Downloaded from: justpaste.it/dik1v

1. BFS DFS----------

```
class BfsDfs:

    def __init__(self):
        self.vis = []

    def generate_graph(self):
        g = {}
        vertices = int(input("Enter the number of vertices: "))
        for i in range(vertices):
            edges = list(map(int, input(f"Enter the vertices connected to vertex {i}: ").split()))
            g[i] = edges

        self.vis = [False] * vertices
        return g

    def dfs(self, g, s):
        self.vis[s] = True
        print(s, end=" ")
        for c in g[s]:
            if not self.vis[c]:
                self.dfs(g, c)

    def bfs(self, g, s):
        q = [s]
        visit = [s]
        print("\nBFS: ")
        while q:
            curr = q.pop(0)
            print(curr, end=" ")
            for c in g[curr]:
                if c not in visit:
                    q.append(c)
                    visit.append(c)

obj = BfsDfs()
g = obj.generate_graph()
s = int(input("\nEnter Starting Vertex: "))
print("\nDFS:")
obj.dfs(g, s)
```

obj.bfs(g, s)

2. A STAR----------

```
global g
g=0
def print_board(elements):
    for i in range(9):
        if i%3 == 0:
            print()
        if elements[i]==-1:
            print("_", end = " ")
        else:
            print(elements[i], end = " ")
    print()

def solvable(start):
    inv=0

    for i in range(9):
        if start[i] <= 1:
            continue
        for j in range(i+1,9):
            if start[j]==-1:
                continue
            if start[i]>start[j]:
                inv+=1
    if inv%2==0:
        return True
    return False

def heuristic(start,goal):
    global g
    h = 0
    for i in range(9):
        for j in range(9):
            if start[i] == goal[j] and start[i] != -1:
                h += (abs(j-i))//3 + (abs(j-i))%3
    return h + g

def moveleft(start,position):
    start[position],start[position-1]= start[position-1],start[position]

def moveright(start,position):
    start[position],start[position+1]= start[position+1],start[position]

def moveup(start,position):
    start[position],start[position-3]= start[position-3],start[position]

def movedown(start,position):
    start[position],start[position+3]= start[position+3],start[position]


def movetile(start,goal):
    emptyat = start.index(-1)
```

```
    row = emptyat//3
    col = emptyat%3

    t1,t2,t3,t4 = start[:],start[:],start[:],start[:]
    f1,f2,f3,f4 = 100,100,100,100

    if col-1 >= 0:
        moveleft(t1, emptyat)
        f1 = heuristic(t1, goal)
    if col+1 < 3:
        moveright(t2, emptyat)
        f2 = heuristic(t2, goal)
    if row + 1 < 3:
        movedown(t3, emptyat)
        f3 = heuristic(t3, goal)
    if row-1 >= 0 :
        moveup(t4, emptyat)
        f4 = heuristic(t4, goal)


    min_heuristic = min(f1, f2, f3, f4)

    if f1==min_heuristic:
        moveleft(start, emptyat)
    elif f2==min_heuristic:
        moveright(start, emptyat)
    elif f3==min_heuristic:
        movedown(start, emptyat)
    elif f4 == min_heuristic:
        moveup(start, emptyat)


def solveEight(start,goal):
    global g
    g+=1
    movetile(start,goal)
    print_board(start)
    f = heuristic(start,goal)
    print(f"f(n): {f}")
    if f == g:
        print(f"\nSolved in {f} moves")
        return
    solveEight(start,goal)

start = list()
goal = list()
print("Enter the start state:(Enter -1 for empty):")
for i in range(9):
    start.append(int(input()))
print("Enter the goal state:(Enter -1 for empty):")
for i in range(9):
    goal.append(int(input()))


print('Start state')
print_board(start)
print("-------------------------------------------- ")
```

```
if solvable(start):
    solveEight(start,goal)
else:
    print("Not possible to solve")
```

```
3.1. SELECTION SORT-------------------------------------------------------------------------
```

```
import java.util.*;

public class SelectionSort {
   public static void selectionSort(int[] arr) {
      int n = arr.length;

      for (int i = 0; i < n - 1; i++) {
         int min_idx = i;
         for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
               min_idx = j;
            }
         }

         int temp = arr[min_idx];
         arr[min_idx] = arr[i];
         arr[i] = temp;
      }
   }

   public static void main(String args[]) {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter the size of the input: ");
      int n = in.nextInt();
      int arr[] = new int[n];
      System.out.println("Enter the elements of the array");
      for (int i = 0; i < n; i++) {
         System.out.print("Enter the " + (i + 1) + " element: ");
         arr[i] = in.nextInt();
      }
      System.out.println("Unsorted array:");
      for (int i = 0; i < n; i++) {
         System.out.print(arr[i] + " ");
      }
      selectionSort(arr);
      System.out.println("\nSorted array:");
```

```
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

3. 2. JOB SCHEDULING----------

