Fibonacci matrics multiplication

#include <stdio.h>

void multiply(int F[2][2], int M[2][2]);

void power(int F[2][2], int n);

/\* function that returns nth Fibonacci number \*/

int fib(int n)

{

  int F[2][2] = {{1,1},{1,0}};

  if (n == 0)

    return 0;

  power(F, n-1);

  return F[0][0];

}

/\* Optimized version of power() in method 4 \*/

void power(int F[2][2], int n)

{

  if( n == 0 || n == 1)

      return;

  int M[2][2] = {{1,1},{1,0}};

  power(F, n/2);

  multiply(F, F);

  if (n%2 != 0)

     multiply(F, M);

}

void multiply(int F[2][2], int M[2][2])

{

  int x =  F[0][0]\*M[0][0] + F[0][1]\*M[1][0];

  int y =  F[0][0]\*M[0][1] + F[0][1]\*M[1][1];

  int z =  F[1][0]\*M[0][0] + F[1][1]\*M[1][0];

  int w =  F[1][0]\*M[0][1] + F[1][1]\*M[1][1];

  F[0][0] = x;

  F[0][1] = y;

  F[1][0] = z;

  F[1][1] = w;

}

/\* Driver program to test above function \*/

int main()

{

  int n = 9;

  printf("%d", fib(9));

  getchar();

  return 0;

}

BINOMIAL EXPONENTION By Monmoy

#include <bits/stdc++.h>

using namespace std;

int func(int a,int b)

{

if(b == 1)

return a;

if(b%2!=0)

return a\*func(a,b-1);

else

{

int x = func(a,b/2);

return x\*x;

}

}

int main()

{

int a,b,c;

scanf("%d %d",&a,&b);

cout<<func(a,b);

return 0;

}

GCD LCM

int gcd (int a, int b) {

return b ? gcd (b, a % b) : a;

}

int lcm (int a, int b) {

return a / gcd (a, b) \* b;

}

SIEVE BY Monmoy

void sieve()

{

bt.flip();

bt[0]=0,bt[1]=0;

prime.push\_back(2);

for(ll i = 3 ; i <= MAX; i+=2)

{

if(bt[i])

{

for(ll j = i\*i; j <= MAX; j+=i+i)

bt[j]=0;

prime.push\_back(i);

}

}

}

Eular Phi By Monmoy

ll eularphi(ll m)

{

int index=0,pf=prime[index],ans=m;

while(pf\*pf<=m)

{

if(m%pf==0)

ans-=ans/pf;

while(m%pf==0)

m/=pf;

pf = prime[++index];

}

if(m!= 1)

ans-=ans/m;

return ans;

}

Factorization By Monmoy

ll eularphi(ll m)

{

int index=0,pf=prime[index],ans=m;

while(pf\*pf<=m)

{

while(m%pf==0)

{

m/=pf;

mnmy.push\_back(pf);

}

pf = prime[++index];

}

if(m!= 1)

mnmy.push\_back(m);

}

LONGEST COMMON SUBSEQUENCE

int lcs( char \*X, char \*Y, int m, int n )

{

   int L[m+1][n+1];

   int i, j;

   /\* Following steps build L[m+1][n+1] in bottom up fashion. Note

      that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] \*/

   for (i=0; i<=m; i++)

   {

     for (j=0; j<=n; j++)

     {

       if (i == 0 || j == 0)

         L[i][j] = 0;

       else if (X[i-1] == Y[j-1])

         L[i][j] = L[i-1][j-1] + 1;

       else

         L[i][j] = max(L[i-1][j], L[i][j-1]);

     }

   }

   /\* L[m][n] contains length of LCS for X[0..n-1] and Y[0..m-1] \*/

   return L[m][n];

}

/\* Utility function to get max of 2 integers \*/

int max(int a, int b)

{

    return (a > b)? a : b;

}

FLOYED WARSHELL ALGO

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

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

ara[i][j]=inf;

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

ara[i][i]=0;

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

{

scanf("%d %d", &c, &d);

ara[c][d]=ara[d][c]=1;

}

for(int k = 0; k < b; k++)

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

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

ara[i][j]=min(ara[i][j],ara[i][k]+ara[k][j]);

Disjoint Set Data Structure

class uf

{

vector<int>p,setsize,rank;

int disjoint;

public:

uf(int n)

{

disjoint = n;

setsize.assign(n,1);

rank.assign(n,0);

p.assign(n,0);

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

p[i]=i;

