Program results :

minimum path / length from 1 to 100 I
100 to 1 II

Graph 1k

I [1, 5, 487, 175, 699, 624, 100] of length 6

II [100, 416, 354, 865, 109, 1] of length 5

Graph 10k

I [1, 7317, 4118, 2404, 690, 1494, 739, 4722, 100] of length 8

II [100, 5568, 2781, 1451, 4997, 529, 4260, 1] of length 7

Graph 100k

I [1, 17024, 27471, 14969, 3075, 4156, 32753, 14973, 100] of length 8

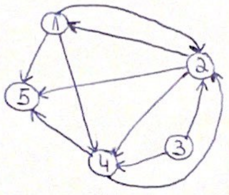
II [100, 44340, 54527, 6606, 53263, 95930, 98655, 58288, 1] of length 8

First example

"find the lowest length path between two given vertices (source, target) by using backward breadth first search"

source vertex : 2
target vertex : 5

Graph :



Din dictionary

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

backwardBreadthFirstSearch 5 [None, None, None, None] [False, False, False, False, True]

start : 5

node	in node	neighbours	queue tail head	previous nodes	visited
5	1	[1, 2, 4]	1	[5, None, None, None, None]	[True, False, False, False, True]
	2		2, 1	[5, 5, None, None, None]	[True, True, False, False, True]
	4		4, 2, 1	[5, 5, None, 5, None]	[True, True, False, True, True]
1	2	[2]	4, 2	— ‖ —	— ‖ —
2	1	[1, 3, 4]	4	— ‖ —	— ‖ —
	3		3, 4	[5, 5, 4, 5, None] None]	[True, True, True, True, True, False]
	4		3, 4	— ‖ —	— ‖ —
4	1	[1, 2, 3]	3	— ‖ —	— ‖ —
	2		3	— ‖ —	— ‖ —
	3		3	— ‖ —	— ‖ —
3		[ ]		— ‖ —	— ‖ —
done!				[5, 5, 4, 5, None]	[True, True, True, True, True]

returns [5, 5, 4, 5, None]

=> the algorithm returns the path [2,5] and the length 1

reconstructPath

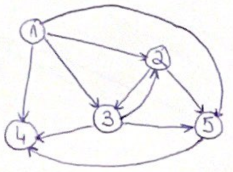
[5,5,4,5,None][illegible] [ ]

start vertex : 2 end vertex : 5

node	length	list of prev. nodes	final path
2	1	[5,5,4,5,None][illegible]	[2]
5	2	[5,5,4,5,None][illegible]	[2,5]
None		— ‖ —	— ‖ —
done!			

Second example

Graph



Din dictionary

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

source vertex : 2
target vertex : 4

backwardBreadthFirstSearch
start : 4

			4	[None, None, None, None]	[False, False, False, False]
node	in node	neighbours	queue tail head	previous nodes	visited
4	1	[1,3,5]	1	[4, None, None, None, None]	[True, False, False, True, False]
	3		3, 1	[4, None, 4, None, None]	[True, False, True, True, False]
	5		5, 3, 1	[4, None, 4, None, 4]	[True, False, True, True, True]
1		[ ]	5, 3	—— ‖ ——	—— ‖ ——
3	1	[1,2]	5	—— ‖ ——	—— ‖ ——
	2		2,5	[4, 3, 4, None, 4]	[True, True, True, True, True]
5	1	[1,2,3]	2	—— ‖ ——	—— ‖ ——
	2		2	—— ‖ ——	—— ‖ ——
	3		2	—— ‖ ——	—— ‖ ——
2	1	[1, 3]		—— ‖ ——	—— ‖ ——
	3			—— ‖ ——	—— ‖ ——
done!				[4,3,4, None, 4]	[True, True, True, True, True]

returns [4,3,4, None, 4]

reconstructedPath [4,3,4,None,4] [ ]

start vertex: 2 end vertex: 4

⇒ the algorithm returns the path [2,3,4] and the length 2

node	length	list of prev nodes	final path
2	1	[4,3,4,None,4]	[2]
3	2	— || —	[2,3]
4	3	— || —	[2,3,4]
None		— || —	— || —

done!