```
import java.util.*;

class Job {
    int id, deadline, profit;

    public Job(int id, int deadline, int profit) {
        this.id = id;
        this.deadline = deadline;
        this.profit = profit;
    }
}

class JobScheduling {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the no of Job you want to enter: ");
        int n=in.nextInt();
        Job[] jobs =new Job[n];
        System.out.print("Enter the details of the Job: \n");
        for(int i=0;i<n;i++){
            System.out.println("Job "+(i+1)+" :");
            System.out.print("Enter the id of Job: ");
            int id=in.nextInt();
            System.out.print("Enter the deadline of Job: ");
            int deadline=in.nextInt();
            System.out.print("Enter the profit of Job: ");
            int profit=in.nextInt();
            jobs[i]=new Job(id, deadline, profit);
        }
        Arrays.sort(jobs, (a, b) -> b.profit - a.profit);

        int maxDeadline = Integer.MIN_VALUE;
        for (Job job : jobs) {
            maxDeadline = Math.max(maxDeadline, job.deadline);
        }

        int[] slots = new int[maxDeadline + 1];
```

```
        int totalProfit = 0;
        for (Job job : jobs) {

            for (int i = job.deadline; i > 0; i--) {
                if (slots[i] == 0) {
                    slots[i] = job.id;
                    totalProfit += job.profit;
                    break;
                }
            }
        }

        System.out.print("Scheduled Jobs: ");
        for (int i = 1; i < slots.length; i++) {
            if (slots[i] != 0) {
                System.out.print(slots[i] + " ");
            }
        }
        System.out.println("\nTotal Profit: " + totalProfit);
    }
}
```

3.3 Prims----------

```
def minKey(key, mstSet):
    #(8 5 42344 6)
    min = float('inf')
    minIndex = -1

    for i in range(len(key)):
        if not mstSet[i] and key[i] < min:
            min = key[i]
            minIndex = i

    return minIndex


def printMST(parent, graph, sum):
    print("Edge \tWeight")
    for i in range(1, len(parent)):
        print(parent[i], "-", i, "\t", graph[i][parent[i]])
    print("Minimum weight of MST:", sum)


def prim(graph, numVertices):
    parent = [0] * numVertices
```

```
    key = [float('inf')] * numVertices
    mstSet = [False] * numVertices

    key[0] = 0
    parent[0] = -1

    for count in range(numVertices - 1):
        u = minKey(key, mstSet)
        mstSet[u] = True

        for v in range(numVertices):
            if graph[u][v] != 0 and not mstSet[v] and graph[u][v] < key[v]:
                parent[v] = u
                key[v] = graph[u][v]

    sum = 0
    for i in range(numVertices):
        sum += key[i]

    printMST(parent, graph, sum)


n = int(input("Enter the size of the graph: "))
graph = [[0] * n for _ in range(n)]

for i in range(n):
    for j in range(n):
        graph[i][j] = int(input("Enter the weight {}->{} of the graph: ".format(i, j)))

prim(graph, n)
```

3.4 KRUSKALS----------

```
class Edge:
    def __init__(self, src, dest, weight):
        self.src = src
        self.dest = dest
        self.weight = weight

    def __lt__(self, other):
        return self.weight < other.weight

def find(parent, i):
    if parent[i] != i:
```

```
        parent[i] = find(parent, parent[i])
    return parent[i]

def kruskal(graph, numVertices):
    edges = []
    for i in range(numVertices):
        for j in range(i + 1, numVertices):
            if graph[i][j] != 0:
                edge = Edge(i, j, graph[i][j])
                edges.append(edge)

    edges.sort()

    parent = list(range(numVertices))
    mst = []
    total_weight = 0

    for edge in edges:
        src_parent = find(parent, edge.src)
        dest_parent = find(parent, edge.dest)
        if src_parent != dest_parent:
            mst.append(edge)
            parent[src_parent] = dest_parent
            total_weight += edge.weight

    print("Edges in the MST:")
    for edge in mst:
        print(edge.src, "-", edge.dest, ":", edge.weight)
    print("Minimum weight of MST:", total_weight)

n = int(input("Enter the size of the graph: "))
graph = []
for i in range(n):
    row = []
    for j in range(n):
        weight = int(input("Enter the weight " + str(i) + "->" + str(j) + " of the graph: "))
        row.append(weight)
    graph.append(row)

kruskal(graph, n)

