|  |
| --- |
| ***BFS*** |

#include<bits/stdc++.h>

using namespace std;

#define white 0

#define black 1

vector<int> vec[100010];

int level[100010],parent[100010];

int visited[100010];

int node,edge;

void bfs(int s)

{

for(int i=1;i<=node;i++)

visited[i]=white;

queue<int> q;

q.push(s);

visited[s]=black;

level[s]=0;

parent[s]=0;

while(!q.empty())

{

int u=q.front();

q.pop();

for(int i=0;i<vec[u].size();i++)

{

if(visited[vec[u][i]]==white)

{

level[vec[u][i]]=level[u]+1;

visited[vec[u][i]]=black;

q.push(vec[u][i]);

parent[vec[u][i]]=u;

}

}

}

}

int main()

{

int u,v,s;

cin>>node>>edge;

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

{

cin>>u>>v;

vec[u].push\_back(v);

vec[v].push\_back(u);

}

cin>>s;

bfs(s);

cout<<endl<<endl;

for(int i=1;i<=node;i++)

{

cout<<"LEVEL OF NODE "<<i<<" is "<<level[i]<<", parent is "<<parent[i]<<endl<<endl;

}

return 0;

}

|  |
| --- |
| ***DFS*** |

#include<bits/stdc++.h>

using namespace std;

#define white 0

#define black 1

int visited[1000000];

vector<int> vec[100000];

int dfs(int source)

{

int siz=vec[source].size();

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

{

int v=vec[source][i];

if(visited[v]==white)

{

visited[v]=black;

dfs(v);

}

}

}

int main()

{

int node,edge,u,v,source;

cin>>node>>edge;

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

{

cin>>u>>v;

vec[u].push\_back(v);

vec[v].push\_back(u);

}

cin>>source;

dfs(source);

return 0;

}

|  |
| --- |
| ***DFS for Competitive Programming*** |

#include<bits/stdc++.h>

using namespace std;

vector<int>vec[100010];

int visit[100010],n;

void DFS(int start)

{

if(visit[start]==1)

return;

visit[start]=1;

int siz=vec[start].size();

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

{

DFS(vec[start][i]);

}

}

int main()

{

int u,v,start;

cin>>n;

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

{

cin>>u>>v;

vec[u].push\_back(v);

vec[v].push\_back(u);

}

cin>>start;

DFS(start);

return 0;

}

|  |
| --- |
| ***DIJIKSTRA*** |

#include<bits/stdc++.h>

using namespace std;

vector<int> graph[100010],cost[100010];

int visited[100010],par[100010],node,edge;

int distanc[100010];

struct nod

{

int u, w;

nod(int a, int b) { u = a; w = b; }

bool operator <(const nod& p) const { return w>p.w; }

};

void dijstra(int source)

{

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

{

distanc[i]=99999999;

par[i]=-1;

}

distanc[source]=0;

priority\_queue<nod>pq;

pq.push(nod(source,0));

while(!pq.empty())

{

nod top=pq.top();

pq.pop();

int u=top.u;

///if(u==node-1) return;

for(int i=0;i<graph[u].size();i++)

{

int v=graph[u][i];

if(distanc[u]+cost[u][i]<distanc[v])

{

distanc[v]=distanc[u]+cost[u][i];

pq.push(nod(v,distanc[v]));

par[v]=u;

}

}

}

return;

}

int main()

{

int u,v,w,source;

cin>>node>>edge;

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

{

cin>>u>>v>>w;

graph[u].push\_back(v);

graph[v].push\_back(u);

cost[u].push\_back(w);

cost[v].push\_back(w);

}

cin>>source;

dijstra(source);

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

{

cout<<"node = "<<i<<" distanc = "<<distanc[i]<<" parent = "<<par[i]<<endl;

}

return 0;

}

|  |
| --- |
| ***Binary Search Update*** |

#include<bits/stdc++.h>

using namespace std;

int binarysearch(int low,int high,int \*a,int item)

