Artful

int sumDiv(int n)

{

int sum = 1,n1 = n;

for(int k = 2; k\*k<=n ; ++k)

{

int p=1;

while(n % k == 0)

{

p = p\*k+1;

n /= k;

}

sum \*= p;

}

if(n>1)sum \*= 1+n;

//return sum; ///if all

return sum - n1; ///if proper

}

int countDiv(int n)

{

int cnt = 2, n1 = sqrt(n);

for (int i = 2; i<=n1 ; i++) {

if (n%i==0){

// If div are equal,

// count only one

if (n/i==i)cnt++;

//if (n==i\*i)cnt++;

else cnt = cnt + 2;// Otherwise count both

}

}

return cnt;

}

void printDiv(int n)

{

int n1 = sqrt(n);

for (int i=2; i<=n1; i++)

{

if (n%i == 0)

{

if (n/i == i)cout<<i<<' ';

else cout<<i<<' '<<n/i<<' ';

}

}

}

void sieve(bool prime[],int n)

{

for(int p=2 ; p\*p<=n ; p++)

{

if(prime[p])

{

for(int i=p\*p ; i<=n; i+=2\*p)prime[i] = false;

}

}

}

int main()

{

bool prime[101]; ///declare N+1

memset(prime,true,sizeof(prime));

sieve(prime,101);

for(int i = 2;i <= 100; i++)if(prime[i])cout<<i<<" "; ///run till N

return 0;

}



divFromAtoBbyK()

{

long long a,b,k,ans=0;

cin>>k>>a>>b;

if(a<=0&&b>=0)

{

a =- a;

ans = 1+(a/k)+(b/k);

}

else if(a<=0&&b<=0)

{

a=-a;

b=-b;

swap(a,b);

ans=(b/k)-(a-1)/k;

}

else ans=(b/k)-(a-1)/k;

cout<<ans<<endl;

}

rndm()

{

srand(time(0)); ///assign for randomness

fflush(stdin);

r = (rand()% to) + from;

}

bool bSeach(int \*p, int &x,int size)

{

int first = 0, last = size-1, mid;

gap

while(first <= last)

{

mid = first + (last - first) /2; ///{21, 24 ,41 , 47, 84 ,96};

if(x > p[mid])first = mid + 1;

else if(x < p[mid])last = mid - 1;

else

{

x = mid;

return true;

}

}

return false;

}

int main()

{

int h, i=0, v[] = {21, 24,41, 47, 84,96};

sort(v,v+6);

h = 84;

if(bSeach(v,h,6))cout<< " Found at index " << h;

else cout<< " Not found.";

return 0;

}

int upper\_bound(int ar[], int n, int sz)

{

int l = 0, h = sz - 1, mid;

while(l <= h)

{

mid = l + (h - l)/2;

if(n < ar[mid])h = mid - 1;

else l = mid + 1;

}

return l;

}

int lower\_boundd(int ar[], int n, int sz)

{

int l = 0, h = sz - 1, mid;

while(l <= h)

{

mid = l + (h - l)/2;

if(n <= ar[mid])h = mid - 1;

else l = mid + 1;

}

return l;

}

int main()

{

srand(time(0));

int r, ar[7] = {21 ,22 ,26 ,28 ,33 ,35 ,37}, to = 0, from = 7;

//sort(ar,ar+7);

printAr(ar);

int f;

cin>>f;

if(f == \*find(ar,ar+7,f))

{

cout<<" UpperBound: "<<upper\_bound(ar,f,7)<<" Values are smaller then (including) "<<f<<endl;

cout<<" LowerBound: "<<lower\_boundd(ar,f,7)<<" Values are smaller then "<<f<<endl;

}

else

{

cout<< "Not found.!!\n";

cout<<" UpperBound: "<<upper\_bound(ar,f,7)<<" Values are smaller then (including) "<<f<<endl;

cout<<" LowerBound: "<<lower\_boundd(ar,f,7)<<" Values are smaller then "<<f<<endl;

}

int \*p = upper\_bound(ar,ar+7,f);

cout<<\*p;

p = lower\_bound(ar,ar+7,f);

cout<<\*p;

return 0;

}

1. DJK

typedef pair<int, int> pii;

vector<pii> graph[MAX];

int dist[MAX];

using namespace std;

void djxt(int s, int t, int n)

{

int crntN,crntC, nxt, wgt;

memset(dist,-1,sizeof(dist));

priority\_queue<pii,vector<pii>,greater<pii>> pq;

pii pr;

pr.first = 0; ///distance

pr.second = s; ///node

pq.push(pr);

while(!pq.empty())

{

pr = pq.top();

crntN = pr.second;

crntC = pr.first;

pq.pop();

if(dist[crntN] < 0)

{

dist[crntN] = crntC;

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

{

nxt = graph[crntN][i].first; ///node

wgt = graph[crntN][i].second; ///cost

if(dist[nxt] < 0)

{

pq.push(make\_pair(dist[crntN]+wgt, nxt));

}

}

}

}

if(dist[t] == -1)cout<<"Case #"<<n<<": unreachable" ;

else cout<<"Case #"<<n<<": "<<dist[t];

gap;

}

int main()

{

int i, n, e, u, v, w, s, t, N, N1;

cin>> N;

N1 = N;

while( N-- )

{

cin>> n >> e >> s >> t;

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

{

cin>> u >> v >> w ;

graph[u].push\_back(make\_pair(v,w));

graph[v].push\_back(make\_pair(u,w));

}

djxt(s,t, N1 - N);

for(i = 0; i<n; i++)graph[i].clear();

}

return 0;

}

1. DFS

int visit[MAX];

vector<int> graph[MAX];

void dfs(int n)

{

visit[n] = 1;

for(int v, i = 0; i< graph[n].size(); i++)

{

v = graph[n][i];

if(!visit[v])

{

visit[v] = 1;

dfs(v);

}

}

}

int main()

{

int n, e, u,v, c = 0;

cin>>n>>e;

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

{

cin>>u>>v;

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

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

}

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

{

if(!visit[i])

{

visit[i] = 1;

dfs(i);

c++;

}

}

cout<<c;

return 0;

}

void bfs(int n)

{

int visited[MAX], x, y;

memset(visited,0,sizeof(visited));

queue<int> q;

q.push(1);

visited[1] = 1;

while(!q.empty())

{

x = q.front();

q.pop();

for(int i = 0; i<edge[x].size(); i++)

{

y = edge[x][i];

if(!visited[y])

{

q.push(y);

visited[y] = 1;

level[y] = level[x] + 1;

}

}

}

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

{

cout<<1<<" to "<<i<<" : "<<level[i];

gap

}

}

int main()

{

int i, j, n ,e ,h1, h2;

cin>> n >> e;

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

{

cin>> h1 >> h2;

edge[h1].push\_back(h2);

edge[h2].push\_back(h1);

}

gapp

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

{

cout<< " Node " << i<< " is connected with: ";

h1 = edge[i].size();

for(j = 0; j < h1; j++ )cout<<edge[i][j] << " ";

gap

}

gapp

gapp

bfs(n);

return 0;

}