Veri

Yapıları Proje 4B Raporu

Saifullah NASRULLAH 05120000826

2)B-tree ekleme metodunu tasarlanması,kurşunkalemle yazılması ve kodlayıp çalıştırılması isteniyor.

**Kaynak Kod.**

#include<stdio.h>

#include<conio.h>

#include<iostream>

using namespace std;

struct BTreeNode

{

int \*data;

BTreeNode \*\*child\_ptr;

bool leaf;

int n;

}\*root = NULL, \*np = NULL, \*x = NULL;

BTreeNode \* init()

{

int i;

np = new BTreeNode;

np->data = new int[5];

np->child\_ptr = new BTreeNode \*[6];

np->leaf = true;

np->n = 0;

for (i = 0; i < 6; i++)

{

np->child\_ptr[i] = NULL;

}

return np;

}

void traverse(BTreeNode \*p)

{

cout<<endl;

int i;

for (i = 0; i < p->n; i++)

{

if (p->leaf == false)

{

traverse(p->child\_ptr[i]);

}

cout << " " << p->data[i];

}

if (p->leaf == false)

{

traverse(p->child\_ptr[i]);

}

cout<<endl;

}

void sort(int \*p, int n)

{

int i, j, temp;

for (i = 0; i < n; i++)

{

for (j = i; j <= n; j++)

{

if (p[i] > p[j])

{

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

int split\_child(BTreeNode \*x, int i)

{

int j, mid;

BTreeNode \*np1, \*np3, \*y;

np3 = init();

np3->leaf = true;

if (i == -1)

{

mid = x->data[2];

x->data[2] = 0;

x->n--;

np1 = init();

np1->leaf = false;

x->leaf = true;

for (j = 3; j < 5; j++)

{

np3->data[j - 3] = x->data[j];

np3->child\_ptr[j - 3] = x->child\_ptr[j];

np3->n++;

x->data[j] = 0;

x->n--;

}

for (j = 0; j < 6; j++)

{

x->child\_ptr[j] = NULL;

}

np1->data[0] = mid;

np1->child\_ptr[np1->n] = x;

np1->child\_ptr[np1->n + 1] = np3;

np1->n++;

root = np1;

}

else

{

y = x->child\_ptr[i];

mid = y->data[2];

y->data[2] = 0;

y->n--;

for (j = 3; j < 5; j++)

{

np3->data[j - 3] = y->data[j];

np3->n++;

y->data[j] = 0;

y->n--;

}

x->child\_ptr[i + 1] = y;

x->child\_ptr[i + 1] = np3;

}

return mid;

}

void insert(int a)

{

int i, temp;

x = root;

if (x == NULL)

{

root = init();

x = root;

}

else

{

if (x->leaf == true && x->n == 5)

{

temp = split\_child(x, -1);

x = root;

for (i = 0; i < (x->n); i++)

{

if ((a > x->data[i]) && (a < x->data[i + 1]))

{

i++;

break;

}

else if (a < x->data[0])

{

break;

}

else

{

continue;

}

}

x = x->child\_ptr[i];

}

else

{

while (x->leaf == false)

{

for (i = 0; i < (x->n); i++)

{

if ((a > x->data[i]) && (a < x->data[i + 1]))

{

i++;

break;

}

else if (a < x->data[0])

{

break;

}

else

{

continue;

}

}

if ((x->child\_ptr[i])->n == 5)

{

temp = split\_child(x, i);

x->data[x->n] = temp;

x->n++;

continue;

}

else

{

x = x->child\_ptr[i];

}

}

}

}

x->data[x->n] = a;

sort(x->data, x->n);

x->n++;

}

int main()

{

int i, n, t;

cout<<"enter the number of elements to be inserted\n";

cin>>n;

for(i = 0; i < n+1; i++)

{

cout<<"Enter the element\n";

cin>>t;

insert(t);

}

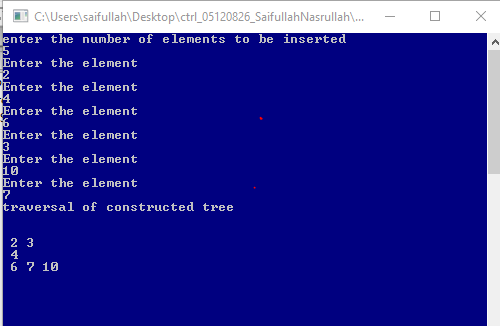
cout<<"traversal of constructed tree\n";

traverse(root);

getch();

}

**Ekran Çıktısı**



**3)**Dijkstra’s ,Prim’s ve DFT algoritmalarının kodlanıp çalıştırılması istenmektedir.

**Kaynak kod**

Dijkstra…

package proje4b;

public class Proje4B {

public static int INFINITY=1000;

public static void main(String[] args) {

int N = 5;

int[][] cost= {

{ 1000, 5, 3, 1000, 2},

{ 1000, 1000, 2, 6, 1000},

{ 1000, 1, 1000, 2, 1000},

{ 1000, 1000, 1000, 1000, 1000},

{ 1000, 6, 10, 4, 8} };

int distances[] = new int[N];

Distance(N, cost, distances);

for (int i=0; i<distances.length; ++i)

System.out.println(distances[i]);

}

public static void Distance(int N,int [][]cost,int[]D)

{

int w,v,min;

boolean visited[]=new boolean[N];

D[0]=0;

for(v=1;v<N;v++){

visited[v]=false;

D[v]=cost[0][v];

}

for(int i=1; i<N; ++i){

min = INFINITY;

for(w=1; w<N; w++)

if(!visited[w])

if(D[w]<min) {

v = w;

min = D[w];

}

visited[v] = true;

for(w=1; w<N; w++)

if(!visited[w])

if(min+cost[v][w] < D[w])

D[w] = min + cost[v][w];

}

Prim’s..