{

int mid,res=-1;

while(low<=high)

{

mid=low+(high-low)/2;

if(a[mid]<item)

{

res=mid;

low=mid+1;

}

else

{

high=mid-1;

}

}

return res;

}

int binarysearch1(int low,int high,int \*a,int item)

{

int mid,res=-1;

while(low<=high)

{

mid=low+(high-low)/2;

if(a[mid]<=item)

{

res=mid;

low=mid+1;

}

else

{

high=mid-1;

}

}

return res;

}

int main()

{

int n,item;

cin>>n;

int a[n+1];

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

cin>>a[i];

sort(a,a+n);

cout<<"AFTER SORT THE ELEMENTS: "<<endl;

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

cout<<a[i]<<"\t";

cout<<endl;

cin>>item;

int res=binarysearch(0,n-1,a,item);

int res1=binarysearch1(0,n-1,a,item);

if(a[res1]!=item)

{

cout<<"THE ITEM IS NOT FOUND"<<endl;

return 0;

}

for(int i=res+1;i<=res1;i++)

cout<<i+1<<"\t";

return 0;

}

|  |
| --- |
| ***Convex Hull*** |

#include<bits/stdc++.h>

using namespace std;

double triangle(int x1,int x2,int x3,int y1,int y2,int y3)

{

double area;

int x=((x1\*y2)+(x2\*y3)+(x3\*y1));

int y=((x2\*y1)+(x3\*y2)+(x1\*y3));

int x\_y=x-y;

area=(1.0\*x\_y)/2;

return area;

}

int main()

{

int n;

cin>>n;

int x[n+1],y[n+1],xx[n+1],yy[n+1];

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

{

cin>>x[i]>>y[i];

}

double m[n+1];

int origin\_x=x[0],origin\_y=y[0];

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

{

if(x[i]<origin\_x)

{

origin\_x=x[i];

origin\_y=y[i];

}

}

int cnt=0;

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

{

if(x[i]==origin\_x&&y[i]==origin\_y)

;

else

{

m[cnt]=((y[i]-origin\_y)\*1.0)/(x[i]-origin\_x);

xx[cnt]=x[i];

yy[cnt]=y[i];

cnt++;

}

}

double temp;

int temp1,temp2,temp3,a,b,c;

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

{

for(int j=i+1; j<cnt; j++)

{

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

{

temp=m[i];

m[i]=m[j];

m[j]=temp;

temp1=xx[i];

xx[i]=xx[j];

xx[j]=temp1;

temp2=yy[i];

yy[i]=yy[j];

yy[j]=temp2;

}

}

}

stack<int> sx;

stack<int> sy;

sx.push(origin\_x);

sx.push(xx[0]);

sy.push(origin\_y);

sy.push(yy[0]);

for(int i=1; i<cnt; i++)

{

sx.push(xx[i]);

sy.push(yy[i]);

temp1=sx.top();

sx.pop();

temp2=sx.top();

sx.pop();

temp3=sx.top();

sx.pop();

a=sy.top();

sy.pop();

b=sy.top();

sy.pop();

c=sy.top();

sy.pop();

double res=triangle(temp1,temp2,temp3,a,b,c);

if(res>=0)

{

sx.push(temp1);

sx.push(temp2);

sx.push(temp3);

sy.push(a);

sy.push(b);

sy.push(c);

}

else

{

sx.push(temp1);

sx.push(temp3);

sy.push(a);

sy.push(c);

while(1)

{

int d=sx.top();

sx.pop();

int e=sx.top();

sx.pop();

int f=sx.top();

sx.pop();

int g=sy.top();

sy.pop();

int h=sy.top();

sy.pop();

int i=sy.top();

sy.pop();

double res1=triangle(d,e,f,g,h,i);

if(res1>=0)

{

sx.push(d);

sx.push(e);

sx.push(f);

sy.push(g);

sy.push(h);

sy.push(i);

break;

