**Graph: Import, export and basic operations on graphs**

**Objectives: After completing this exercise students can:**

* Create a text file containing a graph in the form: adjacency matrix/adjacency list/edge list.
* Build three types of graph storage data structures: adjacency matrix, adjacency list, edge list.
* Write a program: Import/Export text files containing graphs.
* Write a program to perform basic operations on graphs such as: Read the vertices of the graph, check if two vertices are adjacent, Read the adjacent vertices of a vertex, calculate the degree of the vertex

----------------------------------------------------------------------------------------------

**Practice 1**: Degree of an undirected graph

Given an undirected graph 𝐺 = (𝑉, 𝐸) with 𝑛 (𝑛 ≤ 1000) vertices, the vertices are numbered from 1 to 𝑛. The graph 𝐺 is stored in a text file as an adjacency matrix. Organize an adjacency matrix data structure to represent the graph, and write a program to read the graph 𝐺 from the given file into that data structure, then calculate the degree of the vertices in the graph (the degree of a vertex is the number of edges attached to that vertex).

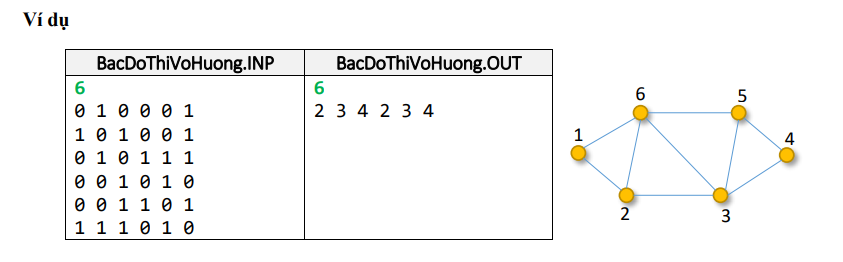
Input data: Text file bacdothivohuong.in

* The first line contains the integer 𝑛 which is the number of vertices in the graph.
* The next 𝑛 lines each contain 𝑛 numbers representing the adjacency matrix of the graph.

Output data: bacdothivohuong.out

* The first line contains the number 𝑛 which is the number of vertices in the graph.
* The second line contains 𝑛 integers corresponding to the degrees of vertices 1, 2, …,𝑛

(Numbers on the same line, separated by 1 space)



public class *Graph* {  
 public static void *main*(*String*[] *args*) {  
 *String* inputFileName = "src/Graphs/data.txt";  
 *String* outputFileName = "src/Graphs/out.txt";  
  
 try {  
 *BufferedReader* br = new *BufferedReader*(new *FileReader*(inputFileName));  
 *BufferedWriter* bw = new *BufferedWriter*(new *FileWriter*(outputFileName));  
  
 *// Read the number of vertices  
 String* firstLine = br.*readLine*().*trim*();  
 int n;  
 try {  
 n = *Integer*.*parseInt*(firstLine);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The first line of the input file must be an integer representing the number of vertices.");  
 }  
  
 *// Initialize the adjacency matrix* int[][] adjacencyMatrix = new int[n][n];  
  
 *// Read the adjacency matrix from the input file* for (int i = 0; i < n; i++) {  
 *String* line = br.*readLine*();  
 if (line == null) {  
 throw new *IllegalArgumentException*("The input file does not contain the correct number of lines for the adjacency matrix.");  
 }  
  
 *String*[] numbers = line.*trim*().*split*("\\s+");  
 if (numbers.length != n) {  
 throw new *IllegalArgumentException*("Each line of the adjacency matrix must contain " + n + " integers.");  
 }  
  
 for (int j = 0; j < n; j++) {  
 try {  
 adjacencyMatrix[i][j] = *Integer*.*parseInt*(numbers[j]);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The adjacency matrix must contain only integers.");  
 }  
 }  
 }  
  
 *// Calculate the degree of each vertex* int[] degrees = new int[n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 degrees[i] += adjacencyMatrix[i][j];  
 }  
 }  
  
