匈牙利算法

int vis[1010],link[1010],a[1010][1010],n,m,k,x,y,sum;

bool dfs(int x) {

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

if (a[x][i] && !vis[i]){

vis[i] = 1;

if (!link[i] || dfs(link[i])) {link[i] = x;return 1;}

}

return 0;

}

int main() {

scanf("%d%d%d", &n, &m, &k);

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

scanf("%d%d", &x, &y);

a[x][y] = 1;

}

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

memset(vis,0,sizeof(vis));

if (dfs(i)) sum++;

}

printf("%d\n",sum);

return 0;

}

输入输出优化

int buf[15];

void writeln(int i){

int p = 0;

if (I < 0) {putchar('-'); I = -i;}

while (i) {buf[p++] = I % 10; I /= 10;}

while (p) putchar('0' + buf[--p]);

putchar('\n');

}

int read(){

char c;int f = 1, x = 0;

while (c=getchar(),(c < '0' || c > '9') && c != '-');

if (c == '-') f = -1;else x = c-'0';

while (c = getchar(),c >= '0' && c <= '9') x = x\*10+c-'0';

return f \* x;

}

int main() {

int i;

while (I = read()) writeln(i);

}

树状数组求逆序对

const int N = 500005;

struct Node {int val,pos;}node[N];

int c[N], reflect[N], n;long long ans;

int compare(const Node& a,const Node& b) {

return a.val < b.val;

}

int lowbit(int x) {

return x & (-x);

}

void update(int x) {

while (x <= n){

c[x]++;

x += lowbit(x);

}

}

int getsum(int x) {

int sum = 0;

while (x > 0) {

sum += c[x];

x -= lowbit(x);

}

return sum;

}

int main()

{

scanf("%d", &n);

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

scanf("%d", &node[i].val);

node[i].pos = i;

}

sort(node+1, node+n+1, compare);

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

reflect[node[i].pos] = i;

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

update(reflect[i]);

ans += I - getsum(reflect[i]);

}

cout<<ans;

}

**Dinic**

const int maxn=250;

struct node{int u,v,next,c;}edge[maxn<<1];

int head[maxn];int dis[maxn];

int cnt;int n,m;int ans;

void add(int a,int b,int c){

edge[cnt].u=a;edge[cnt].v=b;

edge[cnt].c=c;edge[cnt].next=head[a];

head[a]=cnt++;

}

int bfs()// 给各点分层，离源点的远近分

{

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

queue<int> q;

dis[1]=0;q.push(1);

int i;int cur;

while(!q.empty()){

cur = q.front();q.pop();

for (i=head[cur];i != -1;i = edge[i].next){

if(dis[edge[i].v] == -1 && edge[i].c > 0){

dis[edge[i].v] = dis[cur] + 1;

q.push(edge[i].v);

}

}

}

if (dis[m] < 0) return 0;

return 1;

}

int Find (int x,int low){//增广

if (x==m) return low;int a;

for (int i=head[x];i!=-1;i=edge[i].next) {

int v=edge[i].v;

if (dis[v] == dis[x]+1 && edge[i].c > 0 && (a = Find(v,min(low,edge[i].c)))) {

edge[i].c -= a;

edge[i^1].c += a;

return a;

}

}

return 0;

}

void dinic() {

int temp;

while(bfs()) {

temp = Find(1,0x3f3f3f3f);

ans += temp;

}

printf("%d\n", ans);

}

int main() {

while(scanf("%d%d",&n,&m)) {

int a,b,flow;cnt = ans = 0;

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

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

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

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

add(a,b,flow);

add(b,a,0);

}

dinic();

}

return 0;

}

中位数

#define ins(x) ((x)<mid ? L.push(x) : R.push(x))

using namespace std;

priority\_queue <int> L;

priority\_queue <int, vector<int>, greater<int> > R;

int n,x,y,mid;

int main(){

scanf("%d%d",&n,&x);

printf("%d\n",mid=x);

R.push(x);

for (int i=1;i<=((n-1)>>1);i++){

scanf("%d%d",&x,&y);

ins(x);ins(y);

while (L.size()>=R.size()){R.push(L.top());L.pop();}

while (R.size()-1>L.size())