}

else

{

sx.push(d);

sx.push(f);

sy.push(g);

sy.push(i);

}

}

}

}

cout<<endl<<endl;

while(!sx.empty())

{

int q=sx.top();

int r=sy.top();

cout<<q<<" "<<r<<endl;

sx.pop();sy.pop();

}

return 0;

}

|  |
| --- |
| ***Factorial Mod*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

int number,mod,ans=1;

cout<<"PRESS THE FACTORIAL NUMBER: ";

cin>>number;

cout<<endl<<"PRESS THE MOD: ";

cin>>mod;

for(int i=1;i<=number;i++)

{

ans=(ans\*i)%mod;

}

cout<<endl<<"THE ANS IS = "<<ans<<endl;

return 0;

}

|  |
| --- |
| ***Fractional Knapsack*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n,temp1,m;

cin>>n>>m;

int weight[n+1],profit[n+1],total=0;

double ap[n+1],x[n+1]={0},temp,totalprofit=0;

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

{

cin>>profit[i]>>weight[i];

ap[i]=(1.0\*profit[i])/weight[i];

}

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

{

for(int j=i+1;j<n;j++)

{

if(ap[i]<ap[j])

{

temp=ap[i];

ap[i]=ap[j];

ap[j]=temp;

temp1=weight[i];

weight[i]=weight[j];

weight[j]=temp1;

}

}

}

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

{

if(total+weight[i]<=m)

{

total+=weight[i];

x[i]=1;

totalprofit+=weight[i]\*ap[i];

}

else{

double dif=m-total;

totalprofit+=dif\*ap[i];

total+=dif;

x[i]=dif/weight[i];

break;

}

}

cout<<"TOTAL PROFIT = "<<totalprofit<<endl;

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

{

cout<<"weight = "<<weight[i]<<" TAKE = "<<x[i]<<endl;

}

return 0;

}

|  |
| --- |
| ***GCD LCM*** |

#include<bits/stdc++.h>

using namespace std;

long long int gcd(long long int a,long long int b)

{

if(a%b==0)

return b;

else

return gcd(b,a%b);

}

int main()

{

long long int number1,number2,g,lcm;

cout<<"PRESS THE FIRST NUMBER: ";

cin>>number1;

cout<<endl<<"PRESS THE SECOND NUMBER: ";

cin>>number2;

g=gcd(number1,number2);

lcm=(((number1\*1.0)/g)\*number2);

cout<<endl<<"THE GCD IS = "<<g;

cout<<endl<<"THE LCM IS = "<<lcm;

return 0;

}

|  |
| --- |
| ***Job Sequence With Deadline*** |

#include<bits/stdc++.h>

using namespace std;

#define white 0

#define black 1

int cnt,total;

int main()

{

int n,temp,temp1;

cin>>n;

int dead[n+1],profit[n+1],seq[n+1]={0};

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

cin>>dead[i]>>profit[i];

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

{

for(int j=i+1;j<n;j++)

{

if(profit[i]<profit[j])

{

temp=profit[i];

profit[i]=profit[j];

profit[j]=temp;

temp1=dead[i];

dead[i]=dead[j];

dead[j]=temp1;

}

}

}

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

{

for(int j=dead[i]-1;j>=0;j--)

{

if(seq[j]==white)

{

seq[j]=black;

total+=profit[i];

cnt++;

break;

}

}

}

cout<<"PROFIT = "<<total<<endl;

cout<<"ITEM = "<<cnt<<endl;

return 0;

}

|  |
| --- |
| ***Prime Factorization*** |

#include<bits/stdc++.h>

using namespace std;

int all[10000000];

vector<int> vec;

void factorization(int number)

{

int root,root1;

root=sqrt(number)+1;

root1=sqrt(root)+1;

for(int i=3; i<=root1; i=i+2)

{

if(all[i]==0)

{

for(int j=i\*i; j<=root; j=j+2\*i)

all[j]=1;