 *// Write the output* bw.*write*(n + "\n");  
 for (int i = 0; i < n; i++) {  
 bw.*write*(degrees[i] + (i == n - 1 ? "" : " "));  
 }  
  
 *// Close the readers and writers* br.*close*();  
 bw.*close*();  
 } catch (*IOException e*) {  
 *e*.*printStackTrace*();  
 }  
 }  
}

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Practice 2: Entry Degree, Output degree**

Given a directed graph 𝐺 = (𝑉, 𝐸) with 𝑛 vertices numbered from 1 to 𝑛. The entry degree of vertex 𝑖 is the number of incoming arcs at vertex 𝑖. The output degree of vertex 𝑖 is the number of arcs going out of vertex 𝑖. Calculate the input and output degrees of all vertices in the graph town.

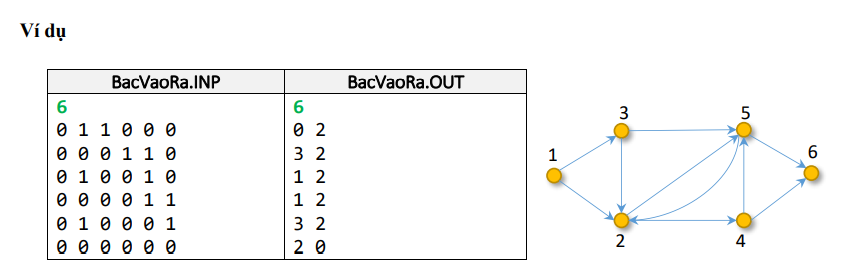
Input data: BacVaoRa.INP text file

* The first line contains the integer 𝑛 (𝑛 ≤ 1000) which is the number of vertices in the graph.
* The next 𝑛 lines each contain 𝑛 numbers representing the adjacency matrix of the graph.

Output data: BacVaoRa.OUT text file

* The first line is a positive integer 𝑛 is the number of vertices in the graph.
* The next 𝑛 line, the 𝑖th line, includes two integers, the input and output steps of vertex 𝑖

(Numbers on the same line, separated by 1 space)



public class *GraphDegrees* {  
 public static void *main*(*String*[] *args*) {  
 *String* inputFileName = "src/Graphs/BacVaoRa\_Inp.txt";  
 *String* outputFileName = "src/Graphs/BacVaoRa\_Out.txt";  
  
 try {  
 *BufferedReader* br = new *BufferedReader*(new *FileReader*(inputFileName));  
 *BufferedWriter* bw = new *BufferedWriter*(new *FileWriter*(outputFileName));  
  
 *// Read the number of vertices  
 String* firstLine = br.*readLine*().*trim*();  
 int n;  
 try {  
 *System*.out.*println*("Reading number of vertices: " + firstLine); *// Debugging output* n = *Integer*.*parseInt*(firstLine);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The first line of the input file must be an integer representing the number of vertices.");  
 }  
  
 *// Initialize the adjacency matrix* int[][] adjacencyMatrix = new int[n][n];  
  
 *// Read the adjacency matrix from the input file* for (int i = 0; i < n; i++) {  
 *String* line = br.*readLine*();  
 if (line == null) {  
 throw new *IllegalArgumentException*("The input file does not contain the correct number of lines for the adjacency matrix.");  
 }  
  
 *String*[] numbers = line.*trim*().*split*("\\s+");  
 *System*.out.*println*("Reading line " + (i + 1) + ": " + line); *// Debugging output* if (numbers.length != n) {  
 throw new *IllegalArgumentException*("Each line of the adjacency matrix must contain " + n + " integers.");  
 }  
  
 for (int j = 0; j < n; j++) {  
 try {  
 adjacencyMatrix[i][j] = *Integer*.*parseInt*(numbers[j]);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The adjacency matrix must contain only integers.");  
 }  
 }  
 }  
  
