200650U – Thilakarathne D.L.J - CS2023 – Lab 12

GitHub link - https://github.com/LasithaJananjaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab12

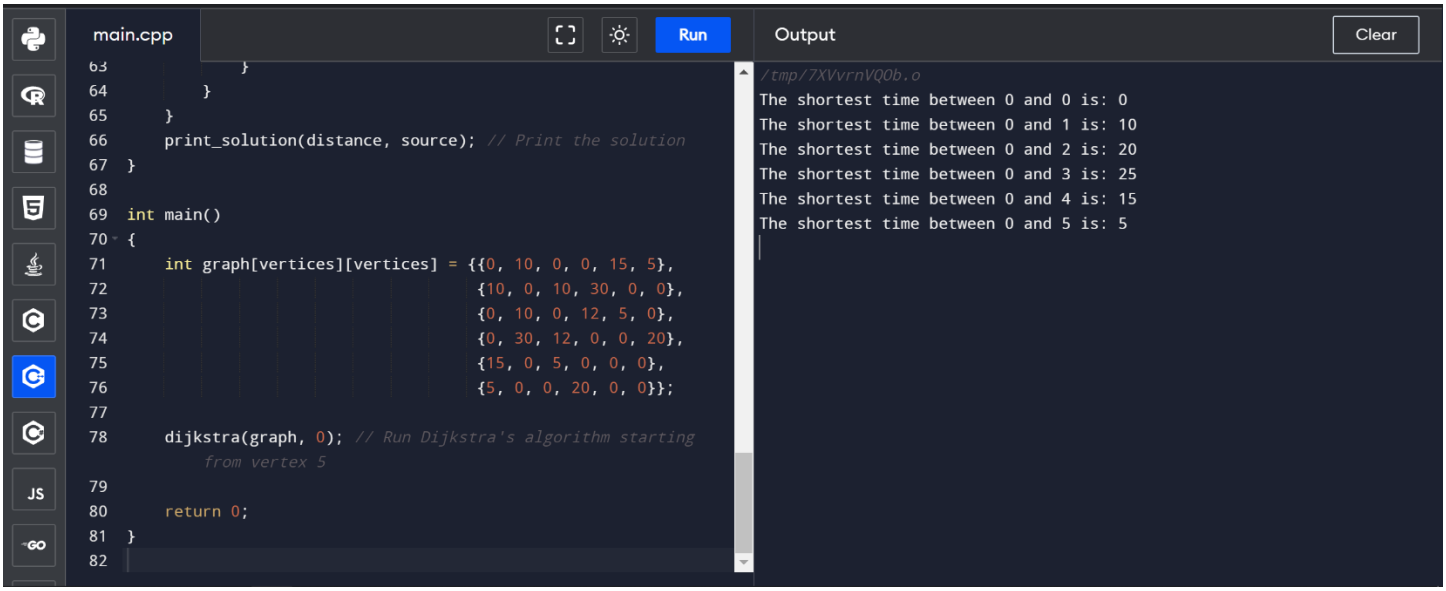
Lab instruction

Here we have 6 cities in a rural area. It has been estimated that for the population of all 6 cities building 1 hospital is sufficient and cost effective. As part of the city planning division, you are tasked to decide which city we must build the hospital in. To do this one must think of a lot of constraints, eg: one can decide to place the hospital at the city with the highest population among the 6 cities. Assume that all the other constraints are equal in each city. You must place the hospital such that the ambulances in the hospital can attend to each city at the shortest time. The weighted undirected graph provided in Fig.1 has cities as nodes and edges as average time taken from each city to another city(where possible not all cities have roads between them). Expected submission,

1. Write the weighted adjacency matrix for the graph on Fig 1.

	0	1	2	3	4	5
0	0	10	0	0	15	5
1	10	0	10	30	0	0
2	0	10	0	12	5	0
3	0	30	12	0	0	20
4	15	0	5	0	0	0
5	5	0	0	20	0	0

2. Implement SSSP Dijkstra's Algorithm
3. By changing the source node in your algorithm take the shortest time between source city to all the other cities. (Eg: if source city 0, then you must calculate the shortest time between City 0 to City 1, City 0 to City 2, City 0 to City 3... etc). Take screenshot of the output for each source city.