}

}

vec.push\_back(2);

for(int i=3; i<=root; i+=2)

{

if(all[i]==0)

vec.push\_back(i);

}

}

int main()

{

int number,cnt=0,p=0,q=0;

vector<int> factor;

cin>>number;

factorization(number);

while(number>1&&vec.size()>cnt)

{

if(number%vec[cnt]==0)

{

number=number/vec[cnt];

factor.push\_back(vec[cnt]);

q=1;

}

else

cnt++;

if(number==1)

p=1;

}

for(int i=0; i<factor.size(); i++)

{

if(i!=factor.size()-1)

cout<<factor[i]<<" x ";

else

cout<<factor[i];

}

if(p==0)

{

if(q==1)

{

cout<<" x "<<number<<endl;

}

else

cout<<number<<endl;

}

return 0;

}

|  |
| --- |
| ***Merge Sort*** |

#include<bits/stdc++.h>

using namespace std;

int temp[100010];

void mergesort(int \*a,int low,int high)

{

if(low>=high)

return;

int mid=(low+high)/2;

mergesort(a,low,mid);

mergesort(a,mid+1,high);

int i=low;

int j=mid+1;

int index=0;

while(i<=mid&&j<=high)

{

if(a[i]<a[j])

{

temp[index]=a[i];

i++;

}

else

{

temp[index]=a[j];

j++;

}

index++;

}

while(i<=mid)

{

temp[index]=a[i];

i++;

index++;

}

while(j<=high)

{

temp[index]=a[j];

j++;

index++;

}

for(int i=low,index=0; i<=high; i++,index++)

{

a[i]=temp[index];

}

}

int main()

{

int n;

cin>>n;

int a[n+1];

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

cin>>a[i];

mergesort(a,0,n-1);

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

cout<<a[i]<<"\t";

return 0;

}

|  |
| --- |
| ***Quick Sort*** |

#include<bits/stdc++.h>

using namespace std;

void quicksort(int \*a,int low,int high)

{

if(low>=high)

return;

int i=low;

int j=high;

int pivot=a[low];

bool cnt=0;

while(i<j)

{

if(cnt==0)

{

if(pivot<=a[j])

{

j--;

}

else{

a[i]=a[j];

i++;

cnt=1;

}

}

else{

if(pivot>=a[i])

{

i++;

}

else{

a[j]=a[i];

j--;

cnt=0;

}

}

}

a[i]=pivot;

quicksort(a,low,i-1);

quicksort(a,i+1,high);

}

int main()

{

int n;

cin>>n;

int a[n+1];

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

cin>>a[i];

quicksort(a,0,n-1);

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

cout<<a[i]<<"\t";

return 0;

}

|  |
| --- |
| ***List*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

list<int>mylist;

list<int>::iterator it;

mylist.push\_back(10);

mylist.push\_back(15);

mylist.push\_back(19);

mylist.push\_back(12);

mylist.push\_front(56);

mylist.push\_front(5);

int siz=mylist.size();

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<" ";

cout<<endl<<endl;

mylist.reverse();

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<" ";

cout<<endl<<endl;

///mylist.clear();

///cout<<mylist.size();

cout<<endl<<endl;

list<int>lis(4,9);

for(it=lis.begin();it!=lis.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

cout<<"COPY"<<endl<<endl;

int a[5]={1,3,5,7,9};

list<int>b(a,a+5);

for(it=b.begin();it!=b.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

cout<<"INSERT"<<endl<<endl;

it=mylist.begin();

it++;

mylist.insert(it,7);

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

it=find(mylist.begin(),mylist.end(),19);

mylist.insert(it,18);

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

cout<<"DELETE......"<<endl<<endl;

it=mylist.begin();

mylist.erase(it);

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

it=find(mylist.begin(),mylist.end(),56);

mylist.erase(it);