 *// Calculate the in-degree and out-degree of each vertex* int[] inDegrees = new int[n];  
 int[] outDegrees = new int[n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 if (adjacencyMatrix[i][j] == 1) {  
 outDegrees[i]++;  
 inDegrees[j]++;  
 }  
 }  
 }  
  
 *// Write the output* bw.*write*(n + "\n");  
 for (int i = 0; i < n; i++) {  
 bw.*write*(inDegrees[i] + " " + outDegrees[i] + "\n");  
 }  
  
 *// Close the readers and writers* br.*close*();  
 bw.*close*();  
 } catch (*IOException e*) {  
 *e*.*printStackTrace*();  
 }  
 }  
}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Practice 3: Adjacency list**

Given an undirected graph 𝐺 = (𝑉, 𝐸) with 𝑛 (𝒏 ≤ 𝟏𝟎𝟓) vertices, vertices are numbered from 1 to 𝑛 and 𝑚 edges (𝑚 ≤ 105).

Organize the data structure for the graph as an adjacency list, and write a program to read the graph 𝐺 from the given file, then calculate the degree of the vertices in the graph.

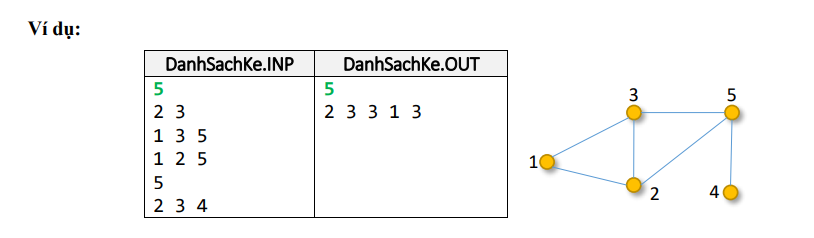
Input data: DanhSachKe.INP text file

* The first line contains the number of vertices 𝑛 of the graph.
* The next 𝑛, 𝑖th line contains a list of vertices, each vertex 𝑗 in the corresponding list with an edge (𝑖,𝑗) of the graph.

Note: If a vertex is isolated (a vertex not connected to other vertices), that line is empty

Output data: Text file DanhSachKe.OUT

* The first line contains the number 𝑛 which is the number of vertices in the graph
* The second line contains 𝑛 integers corresponding to the degrees of vertices 1, 2, … , n



public class *AdjacencyList* {  
 public static void *main*(*String*[] *args*) {  
 *String* inputFileName = "src/Graphs/DanhSachKe\_Inp.txt";  
 *String* outputFileName = "src/Graphs/DanhSachKe\_Out.txt";  
  
 try {  
 *BufferedReader* br = new *BufferedReader*(new *FileReader*(inputFileName));  
 *BufferedWriter* bw = new *BufferedWriter*(new *FileWriter*(outputFileName));  
  
 *// Read the number of vertices  
 String* firstLine = br.*readLine*().*trim*();  
 int n;  
 try {  
 n = *Integer*.*parseInt*(firstLine);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The first line of the input file must be an integer representing the number of vertices.");  
 }  
  
 *// Initialize the degree array* int[] degrees = new int[n];  
  
 *// Read the adjacency list from the input file* for (int i = 0; i < n; i++) {  
 *String* line = br.*readLine*();  
 if (line != null && !line.*trim*().*isEmpty*()) {  
 *String*[] neighbors = line.*trim*().*split*("\\s+");  
 degrees[i] += neighbors.length; *// Count the neighbors for vertex i* }  
 }  
  
 *// Write the output* bw.*write*(n + "\n");  
 for (int i = 0; i < n; i++) {  
 bw.*write*(degrees[i] + (i == n - 1 ? "" : " "));  
 }  
  
 *// Close the readers and writers* br.*close*();  
 bw.*close*();  
 } catch (*IOException e*) {  
 *e*.*printStackTrace*();  
 }  
 }  
}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Practice 4: Edge List**

