# CSE 246: Project on applying Algorithm in various problems
## Marks Distribution

**Project Demonstration (Marks: 4)**

i. Compilation of code without error (Documentation, comment and indentation in the source code must be maintained)
ii. Testing the project with a different set of inputs.
Iii. A set of validation inputs which has been used to test the solution.

**Report (Marks: 7)**

1. **Problem Statement:**
   Write a brief statement of the project in general terms according to the problem description provided. Problem statement should not include any example or detailed description.
2. **Algorithm Discussion and simulation through examples**
   A detailed discussion about the algorithm or approach to solve the problem in your own language along with pseudo code / step by step flow and examples.
3. **Complexity Analysis**
   A detailed formal discussion about the designed algorithm's runtime and memory complexity.
4. **Implementation with proper commenting**
   Proper commenting of the code to indicate a code block's purpose and what it is doing with proper indentation. Focus on important parts of the code.
5. **Applications (Bonus)**
   If you can present any real application scenario that matches with the problem's goal you will get some additional bonus.

**Presentation (Marks: 2)**

i. Content of your presentation
ii. Presentation skill

**Submission Manual:**

- You need to make a folder, where you will provide your code(.c, .cpp, .java are only allowed), a word file containing all the codes and the complete report(pdf). Then, rename the folder with your complete ID, zip it in the same format and submit it in the allotted classroom post. For each group, there will be a single submission.

The Maximum time limit for each presentation will be 7 minutes at most. Questions can be asked of anyone in the group. If any group member fails to provide the answer, it might penalize the whole group.

**N.B: Any kind of extreme plagiarism if detected will be severely penalized.**

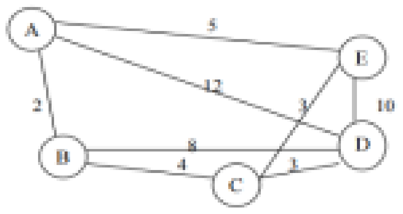
# Problems' Descriptions

## Problem ID: 1

**Project: Traveling Salesman Problem**

**Difficulty: Medium**

A traveler needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited just once. What is the shortest possible route that he visits each city exactly once and returns to the origin city? Given a set of cities and the cost of travel (or distance) between each possible pair, you have to find the best possible way of visiting all the cities and returning to the starting point that minimizes the travel cost (or travel distance). Consider the following set of cities:



The problem lies in finding a minimal path passing from all vertices once. For example, the path Path1 {A, B, C, D, E, A} and the path Path2 {A, B, C, E, D, A} pass all the vertices but Path1 has a total length of 24 and Path2 has a total length of 31.

In this project, you have to write a program that solves the traveling salesman problem and shows the best path along with the minimal cost. You should use **Dynamic Programming** or a recursive algorithm based on **Backtracking** approach in this regard as they have the ability to give the same result in far fewer attempts than Brute Force method trials and thus handles the complexity better.

**Input Format:**

In the first line you will be given an integer N denoting the number of cities and another integer E denoting the number of roads. In the following E lines you will be given information about the roads. Each such line will contain three integers u, v and w which indicates that there is a road between city u and v and the cost of traversing that edge is w.

**Output Format:**

For each test case, your program will print two pieces of information. In the first line, it will print the minimum cost and in the following line a valid path that bears that cost satisfying the given constraints of the problem. As the path you will print the nodes according to their traversal order.

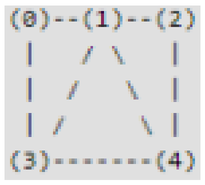
## Problem ID: 2

**Project: Hamiltonian Cycle**

**Difficulty: Medium**

A Hamiltonian cycle is a closed loop on a graph where every node is visited exactly once. It is a Hamiltonian Path such that there is an edge from the last vertex to the first vertex of the path. Determine whether a given graph contains Hamiltonian Cycle or not. If it contains, then what is the path?