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

cout<<"CHECK EMPTY"<<endl<<endl;

if(mylist.empty())

cout<<"EMPTY"<<endl;

else

cout<<"FILLED"<<endl;

cout<<endl<<endl;

cout<<"FIRST AND LAST ELEMENT"<<endl<<endl;

cout<<mylist.front()<<endl;

cout<<mylist.back()<<endl;

cout<<endl<<endl;

cout<<"DELETE FROM FIRST AND LAST"<<endl<<endl;

mylist.pop\_front();

mylist.pop\_back();

for(it=mylist.begin();it!=mylist.end();it++)

cout<<\*it<<"\t";

cout<<endl<<endl;

return 0;

}

|  |
| --- |
| ***Map*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

map<string,int>m;

map<string,int>::iterator it;

m["dipta"]=42;

m["arnob"]=45;

m["arnob"]=48;

m.insert(make\_pair("nabil",49));

printf("%d\n",m["arnob"]);

printf("%d\n",m["nabil"]);

cout<<endl<<endl;

it=m.begin();

cout<<it->first<<"\t"<<it->second<<endl;

it++;

cout<<it->first<<"\t"<<it->second<<endl;

cout<<endl<<endl;

cout<<"LOOP"<<endl<<endl;

for(it=m.begin();it!=m.end();it++)

cout<<it->first<<"\t"<<it->second<<endl;

return 0;

}

|  |
| --- |
| ***Pair*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

pair<string,int>p;

p.first="dipta";

p.second=45;

cout<<p.first<<" "<<p.second<<endl;

cout<<endl<<endl;

vector<pair<int,string> >vec;

vector<pair<int,string> >::iterator it;

vec.push\_back(make\_pair(21,"arnob"));

vec.push\_back(make\_pair(22,"nabil"));

cout<<vec[0].first<<" "<<vec[0].second<<endl;

cout<<vec[1].first<<" "<<vec[1].second<<endl;

cout<<endl<<endl;

cout<<"ITERATION.."<<endl<<endl;

for(it=vec.begin();it!=vec.end();it++)

cout<<it->first<<" "<<it->second<<endl;

return 0;

}

|  |
| --- |
| ***Set*** |

#include<bits/stdc++.h>

using namespace std;

int main()

{

set<string> s;

set<string> :: iterator it;

s.insert("nabil");

s.insert("arnob");

s.insert("mainul");

s.insert("sabbir");

for(it=s.begin();it!=s.end();it++)

cout<<\*it<<endl;

pair<set <string>::iterator,bool> p;

p=s.insert("akhi");

if(p.second==false)

cout<<"cannot insert"<<endl;

else

cout<<"CAN BE INSERTED"<<endl;

return 0;

}

|  |
| --- |
| ***Stack*** |

#include<bits/stdc++.h>

#include<stack>

using namespace std;

int main()

{

stack<string> s;

s.push("dipto");

s.push("nabil");

s.push("arnob");

s.push("mainul");

s.push("sabbir");

s.push("borhan");

s.pop();

cout<<s.top()<<endl;

cout<<endl<<endl;

while(!s.empty())

{

string x;

x=s.top();

cout<<x<<endl;

s.pop();

}

return 0;

}

|  |
| --- |
| ***STL*** |

#include<bits/stdc++.h>

using namespace std;

bool myfunc(int a,int b)

{

return (a>b);

}

int main()

{

vector<int>vec;

int n,a;

cin>>n;

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

{

cin>>a;

vec.push\_back(a);

}

sort(vec.begin(),vec.begin()+3);

for(int i=0;i<vec.size();i++)

cout<<vec[i]<<" ";

cout<<endl<<endl;

sort(vec.begin(),vec.end());

for(int i=0;i<vec.size();i++)

cout<<vec[i]<<" ";

cout<<endl<<endl;

sort(vec.begin(),vec.end(),myfunc);

for(int i=0;i<vec.size();i++)

{

cout<<vec[i]<<" ";

}

return 0;

}