#include <iostream>

#include <conio.h>

#define ROW 7

#define COL 7

#define infi 5000

using namespace std;

class prims

{

int graph[ROW][COL];

int nodes;

public:

prims();

void createGraph();

void primsAlgo();

};

prims :: prims(){

for(int i=0;i<ROW;i++)

for(int j=0;j<COL;j++)

graph[i][j]=0;

}

void prims :: createGraph(){

int i,j;

cout<<"Enter Total Nodes : ";

cin>>nodes;

cout<<"\n\nEnter Adjacency Matrix : \n";

for(i=0;i<nodes;i++){

for(j=0;j<nodes;j++)

cin>>graph[i][j];

}

//Assign infinity to all graph[i][j] where weight is 0.for(i=0;i<nodes;i++){

for(j=0;j<nodes;j++){

if(graph[i][j]==0)

graph[i][j]=infi;

}

}

void prims :: primsAlgo(){

int selected[ROW],i,j,ne;

int min,x,y;

for(i=0;i<nodes;i++)

selected[i]=false;

selected[0]=true;

ne=0;

while(ne < nodes-1){

min = infi;

for(i=0;i<nodes;i++)

{

if(selected[i]==true){

for(j=0;j<nodes;j++){

if(selected[j]==false){

if(min > graph[i][j])

{

min=graph[i][j];

x=i;

y=j;

}

}

}

}

}

selected[y]=true;

cout<<"\n"<<x+1<<" --> "<<y+1;

ne=ne+1;

}

}

int main(){

prims MST;

cout<<"\nPrims Algorithm to find Minimum Spanning Tree\n";

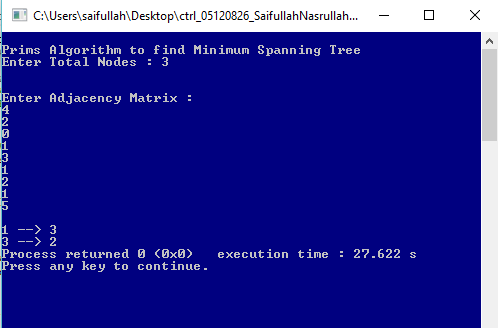
MST.createGraph();

MST.primsAlgo();

return 0;

}

***Ekran Çıktısı..***



**DFT Kaynak Kod**

public class DFS {

public static void main(String[] args) {

Graph theGraph= new Graph();

theGraph.addVertex('A'); // 0

theGraph.addVertex('B'); // 1

theGraph.addVertex('C'); // 2

theGraph.addVertex('D'); // 3

theGraph.addVertex('E'); // 4

theGraph.addEdge(0, 1); // AB

theGraph.addEdge(1, 2); // BC

theGraph.addEdge(0, 3); // AD

theGraph.addEdge(3, 4); // DE

System.out.print("Visits");

theGraph.dfs();

System.out.println();

}

class StackX {

private final int SIZE = 20;

private int[] st;

private int top;

public StackX(){

st = new int[SIZE];

top = -1;

}

public void push(int j){

st[++top]=j;

}

public int pop(){

return st[top--];

}

public int peek(){

return st[top];

}

public boolean isEmpty(){

return (top==-1);

}

}

class Vertex{

public char label;

public boolean wasVisited;

public Vertex(char lab){

label=lab;

wasVisited=false;

}

}

class Graph{

public final int MAX\_VERTS = 20;

public Vertex vertexList[];

public int adjMat[][];

public int nVerts;

public StackX theStack;

public Graph(){

vertexList = new Vertex[MAX\_VERTS]; // adjacency matrix

adjMat = new int[MAX\_VERTS][MAX\_VERTS];

nVerts = 0;

for(int j=0; j<MAX\_VERTS; j++) // set adjacency

for(int k=0; k<MAX\_VERTS; k++) // matrix to 0

adjMat[j][k] = 0; theStack = new StackX();

}

public void addVertex(char lab){

vertexList[nVerts++] = new Vertex(lab);

}

public void addEdge(int start, int end) {

adjMat[start][end] = 1;

adjMat[end][start] = 1;

}

public void displayVertex(int v) {

System.out.print(vertexList[v].label);

}

public void dfs(){

vertexList[0].wasVisited = true;

displayVertex(0);

theStack.push(0);

while( !theStack.isEmpty() ) {

int v = getAdjUnvisitedVertex( theStack.peek() );

if(v == -1)

theStack.pop();

else

{

vertexList[v].wasVisited = true;

displayVertex(v);

theStack.push(v);

}

}

for(int j=0; j<nVerts; j++){

vertexList[j].wasVisited = false;

}

}

public int getAdjUnvisitedVertex(int v) {

for(int j=0; j<nVerts; j++)

if(adjMat[v][j]==1 && vertexList[j].wasVisited==false)

return j; return -1;

}

}

**}**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Dijkstra’s SP** | Prim’s MST | BFT | DFT |
| Zaman Karmaşıklığı  Big-O Notasyonuna Göre | O(V2) | O(|V|2) | O(V+E) | O(|E|) |