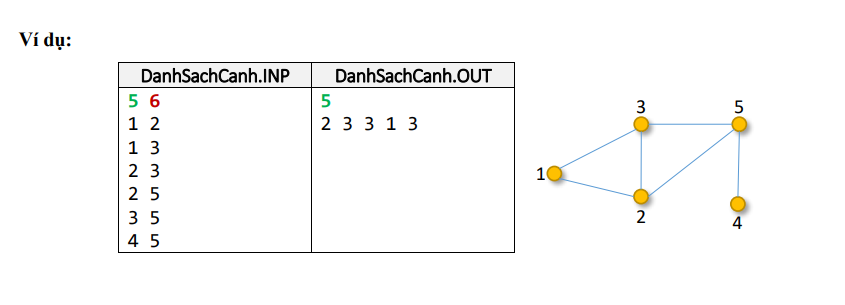
Given an undirected graph 𝐺 = (𝑉, 𝐸) with 𝑛 (𝒏 ≤ 𝟏𝟎𝟓) vertices, vertices are numbered from 1 to 𝑛 and 𝑚 edges (𝑚 ≤ 105) are represented as an edge list in a text file copy. Let's organize the data structure for the graph in the form of an edge list, and write a program to read the graph 𝐺 from the given file, then calculate the degree of the vertices in the graph.

Input data: Text file DanhSachCanh.INP

* The first line contains two integers 𝑛, 𝑚 are the number of vertices and edges of the graph.
* 𝑚 next lines, each line is a pair of numbers representing an edge of the graph (the numbers are separated by 1 space)

Output data: Text file DanhSachCanh.OUT

* The first line is a positive integer 𝑛 is the number of vertices in the graph.
* The second line contains 𝑛 integers corresponding to the degree of vertices 1, 2, …,n



public class *EdgeList* {  
 public static void *main*(*String*[] *args*) {  
 *String* inputFileName = "src/Graphs/DanhSachCanh\_Inp.txt";  
 *String* outputFileName = "src/Graphs/DanhSachCanh\_Out.txt";  
  
 try {  
 *BufferedReader* br = new *BufferedReader*(new *FileReader*(inputFileName));  
 *BufferedWriter* bw = new *BufferedWriter*(new *FileWriter*(outputFileName));  
  
 *// Read the number of vertices and edges  
 String* firstLine = br.*readLine*().*trim*();  
 int n, m;  
 try {  
 *String*[] firstLineParts = firstLine.*split*("\\s+");  
 n = *Integer*.*parseInt*(firstLineParts[0]);  
 m = *Integer*.*parseInt*(firstLineParts[1]);  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("The first line of the input file must contain two integers representing the number of vertices and edges.");  
 }  
  
 *// Initialize the degree array* int[] degrees = new int[n];  
  
 *// Read the edge list from the input file* for (int i = 0; i < m; i++) {  
 *String* line = br.*readLine*();  
 if (line != null && !line.*trim*().*isEmpty*()) {  
 *String*[] edge = line.*trim*().*split*("\\s+");  
 int u, v;  
 try {  
 u = *Integer*.*parseInt*(edge[0]) - 1;  
 v = *Integer*.*parseInt*(edge[1]) - 1;  
 } catch (*NumberFormatException e*) {  
 throw new *IllegalArgumentException*("Each edge must contain two integers.");  
 }  
 if (u < 0 || u >= n || v < 0 || v >= n) {  
 throw new *IllegalArgumentException*("Vertex index out of bounds.");  
 }  
 degrees[u]++;  
 degrees[v]++;  
 }  
 }  
  
 *// Write the output* bw.*write*(n + "\n");  
 for (int i = 0; i < n; i++) {  
 bw.*write*(degrees[i] + (i == n - 1 ? "" : " "));  
 }  
  
 *// Close the readers and writers* br.*close*();  
 bw.*close*();  
 } catch (*IOException e*) {  
 *e*.*printStackTrace*();  
 }  
 }  
}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated