**ОТЧЁТ:**

Входной граф:

7 7

1 2

1 3

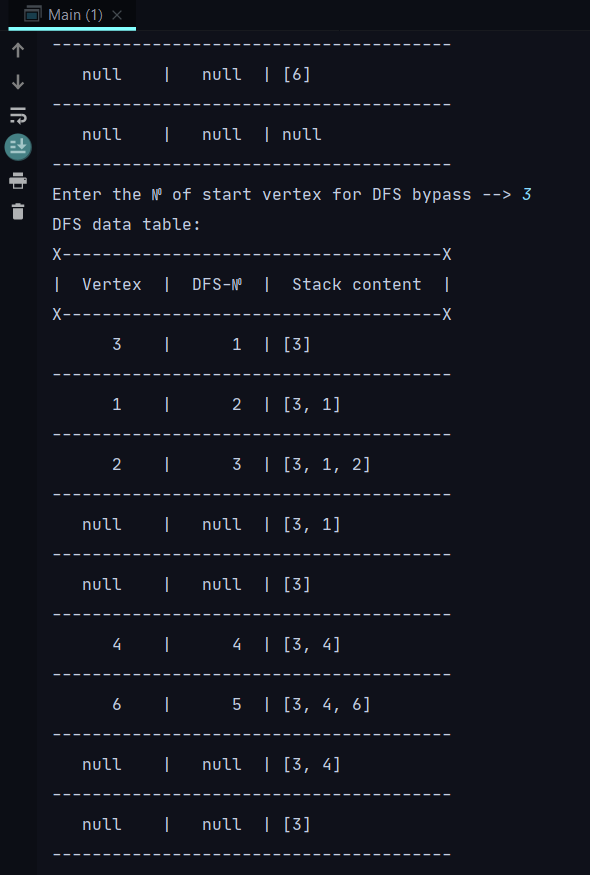
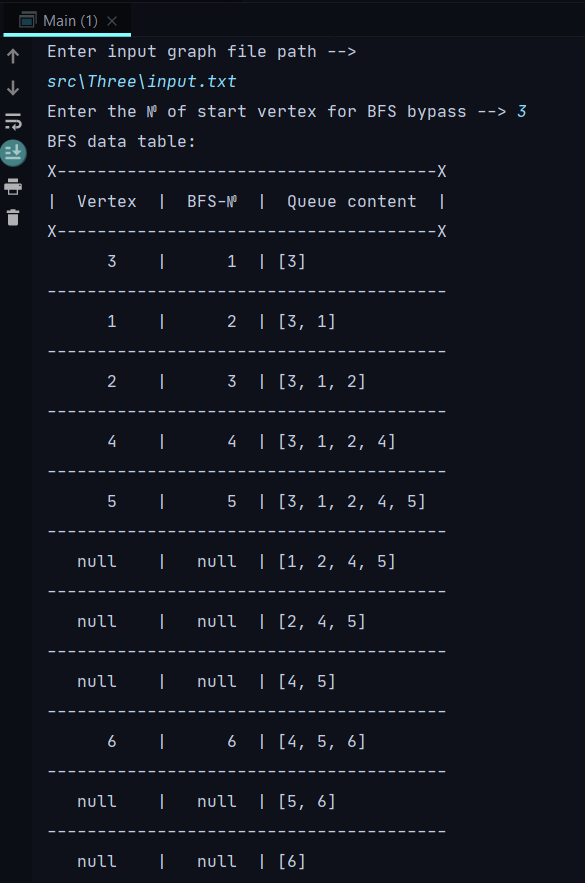
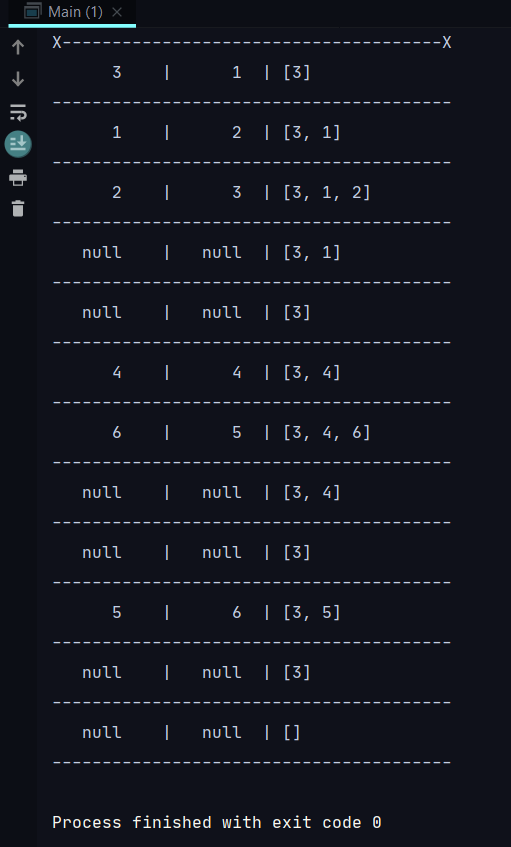
2 3

