daa lab

fib dp

```
#include<bits/stdc++.h>
using namespace std;
vector<int>v;
// d tc-o(n),sc=o(n)
int main(){
    cout<<" give the number"<<endl;
    int n;
    cin>>n;
    v.push_back(0);
    v.push_back(1);
    int dp[n+1];
    dp[0]=0,dp[1]=1;
    for(int i=2;i<=n;i++) {
        dp[i]=dp[i-1]+dp[i-2];
        v.push_back(dp[i]);
    }
    for(int i=0;i<n;i++){
        cout<<" "<<v[i+1];
    }
}
```

fib imperative

```
#include<bits/stdc++.h>
using namespace std;
vector<int>v;
//  imperative tc-o(n),sc=o(1)
void fib(int n){
     v.push_back(0);
    v.push_back(1);
    int a=0,b=1,c;
    for(int i=1;i<=n;i++){
        c=a+b;
        a=b;
        b=c;
        v.push_back(c);
    }
}
int main(){
    cout<<" give the number"<<endl;
    int n;
    cin>>n;
    fib(n);
    for(int i=0;i<n;i++){
        cout<<" "<<v[i+1];
    }
}
```

fib recurrsion

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
// tc o(2^n),sc o(n)
int fib(int n){
    if(n==0 or n==1)return n;
    return fib(n-1)+fib(n-2);
}
int main(){
    cout<<" give ther number"<<endl;
    int n;
    cin>>n;
    auto start=clock();
    for(int i=1;i<=n;i++){
        cout<<" "<<fib(i);
    }
    float t=clock()-start;
    t/=CLOCKS_PER_SEC;
    cout<<endl<<" time taken "<<setprecision(6)<<fixed<<t<<endl;
}
```

merge sort

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
void merge(vector<int>&v,int l,int m,int h){
    Sleep(1);
    vector<int>temp;
    int i=l,j=m+1;
    while(i<=m && j<=h){
        if(v[i]<=v[j]){
            temp.push_back(v[i]);
            i++;
        } else {
            temp.push_back(v[j]);
            j++;
        }
    }
    while(i<=m){
        temp.push_back(v[i]);
        i++;
    }
    while(j<=h){
        temp.push_back(v[j]);
        j++;
    }
    for(int k=l;k<=h;k++){
        v[k]=temp[k-l];
    }
}
void ms(vector<int>&v,int l,int h ){
    if(l>=h) return;
    int m=(l+h)/2;
    ms(v,l,m);
    ms(v,m+1,h);
    merge(v,l,m,h);
}
int main(){
    cout<<" give the number of elements to sort"<<endl;
    int n;
    cin>>n;
    vector<int>v(n);
    cout<<" give the number"<<endl;
    for(int i=0;i<n;i++){
        cin>>v[i];
    }
    auto start=clock();
    ms(v,0,n-1);
    float t=clock()-start;
    t/=CLOCKS_PER_SEC;
    for(int i=0;i<n;i++){
        cout<<" "<<v[i];
    }
    cout<<endl<<" time taken is "<<setprecision(6)<<fixed<<t<<endl;
}
```

quick sort

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
int part(vector<int>&v,int l,int h){
    Sleep(1);
    int pivot=v[l],i=l,j=h;
    while(i<j){
        while(v[i]<=pivot && i<h){
            i++;
        }
        while(v[j]>pivot && j>l){
            j--;
        }
        if(i<j) swap(v[i],v[j]);
    }
   swap(v[l],v[j]);
   return j;
}
void qs(vector<int>&v,int l,int h){
    if(l<h){
        int pi=part(v,l,h);
        qs(v,l,pi-1);
        qs(v,pi+1,h);
    }
}
int main(){
    cout<<" give the number of elements to sort"<<endl;
    int n;
    cin>>n;
    vector<int>v(n);
    cout<<" give the number"<<endl;
    for(int i=0;i<n;i++){
        cin>>v[i];
    }
    auto start=clock();
    qs(v,0,n-1);
    float t=clock()-start;
    t/=CLOCKS_PER_SEC;
    for(int i=0;i<n;i++){
        cout<<" "<<v[i];
    }
    cout<<endl<<" time taken is "<<setprecision(6)<<fixed<<t<<endl;
}
```

heap sort

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
void heapify(vector<int>&v,int n,int idx){
    Sleep(1);
    int l=2*idx+1,r=2*idx+2,mx=idx;
    if(l<n && v[l]>v[mx]){
        mx=l;
    }
    if(r<n &&v[r]>v[mx]){
        mx=r;
    }
    if(mx!=idx){
        swap(v[mx],v[idx]);
        //heapify for making heap
        heapify(v,n,mx);
    }
}
void hs(vector<int>&v,int n){
    //heapify for making heap
    for(int i=(n/2)-1;i>=0;i--){
        heapify(v,n,i);
    }
    //heapify for sort
    for(int i=n-1;i>0;i--){
        swap(v[0],v[i]);
        heapify(v,i,0);
    }
}
int main(){
    cout<<" give the number of elements to sort"<<endl;
    int n;
    cin>>n;
    vector<int>v(n);
    cout<<" give the number"<<endl;
    for(int i=0;i<n;i++){
        cin>>v[i];
    }
    auto start=clock();
    hs(v,n);
    float t=clock()-start;
    t/=CLOCKS_PER_SEC;
    for(int i=0;i<n;i++){
        cout<<" "<<v[i];
    }
    cout<<endl<<" time taken is "<<setprecision(6)<<fixed<<t<<endl;
}
```