"""
Enter the  size of the graph: 5
Enter the weight 0-> 0 of the graph: 0
Enter the weight 0-> 1 of the graph: 2
```

```
Enter the weight 0-> 2 of the graph: 0
Enter the weight 0-> 3 of the graph: 6
Enter the weight 0-> 4 of the graph: 0
Enter the weight 1-> 0 of the graph: 2
Enter the weight 1-> 1 of the graph: 0
Enter the weight 1-> 2 of the graph: 3
Enter the weight 1-> 3 of the graph: 8
Enter the weight 1-> 4 of the graph: 5
Enter the weight 2-> 0 of the graph: 0
Enter the weight 2-> 1 of the graph: 3
Enter the weight 2-> 2 of the graph: 0
Enter the weight 2-> 3 of the graph: 0
Enter the weight 2-> 4 of the graph: 7
Enter the weight 3-> 0 of the graph: 6
Enter the weight 3-> 1 of the graph: 8
Enter the weight 3-> 2 of the graph: 0
Enter the weight 3-> 3 of the graph: 0
Enter the weight 3-> 4 of the graph: 9
Enter the weight 4-> 0 of the graph: 0
Enter the weight 4-> 1 of the graph: 5
Enter the weight 4-> 2 of the graph: 7
Enter the weight 4-> 3 of the graph: 9
Enter the weight 4-> 4 of the graph: 0
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
Minimum weight of MST: 16
"""
```

3.5 DJKSTRAS----------

```
#Djikstra's Algorithm

def djikstra(graph,source):
    distances = {node: float('inf') for node in graph}
    distances[source] = 0

    unvisited_nodes = list(graph.keys())

    while unvisited_nodes:

        current_node = min(unvisited_nodes,key=lambda node: distances[node])
```

```
        unvisited_nodes.remove(current_node)

        for neighbour, weight in graph[current_node].items():
            distance = distances[current_node] + weight

            if distance < distances[neighbour]:
                distances[neighbour] = distance

    return distances

def main():
    graph = {}
    num_edges = int(input("Enter the size of the graph: "))
    for _ in range(num_edges):
        start, end, weight = input("Enter the numbers(Start END Weight): ")
        weight = int(weight)
        if start not in graph:
            graph[start] = {}
        if end not in graph:
            graph[end] = {}
        graph[start][end] = weight
        graph[end][start] = weight

    source_node = input("Enter the source node: ")
    end_node = input("Enter the end node: ")

    distances = djikstra(graph,source_node)
    shortest_distance = distances[end_node]

    print(f"The shortest distance from {source_node} to {end_node} is: {shortest_distance}")

if _name_ == '_main_':
    main()
```

4. GRAPH COLORING ----------

```
import java.util.*;

public class GraphColoring {
    private final int vertices;
    private final int[][] graph;
    private int[] colors;
    private int minColors;

    public GraphColoring(int vertices, int[][] graph) {
        this.vertices = vertices;
```

```
        this.graph = graph;
        this.colors = new int[vertices];
        this.minColors = Integer.MAX_VALUE;
    }

    public void solve() {
        Arrays.fill(colors, -1); // array with -1, indicating that no color has been assigned to any vertex yet.
        tryColoring(0, 1); // first vertex (index 0) with color 1.
    }

    private void tryColoring(int vertex, int numColors) { // Recursive Function
        if (numColors >= minColors)
            return; // Branch and bound condition

        if (vertex == vertices) {
            minColors = numColors;
            printSolution(numColors);
            return;
        } // If all vertices are colored, it updates the minimum colors needed and prints
          // the solution.

        for (int color = 1; color <= numColors; color++) { // It iterates through all possible colors for the current
                                                           // vertex.
            if (isSafe(vertex, color)) {
                colors[vertex] = color;
                tryColoring(vertex + 1, numColors);
                colors[vertex] = -1; // Backtrack
            }
        } // After coloring, it backtracks by resetting the color of the current vertex.