For example, a Hamiltonian Cycle in the following graph is {0, 1, 2, 4, 3, 0}. There are more Hamiltonian Cycles in the graph like {0, 3, 4, 2, 1, 0}.



```
(0)--(1)--(2)
 |   / \   |
 |  /   \  |
 | /     \ |
(3)-------(4)
```

In this project, you have to design a suitable algorithm that finds all the Hamiltonian cycles present in a specific graph and the Hamiltonian path accordingly. You should use **Dynamic Programming** or a recursive algorithm based on **Backtracking** approach in this regard as they have the ability to give the same result in far fewer attempts than Brute Force method trials and thus handles the complexity better.

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges. Each such line will contain two integers u and v which indicates that there is an undirected edge between nodes u and v.

**Output Format:**

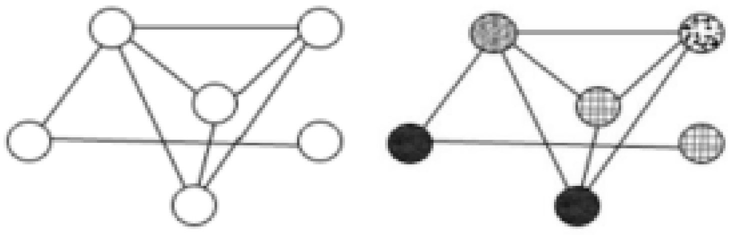
If the given graph contains at least one hamiltonian cycle, you need to print two lines of output for each test case. In the first line, you need to print a string "YES" and in the second line the set of nodes according to their traversal order that satisfies the given constraint. But if the provided graph does not contain any hamiltonian cycle you need to print a single string "NO".

## Problem ID: 3

**Project: Graph Coloring Problem**

**Difficulty: Easy**

Graph Coloring problem is to assign colors to certain elements of a graph subject to certain constraints. The problem is, given an undirected graph and a number m, determine if the graph can be coloured with at most m colors such that no two adjacent vertices of the graph are coloured with the same color. Here coloring of a graph means assignment of colors to all vertices. For example,



**Non-coloured Coloured**

The idea is to assign colors one by one to different vertices, starting from the vertex 0. Before assigning a color, check for safety by considering already assigned colors to the adjacent vertices.

In this project, you have to design a suitable algorithm that solves the Graph Coloring problem. Your solution will be able to detect if the given graph can be colored using at most m colors.

You should use a **Greedy** approach or a recursive algorithm based on **Backtracking** approach in this regard as they have the ability to give same result in far fewer attempts than Brute Force method trials and thus handles the complexity better

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges.

Each such line will contain two integers u and v which indicates that there is an undirected edge between nodes u and v.

**Output Format:**

If the given graph satisfies the coloring constraint (can be colored using at most m colors where the adjacent nodes always bear different colors) you need to print “YES”. Otherwise, you will print “NO”.

## Problem ID: 4

**Project: N-Queen Problem**

**Difficulty: Medium**

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, to solve the eight queens puzzle (when N=8), the problem is to place the eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. To solve the problem, keep a simple rule: last placed, first displaced. In other words, if you successfully place a queen on the $i^{th}$ column but cannot find a solution for $(i+1)^{th}$ queen, then going backwards you will try to find other admissible solution for the $i^{th}$ queen first. This is a Backtrack approach where you try to build a solution one step at a time. If at some step it becomes clear that the current path that you are on cannot lead to a solution, you go back to the previous step (backtrack) and choose a different path. Briefly, once you exhaust all your options at a certain step you go back.

In this project, you have designed a suitable algorithm that solves the N-queen problem and shows all possible placements of N queens. You have to use a Backtracking algorithm in this regard. You should not use **Brute Force** as using it will increase the runtime and complexity with the increasing value of N.

**Input Format:**

Number of queens or the value of N.

**Output Format:**

All possible placements of the queens. You can show it using matrix. For each possible arrangement it will be a (8 x 8) output containing 0 or 1. 1 at a particular position indicates that, in this arrangement there lies a queen where 0 means the absence of queen in that position.

