https://github.com/Gandhiraj754/lab

UNDERSTANDING INPUT AND OUTPUT IN DATA STRUCTURES AND ALGORITHMS:

PROF: DR.ANIMESH CHATURVEDI

TULASINARAYANARAO	22bds004
RAGHU VAMSI	22bds014
CHINTA GANDHI RAJ	22bds017
RAJ KUMAR	22bds055

PRIMS ALGORITHM

PRIM'S ALGORITHM IS A GREEDY ALGORITHM USED TO FIND THE MINIMUM SPANNING TREE (MST) OF A CONNECTED, UNDIRECTED GRAPH WITH WEIGHTED EDGES.

FIND THE PRIM'S ALGORITHM FOR MINIMUM SPANNING TREE.

INPUT:

Enter the number of vertices in the graph: 4
Enter the adjacency matrix for the graph:
0 2 3 0
2 0 0 0
3 0 0 4
0 0 4 0

OUTPUT:

Edge	Weight
0 1	2
0 2	3
2 3	4

KRUSKAL'S ALGORITHM

KRUSKAL'S ALGORITHM IS A GREEDY ALGORITHM FOR FINDING THE MINIMUM SPANNING TREE IN A CONNECTED, WEIGHTED GRAPH BY ADDING MINIMUM-WEIGHT EDGES WITHOUT FORMING CYCLES.

FIND THE MINIMUM SPANNING TREE USING KRUSKAL'S ALGORITHM.

INPUT:
Enter the number of cities (vertices) in the map: 4
ENTER THE NUMBER OF ROADS (EDGES) IN THE MAP: 4
ENTER THE ROADS AND THEIR COSTS (SOURCE DESTINATION COST):
1 2 2
1 4 4
4 3 3
2 3 6

OUTPUT:
Minimum cost to connect all cities in the map: 9

BREADTH-FIRST SEARCH (BFS)

BFS IS A TREE TRAVERSAL ALGORITHM THAT EXPLORES NODES LEVEL BY LEVEL, USING A QUEUE FOR NODE EXPANSION. IT VISITS ALL NODES AT THE SAME DEPTH BEFORE MOVING DEEPER.

FIND BINARY SEARCH TREE (BST) WITH BASIC FUNCTIONS FOR INSERTION AND BREADTH-FIRST TRAVERSAL (BFS).

INPUT :

Enter the elements of the tree (enter 0 to exit):
5 3 7 2 4 6 8 0

OUTPUT :

Elements in the tree using breadth-first traversal (BFS): 5 -> 3 -> 7 -> 2 -> 4 -> 6 -> 8 ->

DEPTH-FIRST SEARCH (DFS)

DFS AN ALGORITHM FOR TRAVERSING OR SEARCHING TREE OR GRAPH DATA STRUCTURES, EXPLORING AS FAR AS POSSIBLE ALONG EACH BRANCH BEFORE BACKTRACKING.

FIND THE DIRECTED GRAPH CYCLE DETECTION USING DEPTH-FIRST SEARCH (DFS).

INPUT:

Enter the number of vertices in the graph: 4
Enter the number of edges in the graph: 5
Enter the edges (source destination) of the graph:
0 1
1 2
2 3
3 1
1 0

OUTPUT:

The directed graph contains a cycle

STACKS

A STACK IS A LINEAR DATA STRUCTURE THAT FOLLOWS THE LAST IN FIRST OUT (LIFO) PRINCIPLE. IT ALLOWS ADDING AND REMOVING ELEMENTS FROM ONE END, KNOWN AS THE TOP, MAKING IT SUITABLE FOR VARIOUS APPLICATIONS LIKE RECURSION, EXPRESSION EVALUATION, AND BACKTRACKING.

FIND STOCK SPAN PROBLEM USING STACK.

INPUT:

Enter the number of days: 3
Enter the prices for each day:
100
120
140

OUTPUT:

Stock Span:
Day 1: 1
Day 2: 2
Day 3: 3

QUEUES

THE CODE IMPLEMENTS A CIRCULAR QUEUE IN C, ALLOWING INSERTION AND REMOVAL OF ELEMENTS FROM THE FRONT AND REAR. IT HANDLES QUEUE FULLNESS AND EMPTINESS, AND DISPLAYS THE FRONT ELEMENT. CIRCULAR QUEUES EFFICIENTLY MANAGE DATA BY OVERWRITING OLD ELEMENTS WHEN FULL.

