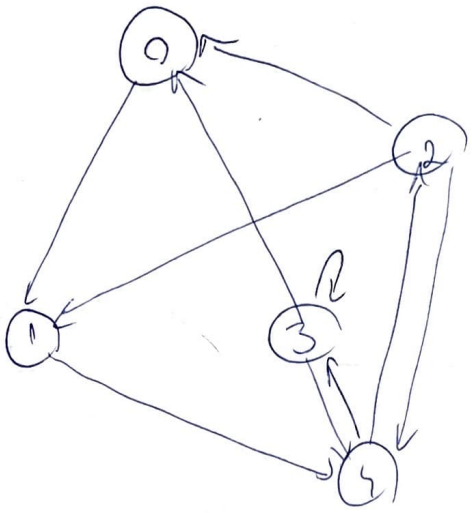
Practical work nr 2

Problem 2 — BFS Backward – lowest length path

start vertex = vertex 1
end vertex = vertex 2



Predecessors dictionary:

0: [2, 3]
1: [0, 2]
2: [4]
3: [3, 4]
4: [1, 2, 3]

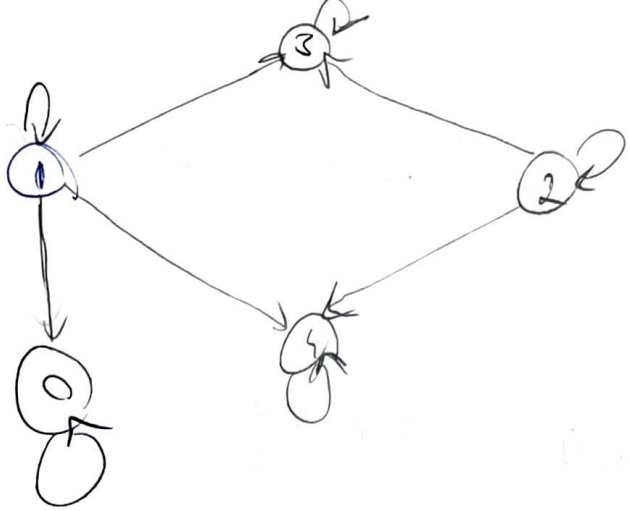
vertex 1 = 1 vertex 2 = 0	current node	~~visited~~ ~~nodes~~ predecessors	visited nodes	queue	children
			{0}	← 0 ←	child[0] = None
initialization			{0}		
iteration 1	0	2	{0, 2}	← 2 ←	child[2] = 0
iteration 1.1.	0	3	{0, 2, 3}	← 2, 3 ←	child[3] = 0
iteration 1.2.	0		{0, 2, 3}	← 3 ←	
iteration 2	2	4	{0, 2, 3, 4}	← 3, 4 ←	child[4] = 2
iteration 2.1	2		{0, 2, 3, 4}	← 4 ←	
iteration 3	3	3	–	–	–
iteration 3.1	3		–	–	–
iteration 3.2	3	4	{0, 2, 3, 4}	–	
iteration 4	4	BREAK	{0, 2, 3, 4, 1}	← 1 ←	child[1] = 4
iteration 4.1.	4				

Path is built like this: ~~path = [ ]~~ ~~node = vertex 1 = 1~~

node = vertex 1 = 1 => path[0] = 1, node = child[1] = 4 => path[1] = 4

~~node = child[4]~~ node = child[4] = 2 => path[2] = 2, node = child[2] = 0

=> path[3] = 0 => path = [1, 4, 2, 0]



Predecessors dictionary:

0: [0, 1]
1: [1]
2: [2]
3: [1, 2, 3]
4: [1, 2, 4]

vertex0 = 3 vertex2 = 4	current node	predecessor	visited	queue	children 0 | 1 | 2 | 3 | 4
initialization			{4}	<- 4 <-	
iteration 1	4		{4}	-	
iteration 1.1.	4	1	{4, 1}	<- 1 <-	| 4 | | |
iteration 1.2.	4	2	{4, 1, 2}	<- 1, 2 <-	| 4 | 4 | |
iteration 1.3.	4	4			
iteration 2	1			<- 2 <-	
iteration 2.1	1	1			
iteration 3	2			<-	
iteration 3.1.	2	2			

iteration 3.2 -> Queue is empty -> path not found between vertex 3 to vertex 4

Graph 1K:
1 - 100 : [1, 5, 437, 175, 698, 624, 100]
100 - 1 : [100, 416, 354, 865, 105, 1]

Graph 10K:
1 - 100 : [1, 7317, 4118, 2404, 630, 1484, 735, 4722, 100]
100 - 1 : [100, 5568, 2701, 1451, 4952, 520, 4260,

Graph cook.txt:

1-100 : [1, 17024, 27471, 11869, 30757, 20733,
85480, 14873, 100]

100-1 : [100, 44340, 54527, 6606, 53263,
95870, 98655, 58288, 1]