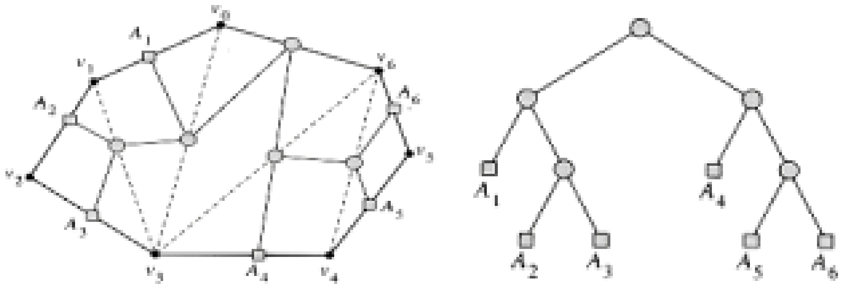
## Problem ID: 5

**Project: Optimal Polygon Triangulation**

**Difficulty: Hard**

A polygon is a two-dimensional closed shape defined by connections between points or vertices.A triangulation of a polygon can be thought of as a set of chords that divide the polygon into triangles such that no two chords intersect (except possibly at a vertex). It can be formed by drawing diagonals between non-adjacent vertices such that the diagonals never intersect. The cost of a triangulation is sum of the weights of its component triangles. Weight of each triangle is its perimeter (sum of lengths of all sides). An optimal triangulation is one that minimizes some cost function of the triangles.

The idea is to divide the polygon into three parts: a single triangle, the sub-polygon to the left, and the sub-polygon to the right. Try all possible divisions like this until you find the one that minimizes the cost of the triangle plus the cost of the triangulation of the two sub-polygons. You can get the cost of triangulation of the two sub-polygons recursively. The base case of the recursion is a line segment (i.e., a polygon with zero area), which has cost 0.



In this project, you have to design a suitable algorithm implementing the Optimal Polygon Triangulation. Show the triangulations along with the minimal cost. You should use **Dynamic Programming** in this regard. Avoid using **Brute Force** as it may take exponential time, while dynamic programming is much faster in comparison.

**Input Format:**

In the first line, you will be given N denoting the number of points or co-ordinates the polygon has. In the following N lines you will be given information about the points in each line having two coordinates, $x_i$ and $y_i$ ,1<=i<=N. You can construct the polygon by connecting points ($x_i$ , $y_i$) and ($x_{i+1}$ , $y_{i+1}$). By connecting the last two coordinate points, ($x_{N-1}$ , $y_{N-1}$) and ($x_1$ , $y_1$) you can complete the polygon.

**Output Format:**

For each test case, there will be a single line of output printing the minimum cost to conduct optimal polygon triangulation for the given polygon.

