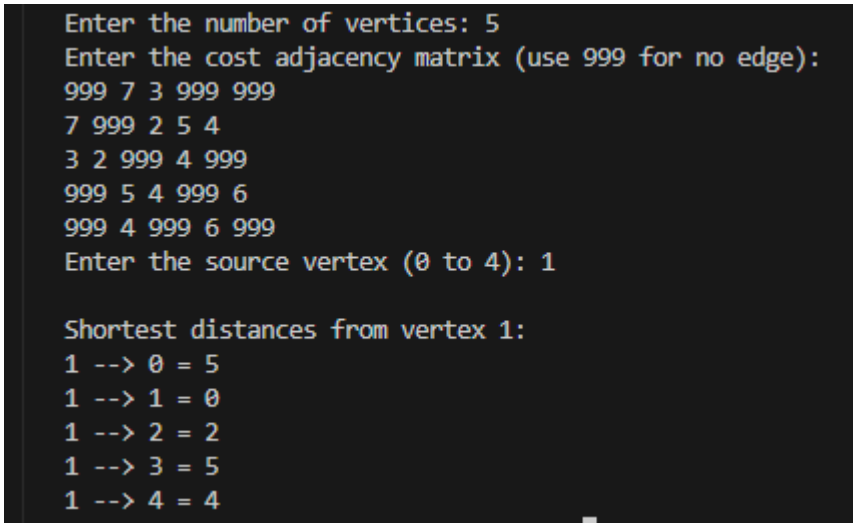
```
      s[u] = 1;

      for (i = 0; i < n; i++) {
         if (!s[i] && d[i] > d[u] + c[u][i]) {
            d[i] = d[u] + c[u][i];
         }
      }
   }

   printf("\nShortest distances from vertex %d:\n", v);
   for (i = 0; i < n; i++) {
      printf("%d --> %d = %d\n", v, i, d[i]);
   }

   return 0;
}
```

**OUTPUT:**



2. Implement 0/1 Knapsack problem using dynamic programming.

```
#include <stdio.h>

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

void knapsack(int n, int w[], int p[], int c) {
   int v[20][20];
   int i, j;

   for (i = 0; i <= n; i++) {
      for (j = 0; j <= c; j++) {
         if (i == 0 || j == 0)
            v[i][j] = 0;
         else if (w[i - 1] > j)
            v[i][j] = v[i - 1][j];
         else
            v[i][j] = max(v[i - 1][j], p[i - 1] + v[i - 1][j - w[i - 1]]);
      }
   }

   printf("\nMaximum Profit is: %d\n", v[n][c]);

   printf("\nDP Table:\n");
   for (i = 0; i <= n; i++) {
      for (j = 0; j <= c; j++) {
         printf("%4d", v[i][j]);
      }
      printf("\n");
   }
}

int main() {
   int n, c, i;
   int w[10], p[10];

   printf("Enter the number of objects: ");
   scanf("%d", &n);

   printf("Enter the weights of the objects:\n");
   for (i = 0; i < n; i++) {
      scanf("%d", &w[i]);
   }

   printf("Enter the profits of the objects:\n");
   for (i = 0; i < n; i++) {
      scanf("%d", &p[i]);
```

```
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &c);

    knapsack(n, w, p, c);

    return 0;
}
```

**OUTPUT:**