main.cpp

```
63              }
64          }
65      }
66      print_solution(distance, source); // Print the solution
67  }
68
69  int main()
70  {
71      int graph[vertices][vertices] = {{0, 10, 0, 0, 15, 5},
72                                       {10, 0, 10, 30, 0, 0},
73                                       {0, 10, 0, 12, 5, 0},
74                                       {0, 30, 12, 0, 0, 20},
75                                       {15, 0, 5, 0, 0, 0},
76                                       {5, 0, 0, 20, 0, 0}};
77
78      dijkstra(graph, 1); // Run Dijkstra's algorithm starting
            from vertex 5
79
80      return 0;
81  }
82
```

Output

```
/tmp/7XVvrnVQOb.o
The shortest time between 1 and 0 is: 10
The shortest time between 1 and 1 is: 0
The shortest time between 1 and 2 is: 10
The shortest time between 1 and 3 is: 22
The shortest time between 1 and 4 is: 15
The shortest time between 1 and 5 is: 15
```

main.cpp

```
63              }
64          }
65      }
66      print_solution(distance, source); // Print the solution
67  }
68
69  int main()
70  {
71      int graph[vertices][vertices] = {{0, 10, 0, 0, 15, 5},
72                                       {10, 0, 10, 30, 0, 0},
73                                       {0, 10, 0, 12, 5, 0},
74                                       {0, 30, 12, 0, 0, 20},
75                                       {15, 0, 5, 0, 0, 0},
76                                       {5, 0, 0, 20, 0, 0}};
77
78      dijkstra(graph, 2); // Run Dijkstra's algorithm starting
            from vertex 5
79
80      return 0;
81  }
82
```

Output

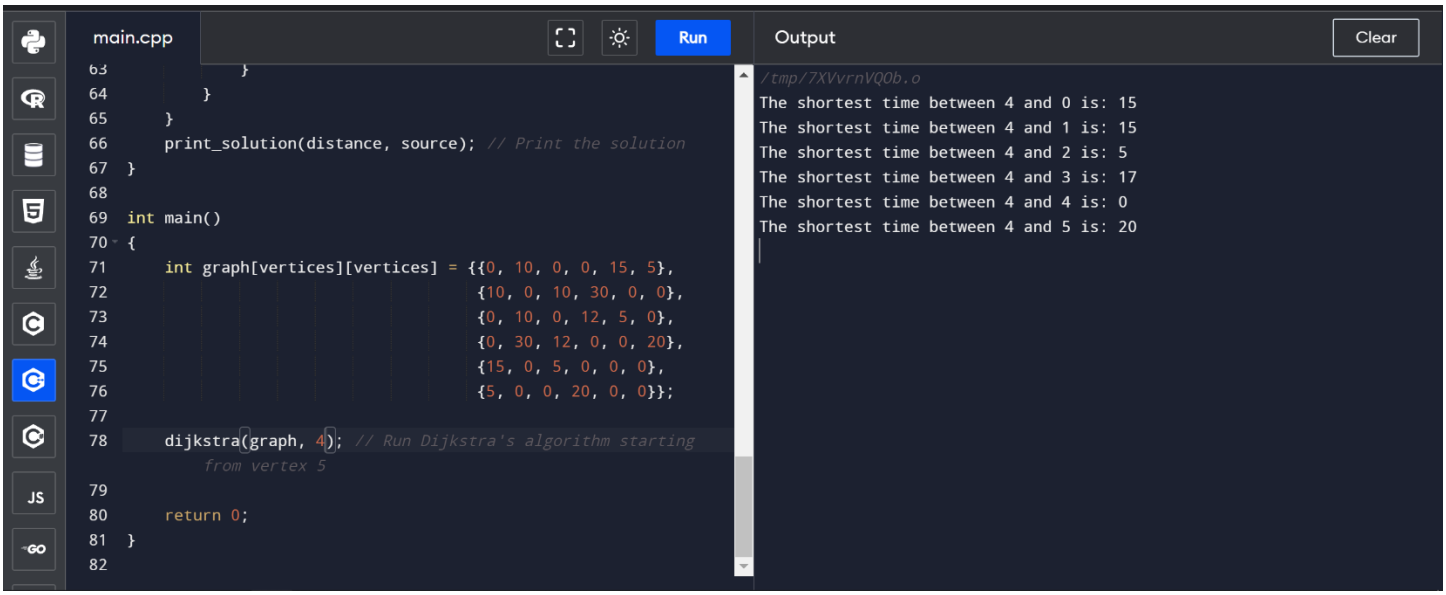
```
/tmp/7XVvrnVQOb.o
The shortest time between 2 and 0 is: 20
The shortest time between 2 and 1 is: 10
The shortest time between 2 and 2 is: 0
The shortest time between 2 and 3 is: 12
The shortest time between 2 and 4 is: 5
The shortest time between 2 and 5 is: 25
```