## Problem ID: 6

**Project: Optimal Binary Search Tree**

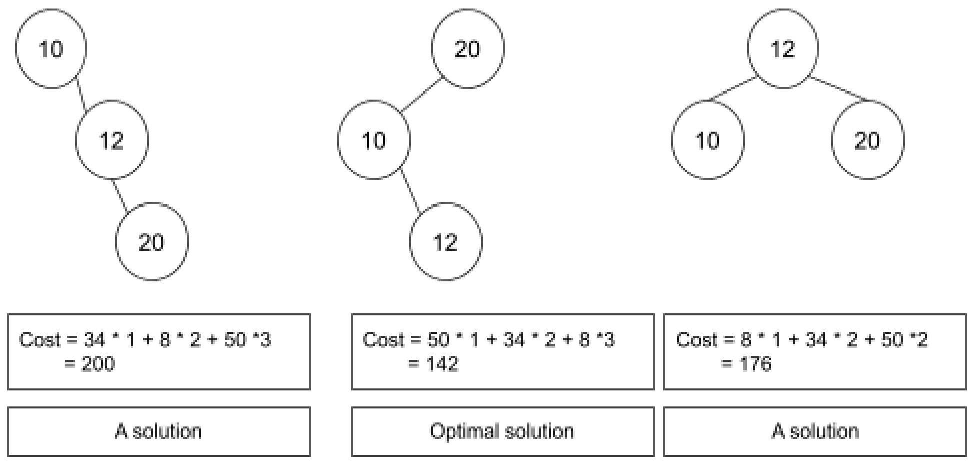
**Difficulty: Hard**

Crispher is a very good coder. After reading anything he tries to implement it in code. About a week ago he was reading about Optimal Binary Search tree. Now he wants to implement it in the code but he finds some problem to do it. His solution was greedy as well as complexity was high. So he wants someone who has good knowledge in Algorithms and knows how to minimize complexity. He stated the problem as below:

Given a sorted array keys [0... n-1] of search keys and an array freq[0... n-1] of frequency counts, where freq[i] is the number of searches to keys[i]. Construct an optimal binary search tree (BST) of all keys such that the total cost of all the searches is as small as possible. For example:

The cost of a BST is the summation of all the nodes' cost where each node's cost denotes its level multiplied by its frequency.

```
Input: keys[] = {10, 12, 20}, freq[] = {34, 8, 50}
```


10
12
20
Cost = 34 * 1 + 8 * 2 + 50 *3
= 200
A solution
20
10
12
Cost = 50 * 1 + 34 * 2 + 8 *3
= 142
Optimal solution
12
10
20
Cost = 8 * 1 + 34 * 2 + 50 *2
= 176
A solution

Use a Dynamic Programming (DP) based approach to find the optimal solution of the above stated problem. Keep in mind that Brute Force solution is not allowed.

**Input Format:**

In the first line you will be given an integer N denoting the number of keys. In the first following line you will get N sorted values denoting the keys and in the second following line, you will be given N values denoting the frequency of each key.

**Output Format:**

For each test case there will be a single output denoting the minimum cost of the constructed optimal binary search tree from the given input set.

## Problem ID: 7

**Project: Johnson's Algorithm**

**Difficulty: Medium**

Johnson's algorithm is a way to find the shortest paths between all pairs of vertices in a sparse, edge-weighted, directed graph. It allows some of the edge weights to be negative numbers, but no negative-weight cycles may exist. It works by using the Bellman–Ford algorithm to compute a transformation of the input graph that removes all negative weights, allowing Dijkstra's algorithm to be used on the transformed graph.

In this project, you have to v that finds shortest paths between every pair of vertices in a given weighted directed Graph where weights may be negative as well using by implementing **Johnson's Algorithm**. You should not use Brute Force as it may increase the complexity and runtime of the program.

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the directed edges. Each such line will contain three integers u, v and w which indicates that there is a directed from u to v and the cost of traversing that edge is w. The graph may contain negative weights for edges.

**Output Format:**

If the graph contains negative cycle(s) the program will print a single line "Negative cycle exists". Otherwise the graph will print all the shortest path distances in the form of u, v, w denoting the minimum cost to reach v from u is w. The printing will maintain lexicographic order of the nodes, e.g, first distances from node 1, then node 2, then node 3 and so on.

## Problem ID: 8

**Project: Sequence Alignment Problem**

**Difficulty: Medium**

In bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. It is a fundamental problem in Biological Science.

Given as an input two strings, $X = x_1 x_2....x_m$, and $Y = y_1 y_2....y_m$, output the alignment of the strings, character by character, so that the net penalty is minimized. The penalty is calculated as:

1. A penalty of $P_{gap}$ occurs if a gap is inserted between the strings.
2. A penalty of $P_{xy}$ occurs for miss-matching the characters of X and Y.