3 4

3 5

4 6

Результат работы програмы:

­­­­­­­­­­­

Код программы:

*import* java.util.Scanner;  
  
*public class* Main {  
 *public static void* main(String[] args) {  
 Scanner sc = *new* Scanner(System.*in*);  
 System.*out*.println("Enter input graph file path --> ");  
 Graph\_3 g = *new* Graph\_3(sc.nextLine());  
  
 BFS bfs = *new* BFS(g);  
 System.*out*.print("Enter the № of start vertex for BFS bypass --> ");  
 bfs.bypass(sc.nextInt());  
  
 DFS dfs = *new* DFS(g);  
 System.*out*.print("Enter the № of start vertex for DFS bypass --> ");  
 dfs.bypass(sc.nextInt());  
 }  
  
}

*import* java.io.BufferedReader;  
*import* java.io.FileNotFoundException;  
*import* java.io.FileReader;  
*import* java.io.IOException;  
  
*public class* Graph\_3 {  
 *private int*[][] adjMatrix;  
 *private int* v\_num, e\_num;  
 *public* Vertex[] vertices;  
 *public* Graph\_3(String path) {  
 readGraph(path);  
 }  
  
 *private void* readGraph(String path) {  
 *try* {  
 BufferedReader br = *new* BufferedReader(*new* FileReader(path));  
 String[] v\_e\_nums = br.readLine().split(" ");  
 v\_num = Integer.*parseInt*(v\_e\_nums[0]);  
 e\_num = Integer.*parseInt*(v\_e\_nums[1]);  
 adjMatrix = *new int*[v\_num][v\_num];  
 vertices = *new* Vertex[v\_num];  
 *int* start, end;  
 *while* (br.ready()) {  
 String[] u\_v\_vertexes = br.readLine().split(" ");  
 *if*(u\_v\_vertexes.length == 2){  
 start = Integer.*parseInt*(u\_v\_vertexes[0]);  
 end = Integer.*parseInt*(u\_v\_vertexes[1]);  
 adjMatrix[start-1][end-1] = 1;  
 adjMatrix[end-1][start-1] = 1;  
 }  
 }  
 *for* (*int* i = 0; i < v\_num; i++) {  
 vertices[i] = *new* Vertex(i + 1);  
 }  
 } *catch* (FileNotFoundException e) {  
 e.printStackTrace();  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 *public int*[][] getAdjMatrix() {  
 *return* adjMatrix;  
 }  
  
 *public int* getV\_num() {  
 *return* v\_num;  
 }  
  
  
 *public class* Vertex{  
 *private int* number;  
 *private int* bfs\_number = 0;  
 *private int* dfs\_number = 0;  
  
 *public int* getDfs\_number() {  
 *return* dfs\_number;  
 }  
  
 *public void* setDfs\_number(*int* dfs\_number) {  
 *this*.dfs\_number = dfs\_number;  
 }  
  
 *private* Vertex(*int* num) {  
 *this*.number = num;  
 }  
 *public int* getNumber() {  
 *return* number;  
 }  
  
 *public int* getBfs\_number() {  
 *return* bfs\_number;  
 }  
  
 *public void* setBfs\_number(*int* bfs\_number) {  
 *this*.bfs\_number = bfs\_number;  
 }  
 }  
}

*import* java.util.LinkedList;  
  