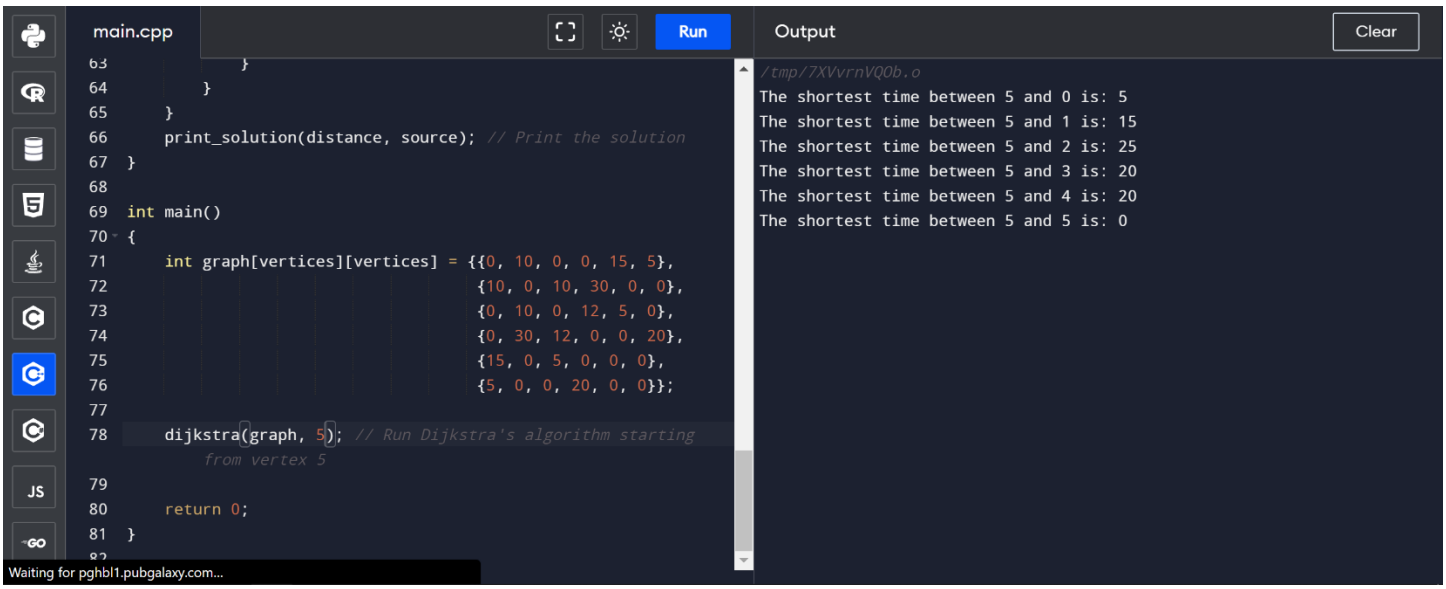
main.cpp

```
63              }
64          }
65      }
66      print_solution(distance, source); // Print the solution
67  }
68
69  int main()
70  {
71      int graph[vertices][vertices] = {{0, 10, 0, 0, 15, 5},
72                                       {10, 0, 10, 30, 0, 0},
73                                       {0, 10, 0, 12, 5, 0},
74                                       {0, 30, 12, 0, 0, 20},
75                                       {15, 0, 5, 0, 0, 0},
76                                       {5, 0, 0, 20, 0, 0}};
77
78      dijkstra(graph, 3); // Run Dijkstra's algorithm starting
            from vertex 5
79
80      return 0;
81  }
82
```

Output

```
/tmp/7XVvrnVQOb.o
The shortest time between 3 and 0 is: 25
The shortest time between 3 and 1 is: 22
The shortest time between 3 and 2 is: 12
The shortest time between 3 and 3 is: 0
The shortest time between 3 and 4 is: 17
The shortest time between 3 and 5 is: 20
```





4. Calculate the average time taken from each source city to the other cities. Pick the city with the smallest average time (if more than 1 city has the same smallest average give all of them)

City 0 -> (10+20+25+15+5)/5 = 15

City 1 -> (10+10+22+15+15)/5 = 14.4

City 2 -> (20+10+12+5+25)/5 = 14.4

City 3 -> (25+22+12+17+20)/5 = 19.2

City 4 -> (15+15+5+17+20)/5 = 14.4

City 5 -> (5+15+25+20+20)/5 = 17