**For Example**:

**Test Case 1:** Input: X = CG, Y = CA, p_gap = 3, p_xy = 7

Output: X = CG_, Y = C_A, Total penalty = 6

**Test Case 2:** Input: X = AGGGCT, Y = AGGCA, p_gap = 3, p_xy = 2

Output: X = AGGGCT, Y = A_GGCA, Total penalty = 5

**Test Case 3:** Input: X = CG, Y = CA, p_gap = 3, p_xy = 5

Output: X = CG, Y = CA, Total penalty = 5

Write a program to find the maximum match of the given strings. Use Dynamic programming to solve the problem. Keep in mind that Brute Force solution is not allowed.

**Discussion of a test case:**

**Input:**

String 1 = "AGCAGTGT";

String 2 = "ACGTATC";

**Output:**

Minimum penalty aligning two problem is: 9

Aligned as:

```
AGCAGTGT_
A_C_GTATC
```

**Input Format:**

As input in the first line, you will be given two strings S1 and S2.

**Output Format:**

For each input there will be three lines of output. First line will contain the minimum penalty to align two strings. In the second line, the aligned condition of S1 and in the third line the aligned condition of S2. '_' means there is a gap induced. The final length of S1 and S2 will be same.

## Problem ID: 9

### Project: Clique Problem

### Difficulty: Hard

The Clique problem is the computational problem of finding cliques (subsets of vertices, all adjacent to each other, also called complete subgraphs) in a graph. It has several different formulations depending on which cliques, and what information about the cliques, should be found.

Common formulations of the clique problem include finding a maximum clique (a clique with the largest possible number of vertices), finding a maximum weight clique in a weighted graph, listing all maximal cliques (cliques that cannot be enlarged), and solving the decision problem of testing whether a graph contains a clique larger than a given size.

In this project, you have to design a suitable algorithm to find the cliques of a given size in a graph. You should not use Brute Force as it may affect the complexity and runtime of the program.

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges. Each such line will contain two integers u and v which indicates that there is an edge between nodes u and v.

After all the edges' information, you will be given another integer K denoting the clique size.

**Output Format:**

You need to print all the cliques of size K found in the given graph. Print each clique's information in separate lines. For each clique, print the nodes' numbers according to their ascending order of ids.

## Problem ID: 10

**Project: Strongly Connected Graph (SCC) to Direct Acyclic Graph (DAG)**

**Difficulty: Medium**

In this problem, you will be given a directed unweighted graph which may contain many cycles. From this given graph, you need to find the strongly connected components (SCC). The **strongly connected components** (**SCC**) of a directed graph are its maximal strongly connected subgraphs. After discovering the SCCs, you also need to construct the directed acyclic graph (DAG) found from this given graph. It is important to note that, by connecting the edges lying between two SCCs, you get a DAG. An example is shown in the given problem.

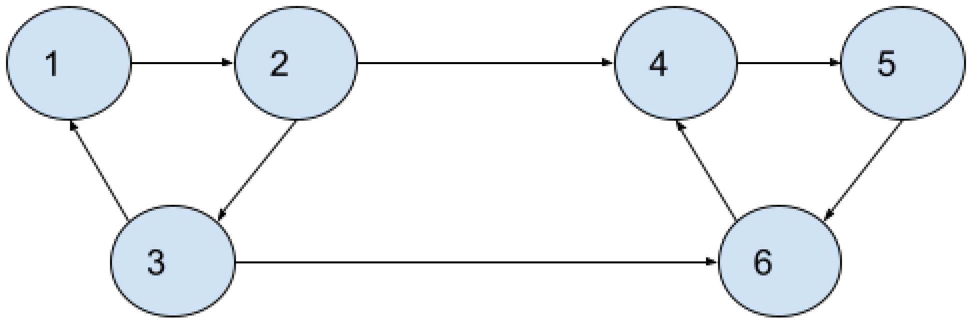
**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges. Each such line will contain two integers u and v which indicates that there is an edge from node u to v.