Bfs traversal

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
int main(){
    cout<<" give the number of vertices and edges"<<endl;
    int n,e;
    cin>>n>>e;
    cout<<"give the graph"<<endl;
    vector<int>adj[n],vis(n,0),bfs;
    for(int i=0;i<e;i++){
        int u,v;
        cin>>u>>v;
        adj[u].push_back(v);
    }
    for(int i=0;i<n;i++){
        if(!vis[i]){
            queue<int>q;
            q.push(i);
            vis[i]=1;
            while(!q.empty()){
            int t=q.front();
            q.pop();
            bfs.push_back(t);
            for(auto it:adj[t]){
             if(!vis[it]){
                q.push(it);
                vis[it]=1;
                    }
                }
            }
        }
    }
    for(auto i:bfs) {
        cout<<" "<<i;
    }
}
```

dfs

```
#include<bits/stdc++.h>
using namespace std;
  void dfs(int node,vector<int>&vis,vector<int>adj[],vector<int>&ans){
    ans.push_back(node);
    vis[node]=1;
    for(auto it:adj[node]){
        if(!vis[it]){
            dfs(it,vis,adj,ans);
        }
    }
  }
int main (){
    int n,e;
    cout<<"Give number of vertices and edges"<<endl;
    cin>>n>>e;
    cout<<" give the graph "<<endl;
    vector<int>adj[n],vis(n,0);
    for(int i=0;i<e;i++){
        int u,v;
        cin>>u>>v;
        adj[u].push_back(v);
        //adj[v].push_back(u);
    }
    cout<<" dfs traversal is"<<endl;
    for(int i=0;i<n;i++){
        if(!vis[i]){
            vector<int>ans;
             dfs(i,vis,adj,ans);
             for(auto it:ans){
                cout<<it<<" ";
             }
             cout<<endl;
        }
    }

return 0;
}
```

0/1 knapsack

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
int main(){
    cout<<" give number of items and max weight"<<endl;
    int n,t,ans=0;
    cin>>n>>t;
    cout<<" give weight and value of items"<<endl;
    vector<vector<int>>item;
    int wt[n],val[n];
    for(int i=0;i<n;i++){
        cin>>wt[i]>>val[i];
    }
    vector<vector<int>>dp(n,vector<int>(t+1,0));
    auto start=clock();
    for(int i=wt[0];i<=t;i++){
        dp[0][i]=val[0];
    }
    for(int i=1;i<n;i++){
        Sleep(1);
        for(int cap=0;cap<=t;cap++){
            int nt=dp[i-1][cap];
            int take =INT_MIN;
            if(wt[i]<=cap) take=val[i]+dp[i-1][cap-wt[i]];
            dp[i][cap]=max(nt,take);
        }
    }
    float time=clock()-start;
    time/=CLOCKS_PER_SEC;
    cout<<" max value is "<<dp[n-1][t]<<endl;
    cout<<" total time taken"<<setprecision(6)<<fixed<<time<<endl;
 }
```

