

DATA STRUCTURES

1) Write a c program to implement single source shortest algorithm (Dijkstra's)?

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
    int G[MAX][MAX],i,j,n,u;
    printf("Enter no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the adjacency matrix:\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);
    printf("\nEnter the starting node:");
    scanf("%d",&u);
    dijkstra(G,n,u);
    return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{
    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int visited[MAX],count,mindistance,nextnode,i,j;

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(G[i][j]==0)
                cost[i][j]=INFINITY;
            else
                cost[i][j]=G[i][j];

    for(i=0;i<n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
    distance[startnode]=0;
    visited[startnode]=1;
    count=1;
    while(count<n-1)
    {
```

```

mindistance=INFINITY;

for(i=0;i<n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
nextnode=i;
}

visited[nextnode]=1;
for(i=0;i<n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}

for(i=0;i<n;i++)
if(i!=startnode)
{
printf("\nDistance of node%d=%d",i,distance[i]);
printf("\nPath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}while(j!=startnode);
}
}

```

```
C:\Users\babue\OneDrive\Desktop\data structures\dijkstra algorithm.cpp - Embarcadero Dev C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
Project: (globals)
dijkstra algorithm.cpp
//dijkstra.cpp
//Dijkstra's Algorithm
//Author: Babue
//Date: 20/01/2020
//Version: 1.0
//Description: This program implements Dijkstra's Algorithm to find the shortest path from a source node to all other nodes in a weighted undirected graph.

#include <iostream>
#include <vector>
#include <queue>
#include <limits>
using namespace std;

const int MAX = 100;
const int INF = 1000000000;

//Adjacency list representation
vector<vector<int>>> adj;
//Distance array
vector<int> dist;
//Visited array
vector<bool> visited;

//Function to add an edge to the adjacency list
void addEdge(int u, int v, int w) {
    adj[u].push_back(v);
    adj[v].push_back(u);
}

//Function to initialize the graph
void initGraph(int n) {
    adj.resize(n);
    dist.resize(n, INF);
    visited.resize(n, false);
}

//Function to find the shortest path from source node to all other nodes
void dijkstra(int s) {
    dist[s] = 0;
    priority_queue<int> pq;
    pq.push(s);
    while (!pq.empty()) {
        int u = pq.top();
        pq.pop();
        if (visited[u]) continue;
        visited[u] = true;
        for (int v : adj[u]) {
            if (dist[v] > dist[u] + adj[u][v]) {
                dist[v] = dist[u] + adj[u][v];
                pq.push(v);
            }
        }
    }
}

//Function to print the shortest path from source node to all other nodes
void printPath(int s) {
    for (int i = 0; i < dist.size(); i++) {
        cout << "Distance of node" << i << " = " << dist[i] << endl;
    }
}

//Function to print the shortest path from source node to a specific node
void printPath(int s, int t) {
    vector<int> path;
    int u = t;
    while (u != s) {
        path.push_back(u);
        for (int v : adj[u]) {
            if (dist[v] < dist[u]) {
                u = v;
                break;
            }
        }
    }
    path.push_back(s);
    reverse(path.begin(), path.end());
    cout << "Path = ";
    for (int i = 0; i < path.size(); i++) {
        cout << path[i] << " ";
        if (i % 10 == 9) cout << "\n";
    }
    cout << endl;
}

//Main function
int main() {
    int n, s, t;
    cout << "Enter no. of vertices: ";
    cin >> n;
    cout << "Enter the adjacency matrix:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int w;
            cout << "Weight: ";
            cin >> w;
            addEdge(i, j, w);
        }
    }
    initGraph(n);
    cout << "Enter the starting node: ";
    cin >> s;
    dijkstra(s);
    printPath(s);
    cout << "Enter the ending node: ";
    cin >> t;
    printPath(s, t);
    return 0;
}

//Output:
//Enter no. of vertices: 5
//Enter the adjacency matrix:
//0 10 0 30 100
//10 0 50 0 0
//0 50 0 20 10
//30 0 20 0 60
//100 0 10 60 0
//Enter the starting node: 0
//Distance of node=0
//Path=1<-0
//Distance of node2=50
//Path=2<-3<-0
//Distance of node3=30
//Path=3<-0
//Distance of node4=60
//Path=4<-2<-3<-0
//Process exited after 89.65 seconds with return value 0
//Press any key to continue . . .
```

2) Write a c program to implement minimum spanning tree from prim's algorithm?

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define V 5

int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}

void printMST(int parent[], int graph[V][V]) {
    printf("Edge  Weight\n");
    for (int i = 1; i < V; i++) {
        printf("%d - %d  %d \n", parent[i], i, graph[i][parent[i]]);
    }
}

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V];
    int mstSet[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    }
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
    printMST(parent, graph);
}

int main() {
    int graph[V][V];
    printf("Enter the adjacency matrix for the graph:\n");
    for (int i = 0; i < V; i++) {
```

```

        for (int j = 0; j < V; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    primMST(graph);
    return 0;
}

```

The screenshot shows the OnlineGDB interface with the following content:

Code Editor:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <limits.h>
4 #define V 5
5 int minKey(int key[], int mstSet[]) {
6     int min = INT_MAX, min_index;
7     for (int v = 0; v < V; v++) {
8         if (!mstSet[v] && key[v] < min) {
9             min = key[v];
10            min_index = v;
11        }
12    }
13    return min_index;
14 }
15 void printMST(int parent[], int graph[V][V]) {
16     printf("Edge \t Weight\n");
17     for (int i = 1; i < V; i++) {
18         printf("%d - %d \t %d \n", parent[i], i, graph[i][parent[i]]);
19     }
20 }

```

Input:

```

Enter the adjacency matrix for the graph:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

```

Output:

```

Edge \t Weight
0 - 1 \t 10
3 - 2 \t 20
0 - 3 \t 30
2 - 4 \t 10

```

...Program finished with exit code 0
Press ENTER to exit console.

3) Write a c program to implement Kruskal's algorithm?

Code:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int comparator(const void* p1, const void* p2)
{
    const int(*x)[3] = p1;
    const int(*y)[3] = p2;
    return (*x)[2] - (*y)[2];
}
```

```
void makeSet(int parent[], int rank[], int n)
{
    for (int i = 0; i < n; i++) {
        parent[i] = i;
        rank[i] = 0;
    }
}
```

```
int findParent(int parent[], int component)
{
    if (parent[component] == component)
        return component;
    return parent[component]
    = findParent(parent, parent[component]);
}
```

```
void unionSet(int u, int v, int parent[], int rank[], int n)
{
```

```
    u = findParent(parent, u);
    v = findParent(parent, v);
    if (rank[u] < rank[v]) {
        parent[u] = v;
    }
```

```
    else if (rank[u] > rank[v]) {
        parent[v] = u;
    }
```

```
    else {
        parent[v] = u;
```

```
    rank[u]++;
    }
}
```

```
void kruskalAlgo(int n, int edge[n][3])
{
```

```

qsort(edge, n, sizeof(edge[0]), comparator);
int parent[n];
int rank[n];

makeSet(parent, rank, n);

int minCost = 0;
printf(
"Following are the edges in the constructed MST\n");
for (int i = 0; i < n; i++) {
int v1 = findParent(parent, edge[i][0]);
int v2 = findParent(parent, edge[i][1]);
int wt = edge[i][2];

if (v1 != v2) {
unionSet(v1, v2, parent, rank, n);
minCost += wt;
printf("%d -- %d == %d\n", edge[i][0],
edge[i][1], wt);
}
}
printf("Minimum Cost Spanning Tree: %d\n", minCost);
}

int main()
{
int edge[5][3] = { { 0, 1, 10 },
{ 0, 2, 6 },
{ 0, 3, 5 },
{ 1, 3, 15 },
{ 2, 3, 4 } };
kruskalAlgo(5, edge);
return 0;
}

```

prims program in c - Yahoo... | Prim's Algorithm for Min... | Prim's Algorithm... | Prim's MST Algorithm in C... | gdb compiler - Yahoo India... | GDB online Debugger | Co... | +

onlinegdb.com

Gmail YouTube Maps News Translate Clipdrop - Uncrop

OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.

IDE
My Projects
Classroom **new**
Learn Programming
Programming Questions
Jobs **new**
Sign Up
Login

Learn Python with
KodeKloud

About • FAQ • Blog • Terms of Use • Contact Us • GDB
Tutorial • Credits • Privacy
© 2016 - 2023 GDB Online

main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int comparator(const void* p1, const void* p2)
5 {
6     const int(*x)[3] = p1;
7     const int(*y)[3] = p2;
8     return (*x)[2] - (*y)[2];
9 }
10
11 void makeSet(int parent[], int rank[], int n)
12 {
13     for (int i = 0; i < n; i++) {
14         parent[i] = i;
15         rank[i] = 0;
16     }
17 }
18
19 int findParent(int parent[], int component)
20 {
21     if (parent[component] == component)
22         return component;
23     return parent[component];
24 }
25
26 void unionSet(int u, int v, int parent[], int rank[], int n)
27 {
28     int pu = findParent(parent, u);
29     int pv = findParent(parent, v);
30     if (rank[pu] < rank[pv])
31         parent[pu] = pv;
32     else if (rank[pu] > rank[pv])
33         parent[pv] = pu;
34     else
35         parent[pv] = pu;
36 }
```

Following are the edges in the constructed MST

```
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

...Program finished with exit code 0
Press ENTER to exit console.

4) Write a c program to implement the graph traversals?

