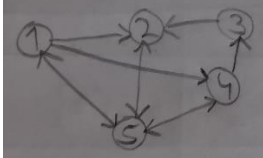
Lowest length path between s and t using breadth-first search from s:



Dictionary/List:

1 - [2, 4, 5]
2 - [5]
3 - [2]
4 - [3, 5]
5 - [1, 2, 4]

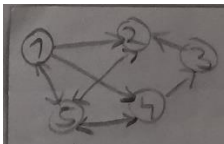
s=1, t=3	currVertex	neighborVertex	Queue	Visited	Dist.	Predecessors dictionary
initialization			[(1,0)]	{1}		{}
iteration 1	1	2 4 5	[(2,1)] [(3,1)][(4,1)] [(2,1)][(4,1)][(5,1)]	{1,2} {1,2,4} {1,2,4,5}	0	{2:1} {2:1, 4:1} {2:1, 4:1, 5:1}
iteration 2	2	5	[(4,1)][(5,1)]	{1,2,4,5}	1	{2:1, 4:1, 5:1}
iteration 3	4	3 5	[(5,1)] [(5,1)][(3,2)]	{1,2,3,4,5}	1	{2:1, 4:1, 5:1, 3:4}
iteration 4	5	1 2 4	[(3,2)]	{1,2,3,4,5}	1	{2:1, 4:1, 5:1, 3:4}
iteration 5	3			{1,2,3,4,5}	2	{2:1, 4:1, 5:1, 3:4}

The reverse path is built from predecessors dict. beginning with t = 3

pred[3] = 4, pred[4] = 1 = s

=> path: 1 4 3

length: 2



Dout

1 - [2, 4, 5]
2 - [5]
3 - [2]
4 - [3, 5]
5 - [1, 2, 4]

s=3, t=5	currVertex	neighbor vertex	Dist	Queue	Visited	Predecessors dict
init.				[(3,0)]	{3}	{}
Iteration 1	3	2	0	[(2,1)]	{3,2}	{2:3}
Iteration 2	2	5	1	[(5,2)]	{3,2,5}	{2:3, 5:2}
Iteration 3	5		2		{3,2,5}	{2:3, 5:2}

pred[5] = 2, pred[2] = 3 = s

=> path: 3 2 5

length: 2

The minimum length paths and their lengths from 1 to 100 and from 100 to 1 in graph1k, graph10k, graph100k as they are determined by your program:

graph1k.txt

Shortest path from 1 to 100: 1 5 487 175 699 624 100; Length of shortest path: 6

Shortest path from 100 to 1: 100 416 354 865 109 1; Length of shortest path: 5

graph10k.txt

Shortest path from 1 to 100: 1 3300 2607 523 6311 5359 9794 5173 100 ; Length of shortest path: 8

Shortest path from 100 to 1: 100 2398 3054 5232 8217 2478 7151 1; Length of shortest path: 7

graph100k.txt

Shortest path from 1 to 100: 1 17024 27471 14969 3075 4156 32753 14973 100; Length of shortest path: 8

Shortest path from 100 to 1: 100 44340 54527 6606 53263 95930 98655 58288 1; Length of shortest path:8