fractional kanpsack

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,t;
    cout<<"give number of items and total weight"<<endl;
    cin>>n>>t;
    vector<vector<int>>items;
    cout<<" give weight and value of items"<<endl;
    for(int i=0;i<n;i++){
        vector<int>wv;
        int w,v;
        cin>>w>>v;
        wv.push_back(w);
        wv.push_back(v);
        items.push_back(wv);
    }
    vector<pair<double,int>>vp;
    for(int i=0;i<n;i++){
        double v=items[i][1];
        double w=items[i][0];
        vp.push_back({v/w,i});
    }
    sort(vp.begin(),vp.end());
         double val=0;
         for(int i=n-1;i>=0;i--)
         {
             int index=vp[i].second;
             if(items[index][0]<t){
                 t=t-items[index][0];
                 val=val+items[index][1];
             }
             else{
                 val=val+(vp[i].first*t);
                 break;
             }
         }
    cout<<" total value is "<<val<<endl;
}
```

dj

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
vector<int>dj(vector<pair<int,int>>adj[],int src,int n){
    Sleep(1);
    vector<int>dist(n,INT_MAX);
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
    dist[src]=0;
    pq.push({0,src});
    while(!pq.empty()){
        auto t=pq.top();
        pq.pop();
        for(auto i:adj[t.second]){
            if(dist[i.first]>t.first+i.second){
                dist[i.first]=t.first+i.second;
                pq.push({dist[i.first],i.first});
            }
        }
    }
    return dist;
}
int main(){
   cout<<"give the number of vertices and edges"<<endl;
   int n,e;
   cin>>n>>e;
   cout<<" give the graph"<<endl;
   vector<pair<int,int>>adj[n];
   for(int i=0;i<e;i++){
    int u,v,w;
    cin>>u>>v>>w;
    adj[u].push_back({v,w});
   }
   auto start=clock();
   vector<int>dist=dj(adj,0,n);
   float t=clock()-start;
   t/=CLOCKS_PER_SEC;
   cout<<" node->"<<"   "<<"distance"<<endl;
   for(int i=0;i<n;i++){
    cout<<i<<"  "<<dist[i]<<endl;
   }
   cout<<endl;
    cout<<" time taken"<<setprecision(6)<<fixed<<t<<endl;
 }
```

prims

```
#include<bits/stdc++.h>
#include<windows.h>
using namespace std;
int main(){
    cout<<" give number of vertices and edges"<<endl;
    int v,e,ans=0;
    cin>>v>>e;
    cout<<"give the nodes and weights"<<endl;
    vector<pair<int,int>>adj[v];
    for(int i=0;i<e;i++){
        int u,V,w;
        cin>>u>>V>>w;
        adj[u].push_back({V,w});
    }
    vector<int>parent(v,-1),key(v,INT_MAX),mst(v,0);
    key[0]=0;
    auto start=clock();
     priority_queue<pair<int,int> ,vector<pair<int,int>> ,greater<pair<int,int>>> pq;
     pq.push({0,0});
    for(int i=0;i<v-1;i++){
        Sleep(1);
        int u=pq.top().second;
        pq.pop();
        mst[u]=1;
        for(auto it:adj[u]){
            int j=it.first;
            int w=it.second;
            if(mst[j]==0 && w<key[j]){
                parent[j]=u;
                pq.push({key[j],j});
                key[j]=w;
            }
        }
    }
    float t=clock()-start;
    t/=CLOCKS_PER_SEC;
    cout<<" parent"<<"  "<<"child"<<endl;
    for(int i=1;i<v;i++){
        cout<<parent[i]<<" ->"<<i<<endl;
        ans+=key[i];
    }
    cout<<"total mst weight is"<< " "<<ans<<endl;
    cout<<" time taken is "<<setprecision(10)<<fixed<<t<<endl;
    return 0;
}
```

bellman ford

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    cout<<" give number of vertices and edges"<<endl;
    int n,e,f=1;
    cin>>n>>e;
    cout<<" give the graph"<<endl;
    vector<vector<int>>adj;
    for(int i=0;i<e;i++){
        vector<int>temp;
        for(int j=0;j<3;j++){
            int x;
            cin>>x;
            temp.push_back(x);
        }
        adj.push_back(temp);
    }
    vector<int>dist(n,INT_MAX),pred(n,-1);
    //yhi par function bnana hai agar koi bhi source se lene ko kha jaye to dist[src]=0
hoga aur baki same
    dist[0]=0;
    //tb yha ek cnt lena padega jo ki node times hi chlne de loop for(i->n)ye wala loop
nhi rhega tb while(cnt<v)
    for(int i=0;i<n;i++){
        for(auto it:adj){
            int u=it[0],v=it[1],w=it[2];
            if(dist[u]+w<dist[v]){
                dist[v]=dist[u]+w;
                pred[v]=u;
            }
        }
    }
    for(auto i:adj){
        int u=i[0],v=i[1],w=i[2];
        if(dist[u]+w<dist[v] && dist[u]!=INT_MAX){
            f=0;
        }
    }
    if(f!=0){
        cout<<"from"<<"  "<<" to"<<" "<<"distance"<<endl;
        for(int i=0;i<n;i++){
            cout<<pred[i]<<"  "<<i<<" "<<dist[i]<<" "<<endl;;
        }
    } else {
        cout<<"negative cycle "<<endl;
    }
}
```

floyed warshall