WRITE A C CODE IMPLEMENTING OF A CIRCULAR QUEUE USING AN ARRAY.

INPUT:

```
Enter elements to enqueue (enter -1 to stop):
10
20
30
-1
```

OUTPUT:

```
Element removed: 10
Enter elements to enqueue (enter -1 to stop):
40
50
--1
Element at front: 20
 ---------------------
 Index : 5 4 3 2 1 0
 ---------------------
 Queue: 20 40 50
```

LINKED LISTS

A LINKED LIST IS A LINEAR DATA STRUCTURE WHERE EACH ELEMENT (NODE) CONTAINS DATA AND A REFERENCE (POINTER) TO THE NEXT NODE. IT PROVIDES DYNAMIC MEMORY ALLOCATION FOR FLEXIBLE SIZE AND EASY INSERTION/DELETION.

FIND THE LINKED LIST SUM CALCULATION.

INPUT:

```
Enter numbers (enter -1 to stop):
10
20
30
-1
```

OUTPUT:

```
Sum of the numbers: 60
```

BINARY HEAP

A BINARY HEAP IS A BINARY TREE WITH A SPECIAL PROPERTY: THE VALUE OF EACH NODE IS EITHER GREATER THAN OR EQUAL TO (MAX HEAP) OR LESS THAN OR EQUAL TO (MIN HEAP) ITS CHILD NODES.

FIND Binary Heap Implementation in C with Insertion and Removal.

INPUT:

Enter the number of elements to insert into the binary heap: 3
Enter the elements: 10 30 20

OUTPUT:

Binary Heap after insertion: 10 30 20
Removing minimum element: 10
Binary Heap after removal: 20 30
Removing minimum element: 20
Binary Heap after removal: 30
Removing minimum element: 30
Binary Heap after removal

DIJKSTRA'S ALGORITHM

IT OPERATES BY ITERATIVELY SELECTING THE VERTEX WITH THE MINIMUM DISTANCE FROM THE SOURCE AND UPDATING THE DISTANCES OF ITS NEIGHBOURS, UNTIL ALL VERTICES ARE VISITED. THE ALGORITHM IS GREEDY AND CAN HANDLE NON-NEGATIVE EDGE WEIGHTS.

FIND DIJKSTRA'S SHORTEST PATH ALGORITHM.

INPUT:

```
Enter the number of nodes in the graph: 5
Enter the number of edges in the graph: 7
Enter the adjacency matrix of the graph:
0 10 0 5 0
0  0  1 2 0
0  0  0 0 4
0  3  9 0 2
7  0  6 0 0
Enter the source vertex: 0
```

OUTPUT:

Vertex	Distance	Path
0	0	0
1	8	0 3 1
2	9	0 3 1 2
3	5	0 3
4	7	0 3 4

BELLMAN-FORD ALGORITHM

BELLMAN-FORD IS A SINGLE-SOURCE SHORTEST-PATH ALGORITHM THAT HANDLES NEGATIVE EDGE WEIGHTS. IT ITERATES THROUGH ALL EDGES, RELAXING THEM BY UPDATING DISTANCES UNTIL THE OPTIMAL SHORTEST PATHS ARE ACHIEVED OR NEGATIVE CYCLES DETECTED.

FIND THE BELLMAN-FORD ALGORITHM FOR FINDING SHORTEST DISTANCES IN A WEIGHTED GRAPH.

INPUT:

Enter the number of vertices in the graph: 5
Enter the number of edges in the graph: 6
Enter the edges (source destination) of the graph:
0 1 3
0 2 6
1 3 2
2 3 1
3 4 4
4 2 7
Enter the source vertex: 0

OUTPUT:

Shortest distances from vertex 0 to all other vertices:
Vertex 0: Distance = 0, Parent = -1
Vertex 1: Distance = 3, Parent = 0
Vertex 2: Distance = 6, Parent = 0
Vertex 3: Distance = 5, Parent = 1
Vertex 4: Distance = 9, Parent = 3

THANK YOU