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int vertex;
    struct node *next;
};
struct node *createNode(int);
struct Graph
{
    int numVertices;
    struct node **adjLists;
    int *visited;
};
struct Graph *createGraph(int vertices)
{
    struct Graph *graph = malloc(sizeof(struct Graph));
    graph->numVertices = vertices;
    graph->adjLists = malloc(vertices * sizeof(struct node *));
    graph->visited = malloc(vertices * sizeof(int));
    int i;
    for (i = 0; i < vertices; i++)
    {
        graph->adjLists[i] = NULL;
        graph->visited[i] = 0;
    }
    return graph;
}
void addEdge(struct Graph *graph, int src, int dest)
{
    struct node *newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;
    newNode = createNode(src);
    newNode->next = graph->adjLists[dest];
    graph->adjLists[dest] = newNode;
}
struct node *createNode(int v)
{
    struct node *newNode = malloc(sizeof(struct node));
    newNode->vertex = v;
    newNode->next = NULL;
    return newNode;
}
void printGraph(struct Graph *graph)
{

```

```

int v;
for (v = 0; v < graph->numVertices; v++)
{
    struct node *temp = graph->adjLists[v];
    printf("\n Adjacency list of vertex %d\n ", v);
    while (temp)
    {
        printf("%d -> ", temp->vertex);
        temp = temp->next;
    }
    printf("\n");
}

void bfs(struct Graph *graph, int startVertex)
{
    struct node *queue = NULL;
    graph->visited[startVertex] = 1;
    enqueue(&queue, startVertex);
    while (!isEmpty(queue))
    {
        printQueue(queue);
        int currentVertex = dequeue(&queue);
        printf("Visited %d ", currentVertex);
        struct node *temp = graph->adjLists[currentVertex];
        while (temp)
        {
            int adjVertex = temp->vertex;
            if (graph->visited[adjVertex] == 0)
            {
                graph->visited[adjVertex] = 1;
                enqueue(&queue, adjVertex);
            }
            temp = temp->next;
        }
    }

    int isEmpty(struct node *queue)
    {
        return queue == NULL;
    }

    void enqueue(struct node **queue, int value)
    {
        struct node *newNode = createNode(value);
        if (isEmpty(*queue))
        {
            *queue = newNode;
        }
        else

```

```

{
struct node *temp = *queue;
while (temp->next)
{
temp = temp->next;
}
temp->next = newNode;
}
}
int dequeue(struct node **queue)
{
int nodeData = (*queue)->vertex;
struct node *temp = *queue;
*queue = (*queue)->next;
free(temp);
return nodeData;
}
void printQueue(struct node *queue)
{
while (queue)
{
printf("%d ", queue->vertex);
queue = queue->next;
}
printf("\n");
}
int main(void)
{
struct Graph *graph = createGraph(6);
printf("\nWhat do you want to do?\n");
printf("1. Add edge\n");
printf("2. Print graph\n");
printf("3. BFS\n");
printf("4. Exit\n");
int choice;
scanf("%d", &choice);
while (choice != 4)
{
if (choice == 1)
{
int src, dest;
printf("Enter source and destination: ");
scanf("%d %d", &src, &dest);
addEdge(graph, src, dest);
}
else if (choice == 2)
{
printGraph(graph);

```

```

}
else if (choice == 3)
{
int startVertex;
printf("Enter starting vertex: ");
scanf("%d", &startVertex);
bfs(graph, startVertex);
}
else
{
printf("Invalid choice\n");
}
printf("What do you want to do?\n");
printf("1. Add edge\n");
printf("2. Print graph\n");
printf("3. BFS\n");
printf("4. Exit\n");
scanf("%d", &choice);
}
return 0;
}

```

The screenshot shows the OnlineGDB interface with the following content:

OnlineGDB beta
 online compiler and debugger for c/c++
 code . compile . run . debug . share .

IDE
 My Projects
 Classroom **new**
 Learn Programming
 Programming Questions
 Jobs **new**
 Sign Up
 Login

Learn Python with
 KodeKloud

About • FAQ • Blog • Terms of Use • Contact Us • GDB
 Tutorial • Credits • Privacy
 © 2016 - 2023 GDB Online

Input

```

What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
1
Enter source and destination: 0 1
What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
1
Enter source and destination: 0 2
What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
1
Enter source and destination: 1 2
What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
1
Enter source and destination: 2 3
What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
2
Adjacency list of vertex 0
2 -> 1 ->

```

prims program in c - Yahoo

Prim's Algorithm for Minimi

Prim's Algorithm

Prim's MST Algorithm in C

gdb compiler - Yahoo India

GDB online Debugger | Co

onlinegdb.com

GmailYouTubeMapsNewsTranslateClipdrop - Uncrop

OnlineGDB beta

online compiler and debugger for c/c++

code.compile.run.debug.share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Jobs new

Sign Up

Login

Learn Python with KodeKloud

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy © 2016 - 2023 GDB Online

Input

2. Print graph
3. BFS
4. Exit
2

Adjacency list of vertex 0
2 -> 1 ->
Adjacency list of vertex 1
2 -> 0 ->
Adjacency list of vertex 2
3 -> 1 -> 0 ->
Adjacency list of vertex 3
2 ->
Adjacency list of vertex 4
< Adjacency list of vertex 5

What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit
3
Enter starting vertex: 0
0
Visited 0 2 1
Visited 2 1 3
Visited 1 3
Visited 3 What do you want to do?
1. Add edge
2. Print graph
3. BFS
4. Exit