```
#include<bits/stdc++.h>
using namespace std;
void print_path(int i,int j,vector<vector<int>>&p){
        if(i!=j) {
            print_path(i,p[i][j],p);
        }
        cout<<j<<" ";
    }
int main(){
    cout<<"give number of vertices and edges"<<endl;
    int n,e,f=1;
    cin>>n>>e;
    vector<vector<int>>mat(n,vector<int>(n,1e8)),p(n,vector<int>(n,-1));
    for(int i=0;i<n;i++){
        mat[i][i]=0;
    }
    cout<<" give the graph"<<endl;
    for(int i=0;i<e;i++){
        vector<int>temp(3);
        for(int j=0;j<3;j++)cin>>temp[j];
        mat[temp[0]][temp[1]]=temp[2];
        p[temp[0]][temp[1]]=temp[0];
             }
    for( int k=0;k<n;k++){
        for( int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(mat[i][k]+mat[k][j]<mat[i][j]){
                    mat[i][j]=mat[i][k]+mat[k][j];
                    p[i][j]=p[k][j];
                }
            }
        }
    }
    for(int i=0;i<n;i++){
        if(mat[i][i]<0) f=0;
    }
    if(f!=1) cout<<" negative cycle is present"<<endl;
    else {
    cout<<"cost matrix is "<<endl;
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cout<<mat[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<" predecissior matrix is "<<endl;
     for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cout<<p[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<" enter source and destination node"<<endl;
    int i,j;
    cin>>i>>j;
    print_path(i,j,p);
    }
}
```

krushkal algorithm

```
#include <bits/stdc++.h>
#include <windows.h>
using namespace std;
static int V;
bool cmp(vector<int> a, vector<int> b)
{
    return (a[2] < b[2]);
}
int findParent(vector<int> parent, int node)
{
    Sleep(1);
    if(parent[node]==node)
        return node;
    return (parent[node] = findParent(parent, parent[node]));
}
void unionSet(int u, int v, vector<int> &parent, vector<int> &rank)
{
    Sleep(1);
    u = findParent(parent, u);
    v = findParent(parent, v);
    if(rank[u] < rank[v])
        parent[u] = v;
    else if(rank[v] < rank[u])
        parent[v] = u;
    else
    {
        parent[v] = u;
        rank[u]++;
    }
}
void kruskalMST(vector<vector<int>> &edges, vector<vector<int>> &f_edges, int &ans)
{
    Sleep(1);
    sort(edges.begin(), edges.end(), cmp);
    vector<int> parent(V);
    vector<int> rank(V);
    for(int i=0; i<V ; i++)
    {
        parent[i]=i;
        rank[i]=0;
    }
    int minWeight = 0;
    for(int i=0; i<edges.size(); i++)
    {
        int u = findParent(parent, edges[i][0]);
        int v = findParent(parent, edges[i][1]);
        int wt= edges[i][2];
        if(u != v)
        {
            f_edges.push_back({edges[i][0],edges[i][1],wt});
            minWeight += wt;
            unionSet(u, v, parent, rank);
        }
    }
    ans=minWeight;
    return;
}
int main()
{
    int E;
    cout << "Enter the no. vertices and edges\n";
    cin >> V >> E;
    vector<vector<int>> edges;
    cout << "Enter the edges followed by its weight:\n";
    for (int i = 0; i < E; i++)
    {
        vector<int> edgewt(3);
        for (int j = 0; j < 3; j++)
            cin >> edgewt[j];
        edges.push_back(edgewt);
    }
    auto start = clock();
    vector<vector<int>> f_edges;
    int ans;
    kruskalMST(edges, f_edges, ans);
    cout << "\nEdge \tWeight\n";
  for (auto x:f_edges)
  {
        cout << x[0] << "  " << x[1] << " \t"
            << x[2] << " \n";
    }
    cout<<"Min MST Weight = "<<ans<<endl;
    float duration = clock() - start;
  duration = (duration) / CLOCKS_PER_SEC;
  cout << "Time taken by function: " << setprecision(10) << fixed << duration << "
seconds" << endl;
    return 0;
}
```