*public class* BFS {  
 *private* Graph\_3 g;  
 *private* LinkedList<Integer> queue;  
  
 *public* BFS(Graph\_3 g) {  
 *this*.g = g;  
 queue = *new* LinkedList<>();  
 }  
  
 *public void* bypass(*int* start\_vertex) {  
 Graph\_3.Vertex start\_v = g.vertices[start\_vertex-1];  
 *int*[][] adjM = g.getAdjMatrix().clone();  
 *int* bfs\_n = 1;  
 start\_v.setBfs\_number(bfs\_n);  
 queue.add(start\_v.getNumber());  
 printVertexData(start\_v.getNumber(),bfs\_n, queue);  
 bfs\_n++;  
 *int* head = queue.peek();  
 *while*(queue.peek() != *null*) {  
 *for* (*int* i = 0; i < g.getV\_num(); i++){  
 *if* (adjM[head-1][i] != 0 && g.vertices[i].getBfs\_number() == 0) {  
 queue.add(i+1);  
 g.vertices[i].setBfs\_number(bfs\_n);  
 printVertexData(g.vertices[i].getNumber(), bfs\_n, queue);  
 bfs\_n++;  
 }  
 }  
 queue.removeFirst();  
 *if*(queue.peek() != *null*){  
 printVertexData(*null*,*null*, queue);  
 head = queue.peek();  
 }  
 *else* printVertexData(*null*,*null*,*null*);  
 }  
 }  
  
 *private void* printVertexData(Integer v\_number, Integer bfs\_n, LinkedList<Integer> q) {  
 *if* (bfs\_n != *null* && bfs\_n == 1) {  
 System.*out*.println("BFS data table:");  
 System.*out*.println("X--------------------------------------X");  
 System.*out*.println("| Vertex | BFS-№ | Queue content |");  
 System.*out*.println("X--------------------------------------X");  
 }  
 System.*out*.printf(" %4d | %4d |", v\_number, bfs\_n);  
 System.*out*.println(" " + q);  
 System.*out*.println("----------------------------------------");  
 }  
  
}

*import* java.util.LinkedList;  
  
*public class* DFS {  
 *private* Graph\_3 g;  
 *private* LinkedList<Integer> stack;  
  
 *public* DFS(Graph\_3 g) {  
 *this*.g = g;  
 stack = *new* LinkedList<>();  
 }  
  
 *public void* bypass(*int* start\_vertex) {  
 Graph\_3.Vertex focus\_v = g.vertices[start\_vertex-1];  
 *int*[][] adjM = g.getAdjMatrix().clone();  
 *int* dfs\_n = 1;  
 focus\_v.setDfs\_number(dfs\_n);  
 stack.add(focus\_v.getNumber());  
 printVertexData(focus\_v.getNumber(),dfs\_n, stack);  
 *while* (stack.peekLast() != *null*) {  
 *for*(*int* i = 0; i < g.getV\_num(); i++){  
 *if* (adjM[focus\_v.getNumber() - 1][i] != 0 && g.vertices[i].getDfs\_number() == 0) {  
 dfs\_n++;  
 stack.add(i+1);  
 focus\_v = g.vertices[i];  
 focus\_v.setDfs\_number(dfs\_n);  
 printVertexData(focus\_v.getNumber(), dfs\_n, stack);  
 i = 0;  
 }  
 }  
 *if* (stack.peekLast() != *null*) {  
 stack.removeLast();  
 printVertexData(*null*, *null*, stack);  
 *if* (stack.peekLast() != *null*) focus\_v = g.vertices[stack.peekLast() - 1];  
 }  
 }  
 }  
  
 *private void* printVertexData(Integer v\_number, Integer dfs\_n, LinkedList<Integer> s) {  
 *if* (dfs\_n != *null* && dfs\_n == 1) {  
 System.*out*.println("DFS data table:");  
 System.*out*.println("X--------------------------------------X");  
 System.*out*.println("| Vertex | DFS-№ | Stack content |");  
 System.*out*.println("X--------------------------------------X");  
 }  
 System.*out*.printf(" %4d | %4d |", v\_number, dfs\_n);  
 System.*out*.println(" " + s);  
 System.*out*.println("----------------------------------------");  
 }  
  
}

­­