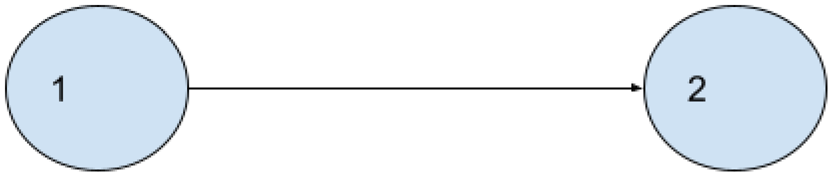
**Output Format:**

As output for each test case, first you need to print the number of SCC components found in the given graph. Let the number be n. In the following n lines you will provide the description of each SCC. Line i will contain the information about the ith SCC. Description of each SCC means which nodes comprise the SCC.

After printing the SCCs you will focus on DAG. In the first following line after print the SCCs you will print the number of edges of the DAG. Let the number be e. So, in the e following lines you will print the information about the edges of the DAG where each line will contain two integer values u and v denoting that from the uth SCC there is an edge to vth SCC.



Input graph



SCC->DAG

From the figure shown above, the output will be as follows,

```
2
1: 1 2 3
2: 4 5 6
```

```
1
1 -> 2
```

## Problem ID: 11

**Project:Multiple Shortest Path Printing**

**Difficulty: Medium**

In this problem, you will be given a directed weighted graph containing no negative cycles. You will be given a source node S from which you will calculate the distance to another destination node V which will also be given. In this problem the edge weight denotes the to be covered distance between two nodes by traversing through that edge. Here, you also need to find the path that will help you to reach the destination. As there can be multiple paths, to reach a destination, your solution needs to be able to find all the shortest paths or routes. For this problem, you can safely assume that, there will not be be more than 10 shortest paths between any two nodes.

This problem can be solved using modified dijkstra. In the algorithm of dijkstra we use D[v] to denote a node's distance from the source. Here you can use two dimensional array to solve this problem (D[v][p]) where the array will hold the shortest distance of v from the source in path number p. In the similar fashion using 2D array you can also save multiple paths' parent nodes information.

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges. Each such line will contain three integers u , v and w, which indicates that there is an edge from node u to v having weight w. After the graph's information, you will be given two nodes U and V denoting the source and destination vertex respectively.

**Output Format:**

In the first line, you need to print the minimum distance between U and V. In the following line, you will print the number of shortest paths or routes found between U and V that give the same minimum distance. Let the number be e. In the e following lines you will print the intermediate nodes to reach the V from S excluding V and S.

## Problem ID: 12

**Project: Maximum Sum Interval**

**Difficulty: Easy**

The maximum subarray problem is the task of finding the contiguous subarray within a one dimensional array of numbers which has the largest sum. The list usually contains both positive and negative numbers along with 0. Some properties of this problem are:

1. If the array contains all non-positive numbers, then the solution is the number in the array with the smallest magnitude.
2. If the array contains all non-negative numbers, then the problem is trivial and the maximum sum is the sum of all the elements in the list.
3. An empty sets not valid.
4. There can be multiple different sub-arrays that achieve the same maximum sum to the problem.

For example, in the array [-5, 6, 7, 1, 4, -8, 16], the maximum sum is 26. That is because adding 6 + 7 + 1 + 4 + -8 + 16 gives us 26.

In this project, you have to design a suitable algorithm to find the maximum sum interval for a given array. You can use **Divide and Conquer** or **Dynamic Programming** or **AD Hoc** approaches in this regard. You should not use Brute Force as it may affect the complexity and runtime of the program.

**Input:**

An array with n number of index.

**Output:**

The maximum sum interval for the given array.

## Problem ID: 13

**Project: Matrix Chain Multiplication**

**Difficulty: Easy**

Given a sequence of matrices, find the most efficient way to multiply these matrices together. However, the problem is not actually to perform the multiplications, rather you have to decide in which order the multiplications should be performed.

Since matrix multiplication is associative, there are so many options to multiply a chain of matrices. In other words, no matter how we parenthesize the product, **the result will be the same**. For example, if we had four matrices A, B, C, and D, we would have:

A(BCD) = (AB)(CD) = (ABC)D = .......

