

הדגמת הרצת מקרי קצה בתוכנית ובדיקה על ידי gcov של כיסוי כלל התוכנית:

```
aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ ./main
Enter the number of the vertex in the graph:2
Enter the weight of the edges from vertex 0 to the other vertices: 0 1
Enter the weight of the edges from vertex 1 to the other vertices: 2 0
Vertex      Distance from Source
0           2
1           0
Enter the number of the vertex in the graph:1000
Enter the weight of the edges from vertex 0 to the other vertices: -3 2
invalid valueaharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ ./main
Enter the number of the vertex in the graph:1001
The maximum number of vertices is 1000aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ ./main
Enter the number of the vertex in the graph:3
Enter the weight of the edges from vertex 0 to the other vertices: 1 2 3 4
Error: You entered too many values in a line.
aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ ./main
Enter the number of the vertex in the graph:2
Enter the weight of the edges from vertex 0 to the other vertices: 0 1
Enter the weight of the edges from vertex 1 to the other vertices: 2 0
Vertex      Distance from Source
0           2
1           0
Enter the number of the vertex in the graph:a
Error:You entered invalid value.
aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ lcov --capture --directory . --output-file coverage.info
Capturing coverage data from .
Subroutine read_intermediate_text redefined at /usr/bin/geninfo line 2623.
Subroutine read_intermediate_json redefined at /usr/bin/geninfo line 2655.
Subroutine intermediate_text_to_info redefined at /usr/bin/geninfo line 2703.
Subroutine intermediate_json_to_info redefined at /usr/bin/geninfo line 2792.
Subroutine get_output_fd redefined at /usr/bin/geninfo line 2872.
Subroutine print_gcov_warnings redefined at /usr/bin/geninfo line 2900.
Subroutine process_intermediate redefined at /usr/bin/geninfo line 2930.
Found gcov version: 11.4.0
Using intermediate gcov format
Scanning . for .gcda files ...
Found 1 data files in .
Processing main.gcda
Finished .info-file creation
aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ genhtml coverage.info --output-directory out
Reading data file coverage.info
Found 1 entries.
Found common filename prefix "/home/aharonba123/Desktop/H.W/Semester_B/Operating_Systems/exe1"
Writing .css and .png files.
Generating output.
Processing file 4/main.c
Writing directory view page.
Overall coverage rate:
  lines.....: 100.0% (51 of 51 lines)
  functions...: 100.0% (5 of 5 functions)
aharonba123@aharon:~/Desktop/H.W/Semester_B/Operating_Systems/exe1/4$ gcov main.c
File 'main.c':
  Lines executed:100.00% of 51
  Creating 'main.c.gcov'
  Lines executed:100.00% of 51
```

LCOV - coverage.info - 4/main.c

file:///home/aharonba123/Desktop/H.W/Semester_B/Operating_Systems/exe1/4/out/4/main.c.gcov.html67%

LCOV - code coverage report

Current view: top level - 4 - main.c (source / functions)		Hit		Total		Coverage	
Test: coverage.info		Lines:	51		51		100.0 %
Date: 2024-05-16 01:11:23		Functions:	5		5		100.0 %