}

int findset(int i)

{

return (i == p[i]) ? i : findset(p[i]);

}

bool issameset(int i, int j)

{

return findset(i)==findset(j);

}

void unionset(int i , int j)

{

int x = findset(i), y=findset(j);

if(x != y)

{

if(rank[x] > rank[y])

{

p[y]=x;

setsize[x]+=setsize[y];

}

else

{

p[x]=y;

setsize[x]+=setsize[y];

if(rank[x] == rank[y])

rank[y]++;

}

disjoint--;

}

}

int setSize(int i) {

return setsize[findset(i)];

}

};

Kruskal Algo

vector<pair<int,pair<int,int> > > mnmy;

uf dis(n);

mnmy.push\_back(make\_pair(c,pair<int,int>(mp[str],mp[str1])))

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

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

{

pair<int,pair<int,int> > x = mnmy[i];

if(!dis.issameset(x.second.first,x.second.second))

{

sum+=x.first;

dis.unionset(x.second.first,x.second.second);

}

Here dis is disjoint set class

Djiksra

#include <bits/stdc++.h>

#define ll long long int

#define inf ((ll)1)<<61

#define uu first

#define vv second

using namespace std;

typedef pair<int,int> ii;

typedef vector<ii>vii;

typedef vector<int>vi;

vector<vii>mnmy;

class dijkstra

{

vi parent,dist,visited;

int n;

priority\_queue<ii,vector<ii>,greater<ii> >q;

public:

dijkstra(int a)

{

n=a;

dist.assign(n,inf);

mnmy.assign(n,vii());

parent.assign(n,-1);

visited.assign(n,0);

}

void calculation(int a)

{

int count = 0;

dist[a]=0;

q.push(ii(0,a));

while(!q.empty())

{

int v = q.top().uu;

int u = q.top().vv;

q.pop();

if(v>dist[u])

continue;

if(visited[u])

continue;

visited[u]=1;

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

{

ll x = mnmy[u][i].uu;

ll y = mnmy[u][i].second;

if(y+dist[u] < dist[x])

{

dist[x]=y+dist[u];

parent[x]=u;

if(!visited[x])

{

q.push(ii(dist[x],x));

}

}

}

}

}

ll result(int a)

{

return dist[a];

}

int path()

{

vector<ll>out;

int temp = n-1;

while(temp != -1)

{

out.push\_back(temp);

temp = parent[temp];

}

int count = out.size()-1;

for(int i = count; i >= 0; i--)

printf("%d ",out[i]);

printf("\n");

}

};

Second Best Mst kruskal

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

{

pair<int,pair<int,int> > x = mnmy[i];

if(!dis.issameset(x.second.first,x.second.second))

{

sum1+=x.first;

dis.unionset(x.second.first,x.second.second);

solution.push\_back(i);

}

}

ll secondbest=1e7;

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

{

uf ds(m+1);

ll current=0;

for(int j = 0; j < mnmy.size(); j++)

{

if(solution[i] == j)

continue;

pair<int,pair<int,int> > x = mnmy[j];

if(!ds.issameset(x.second.first,x.second.second))

{

current+=x.first;

ds.unionset(x.second.first,x.second.second);

}

}

secondbest = min(current,secondbest);

}

Topological Sort

Using indegree

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

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

{

if(indegree[i] == 0)

{

pq.push(ii(0,i));

}

}

vector<int>result;

while(!pq.empty())

{

int x = pq.top().first,y = pq.top().second;

result.push\_back(y),pq.pop();

for(int i = 0; i < connect[y].size(); i++)

{

int mn = connect[y][i];

indegree[mn]--;

if(indegree[mn] == 0)

pq.push(ii(0,mn));

}

}

Kadane’s algo for finding maximum sum subarray

struct abc

{

int fst;

int lst;

int sum;

}result,max\_so\_far;

void kadane(int a[],int n)

{

max\_so\_far.sum=0;

max\_so\_far.fst=1;

max\_so\_far.lst=-1;

result.sum = -inf;

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

{

max\_so\_far.sum+=a[i];

if(max\_so\_far.sum < 0)

{

max\_so\_far.fst=i+2; //next one + 1 for index start from 1

max\_so\_far.sum=0;

}

max\_so\_far.lst = i+1;

if(max\_so\_far.sum > result.sum)

{

result.sum = max\_so\_far.sum;

result.fst = max\_so\_far.fst;

result.lst = max\_so\_far.lst;

}

}

}