However, the order in which we parenthesize the product **affects the number of simple arithmetic operations** needed to compute the product, or the efficiency. For example, suppose A is a 10 × 30 matrix, B is a 30 × 5 matrix, and C is a 5 × 60 matrix. Then,

1. (AB)C = (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations
2. A(BC) = (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations.

Clearly the first parameterization requires less number of operations.

Use Dynamic Programming (DP) approach to find the best association such that the result is obtained with the minimum number of arithmetic operations. You also need to print the association order of how the matrix is multiplied, e.g, ((AB)C) , (A(BC)), etc. You will be given the matrix information as input, each matrix's row and column information.

## Problem ID: 14

**Project: Articulation Point and Bridges**

**Difficulty: Easy**

In this problem, you will be given an undirected unweighted graph. You need to design an algorithm that will calculate all the articulation points and bridges found in this graph. A node becomes an Articulation point when by removing this node and the edges connected to it, the number of connected components of the graph increases. Similarly an articulation bridge denotes an edge where by removing this edge the number of connected components of the graph increases.

**Input Format:**

In the first line you will be given an integer N denoting the number of nodes and another integer E denoting the number of edges. In the following E lines you will be given information about the edges. Each such line will contain two integers u and v which indicates that there is an undirected edge between nodes u and v.

**Output Format:**

In the first line you will print an integer number n denoting the number of articulation points found in the given graph. In the following line, you will print n nodes where each node will denote an articulation node's vertex.
After printing the articulation points, you will print another integer e which will denote the number of articulation bridges. In the following e lines, you will print e articulation bridges where each line will have two integer values u and v denoting an edge between nodes u and v.

## Problem ID: 15

**Project: Closest Pair of Points**

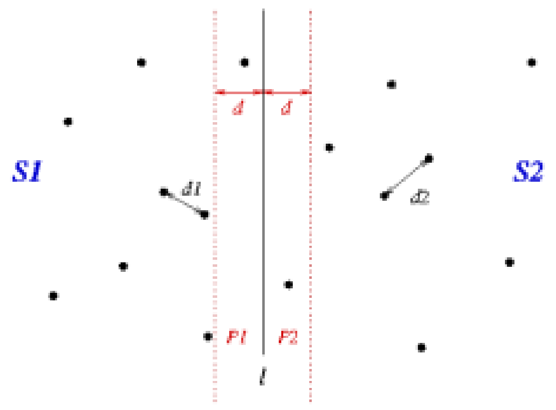
**Difficulty: Medium**

Here, we are going to win a war using sound theory and algorithm.

In a war, two countries are participating, country A and country B. A is planning to launch missiles on B to destroy each of their army camps. Now, A knows the location of each army camp of B as a 2D coordinate (x,y). When a missile attacks a place, it creates a huge sound. So, they are trying to optimize here by designing such a missile that will create very less sound.

In this regard, they are planning to calculate all pair distances for each army camp. T**heir goal is to find the minimum pair distance among them**. They are going to design a missile in such a way so that the sound it creates will be so milder that it would not reach the minimum distance even. For this, the other camps will not be informed early and they can take them by surprise and full power.

Now, to solve this problem, you are going to write a divide and conquer based algorithm that will calculate the minimum pair distance found among all the pair distances.



$$\|pq\| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

**Input**:

An array of N, 2D coordinates in the plane.

**Output**:

The smallest distance found between any pair of points in the given array.

## Problem ID: 16

**Project: Lexicographically Smallest Longest Common Subsequence**

**Difficulty: Medium**

In this problem, you will be given two strings containing any ASCII characters. You need to find the length of the Longest Common Subsequence(LCS) observed between these two given sequences. You also need to find the lexicographically smallest LCS observed here.

**Sample Input and Output**:

| | |
|---|---|
| ABCDEF<br>DEFABC | 3<br>ABC |
| DAEABAA<br>ADAEFBAA | 6<br>DAE BAA |