Line data	Source code
1	1 // C program for Dijkstra's single source shortest path
2	2 // algorithm. The program is for adjacency matrix
3	3 // representation of the graph
4	4
5	5 #include <limits.h>
6	6 #include <stdio.h>
7	7 #include <stdlib.h>
8	8 #include <string.h>
9	9 #include <syslib.h>
10	10
11	11 // Number of vertices in the graph
12	12 #define MAX_VERTICES 1000
13	13
14	14 // A utility function to find the vertex with minimum
15	15 // distance value, from the set of vertices not yet included
16	16 // in shortest path tree
17	17 2 int minDistance(int dist[], bool sptSet[], int V)
18	18 {
19	19 // Initialize min value
20	20 2 int min = INT_MAX, min_index;
21	21
22	22 for (int v = 0; v < V; v++)
23	23 4 if (sptSet[v] == false && dist[v] < min)
24	24 4 min = dist[v], min_index = v;
25	25
26	26 2 return min_index;
27	27
28	28 // A utility function to print the constructed distance
29	29 // array
30	30 2 void printSolution(int dist[], int V)
31	31 {
32	32 printf("Vertex \t\t Distance from Source\n");
33	33 for (int i = 0; i < V; i++)
34	34 printf("%d \t\t\t %d\n", i, dist[i]);
35	35 }
36	36
37	37 // Function that implements Dijkstra's single source
38	38 // shortest path algorithm for a graph represented using
39	39 // adjacency matrix representation
40	40 void Dijkstra(int graph[MAX_VERTICES][MAX_VERTICES], int V, int src)
41	41 {
42	42 4 int dist[V]; // The output array. dist[i] will hold the
43	43 // shortest
44	44 // distance from src to i
45	45
46	46 2 bool sptSet[V]; // sptSet[i] will be true if vertex i is
47	47 // included in shortest
48	48 // path tree or shortest distance from src to i is
49	49 // finalized
50	50
51	51 // Initialize all distances as INFINITE and sptSet[] as
52	52 // false
53	53 for (int i = 0; i < V; i++)
54	54 6 dist[i] = INT_MAX, sptSet[i] = false;
55	55
56	56 // Distance of source vertex from itself is always 0
57	57 2 dist[src] = 0;
58	58
59	59 // Find shortest path for all vertices
60	60 4 for (int count = 0; count < V - 1; count++)
61	61 {
62	62 // Pick the minimum distance vertex from the set of
63	63 // vertices not yet processed. u is always equal to
64	64 // src in the first iteration.
65	65 2 int u = minDistance(dist, sptSet, V);
66	66
67	67 // Mark the picked vertex as processed
68	68 sptSet[u] = true;
69	69
70	70 // Update dist value of the adjacent vertices of the
71	71 // picked vertex.
72	72 6 for (int v = 0; v < V; v++)
73	73 {
74	74 // Update dist[] only if it is not in sptSet,
75	75 // there is an edge from u to v, and total
76	76 // weight of path from src to v through u is
77	77 // smaller than current value of dist[v]
78	78 4 if (sptSet[u] && graph[u][v] && dist[u] + graph[u][v] < dist[v])
79	79 4 dist[v] = dist[u] + graph[u][v];
80	80 }
81	81
82	82 // print the constructed distance array
83	83 2 printSolution(dist, V);

```

77 : // removes some useless values of matrix
78 : if (!topSet(v) && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
79 : {
80 :     dist[v] = dist[u] + graph[u][v];
81 : }
82 :
83 : // print the constructed distance array
84 : 2 : printSolution(dist, V);
85 : 2 : }
86 :
87 : 4 : void scanMatrix(int adjMatrix[MAX_VERTIX][MAX_VERTIX], int numOfVertex)
88 : {
89 :     int ch;
90 :     for (int i = 0; i < numOfVertex; i++)
91 :     {
92 :         printf("Enter the weight of the edges from vertex %d to the other vertices: ", i);
93 :         for (int j = 0; j < numOfVertex; j++)
94 :         {
95 :             int value;
96 :             // Checking that the user enters valid values into the metric
97 :             if (scanf("%d", &value) != 1 || value < 0)
98 :             {
99 :                 printf("invalid value");
100 :                 exit(1);
101 :             }
102 :             adjMatrix[i][j] = value;
103 :         }
104 :         // Checking that the user enters an exact number of values per line
105 :         while ((ch = getchar()) != '\n' && ch != EOF)
106 :         {
107 :             if (!isspace(ch))
108 :             {
109 :                 printf("Error: You entered too many values in a line.\n");
110 :                 exit(1);
111 :             }
112 :         }
113 :     }
114 : }
115 :
116 : // driver's code
117 : 6 : int main()
118 : {
119 :     for (;;)
120 :     {
121 :         int ch;
122 :         int numOfVertex;
123 :         int adjMatrix[MAX_VERTIX][MAX_VERTIX];
124 :         printf("Enter the number of the vertex in the graph:");
125 :         scanf("%d", &numOfVertex);
126 :         // Checking that the user entered a valid value of number of vertices
127 :         if ((ch = getchar()) != '\n' && ch != EOF)
128 :         {
129 :             printf("Error: You entered invalid value.\n");
130 :             return 1;
131 :         }
132 :         else if (numOfVertex > MAX_VERTIX)
133 :         {
134 :             printf("The maximum number of vertices is 1000");
135 :             return 1;
136 :         }
137 :         // After entering a correct value, the scanMatrix function is called in order to receive the graph values from the user
138 :         scanMatrix(adjMatrix, numOfVertex);
139 :         // Running the Dijkstra algorithm
140 :         dijkstra(adjMatrix, numOfVertex, 1);
141 :     }
142 :     return 0;
143 : }

```