        // Try to use a new color
        colors[vertex] = numColors + 1;
        tryColoring(vertex + 1, numColors + 1);
        colors[vertex] = -1;
    }

    private boolean isSafe(int vertex, int color) { // The isSafe() method checks if it's safe to color the vertex
with
                                                    // the given color.
        for (int i = 0; i < vertices; i++) {
            if (graph[vertex][i] == 1 && colors[i] == color) {
                return false;
            }
        }
        return true;
```

```
    }

    private void printSolution(int numColors) {
        System.out.println("Solution found with " + numColors + " colors:");
        for (int i = 0; i < vertices; i++) {
            System.out.println("Vertex " + i + " ---> Color " + colors[i]);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of vertices:");
        int vertices = scanner.nextInt();
        int[][] graph = new int[vertices][vertices];

        System.out.println("Enter the adjacency matrix (0 for no edge, 1 for edge):");
        for (int i = 0; i < vertices; i++) {
            for (int j = 0; j < vertices; j++) {
                System.out.print("Is there an edge between vertex " + i + " and vertex " + j + "? (0/1): ");
                graph[i][j] = scanner.nextInt();
            }
        }

        GraphColoring coloring = new GraphColoring(vertices, graph);
        coloring.solve();
    }
}
```

5. CHATBOT----------

```
import java.util.Scanner;

public class ChatBot {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Welcome to Clothing Store Chatbot!");
        System.out.println("How can I assist you today?");
        System.out.println("You can ask me about products, availability, sizes, and more.");
        System.out.println("Type 'exit' to end the conversation.");

        // Chat loop
```

```
        while (true) {
            System.out.print("You: ");
            String userMessage = scanner.nextLine().toLowerCase();

            // Check if user wants to end the conversation
            if (userMessage.equals("exit")) {
                System.out.println("Clothing Store Chatbot: Thank you for visiting. Have a great day!");
                break;
            }

            // Respond to user input
            String chatbotResponse = generateResponse(userMessage);
            System.out.println("Clothing Store Chatbot: " + chatbotResponse);
        }

        scanner.close();
    }

    // Generate a response based on user input
    public static String generateResponse(String userMessage) {
        // Predefined responses based on user input
        String response = "";
        switch (userMessage) {
            case "hello":
                response = "Hello! Welcome to our online store. How can I assist you today?";
                break;
            case "what products do you offer?":
                response = "We offer a wide range of products including clothing, accessories, shoes, and more!";
                break;
            case "do you have any sales or promotions?":
                response = "Yes, we often have sales and promotions! You can check our website or sign up for
our newsletter to stay updated.";
                break;
            case "can I track my order?":
                response = "Of course! Please provide your order number and I'll look up the status for you.";
                break;
            case "how do I return an item?":
                response = "We have a hassle-free return policy. Simply contact our customer service team and
they'll assist you with the return process.";
                break;
            case "what payment methods do you accept?":
                response = "We accept various payment methods including credit/debit cards, PayPal, and bank
transfers.";
                break;
            case "do you offer international shipping?":
```

```
            response = "Yes, we offer international shipping to many countries. You can check the shipping options during checkout.";
            break;
        case "how long does shipping take?":
            response = "Shipping times depend on your location and the shipping method chosen. You can find estimated delivery times during checkout.";
            break;
        case "is my personal information secure?":
            response = "Absolutely! We take customer privacy and security very seriously. Your personal information is encrypted and protected.";
            break;
        case "what's your refund policy?":
            response = "We offer refunds on eligible items within a specified period. Please refer to our refund policy for more details.";
            break;
        case "do you have a customer loyalty program?":
            response = "Yes, we have a loyalty program where you can earn points for every purchase and redeem them for discounts or rewards.";
            break;
        case "can I speak to a human representative?":
            response = "Certainly! If you need further assistance, please contact our customer service team and they'll be happy to help.";
            break;
        case "bye":
            response = "Thank you for visiting our store! Have a great day!";
            break;
        default:
            response = "I'm sorry, I didn't understand that. Can you please rephrase?";
      }
      return response;
   }
}
```