{L.push(R.top());R.pop();}

printf("%d\n",mid=R.top());

}

return 0;

}

线段树—区间修改

maxv[maxn\*5],minv[maxn\*5],sumv[maxn\*5], addv[maxn\*5],a[maxn];

void build (int o, int L, int R){

int lc = o\*2, rc = o\*2+1; sumv[o] = 0;

if (R==L) sumv[o] = addv[o] = a[L];

else {

int M = L + (R-L)/2;

build(lc, L, M);build(rc, M+1, R);

sumv[o] = sumv[lc] + sumv[rc];

}}

int y1,y2;//[y1,y2]

void maintain(int o, int L, int R) {

int lc = o\*2, rc = o\*2+1;

sumv[o] = minv[o] = maxv[o] = 0;

if (R > L) {

sumv[o] = sumv[lc] + sumv[rc];

minv[o] = min(minv[lc], minv[rc]);

maxv[o] = max(maxv[lc], maxv[rc]);

}

minv[o]+=addv[o];maxv[o]+=addv[o]; sumv[o] += addv[o] \* (R-L+1);

}

int v;

void update(int o, int L, int R) {

int lc = o\*2, rc = o\*2+1;

if (y1 <= L && y2 >= R)

addv[o] += v;

else { int M = L + (R-L)/2;

if (y1 <= M) update(lc, L, M);

if (y2 > M) update(rc, M+1, R);

}

maintain(o, L, R);

}

int \_min, \_max, \_sum;

void query(int o, int L, int R, int add){

if (y1 <= L && y2 >= R) {

\_sum += sumv[o] + add \* (R-L+1);

\_min = min(\_min, minv[o] + add);

\_max = max(\_max, maxv[o] + add);

} else { int M = L + (R-L)/2;

if (y1 <= M) query(o\*2, L, M, add + addv[o]);

if (y2 > M) query(o\*2+1, M+1, R, add + addv[o]);

}}

线段树—点修改

const int maxn=1000000;

int minv[maxn\*5],n;

int ql,qr;//query min:[ql,qr]

int query(int o, int L, int R){

int M = L + (R-L)/2,ans=0x7ffffff;

if (ql<=L && qr>=R) return minv[o];

if (ql<=M) ans=min(ans, query(o\*2, L, M));

if (qr>M) ans=min(ans, query(o\*2+1, M+1, R));

return ans;

}

int p,v;//a[p]=v;

void update(int o, int L, int R){

int M = L + (R-L)/2;

if (L==R) minv[o]=v;

else{

if (p<=M) update(o\*2, L, M);

else update(o\*2+1, M+1, R);

minv[o]=min(minv[o\*2], minv[o\*2+1]);

}

}

int main(){

scanf("%d",&n);

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

scanf("%d%d",&p,&v);update(1,1,n);

}

}

二分图染色

int color[maxn];

bool bipartite(int u) {

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

int v = G[u][i];

if (color[v] == color[u]) return false;

if (!color[v]) {

color[v] = 3 - color[u];

if (!bipartite(v)) return false;

}

}

return true;

}

倍增LCA

int p[500010][50],deep[500010],n,m,k,A,B;

int next[1000010],h[1000010],v[1000010];

void dfs(int x){

for (int i=h[x];i>0;i=next[i])

if (deep[v[i]]==-1)

{deep[v[i]]=deep[x]+1;

p[v[i]][0]=x;

dfs(v[i]);

}

}

int lca(int x,int y){

int t;

if (deep[x]<deep[y]) swap(x,y);

for (t=0;(1<<t)<=deep[x];t++); t--;

if (deep[x]!=deep[y])

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

if (deep[x]-(1<<i)>=deep[y]) x=p[x][i];

if (x==y) return x;

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

if (p[x][i] && (p[x][i]-p[y][i])) {x=p[x][i];y=p[y][i];}

return p[x][0];

}

int main(){

scanf("%d%d%d",&n,&m,&k);

for (int i=1;i<n\*2-1;i+=2)

{scanf("%d%d",&v[i+1],&v[i]);

next[i]=h[v[i+1]];h[v[i+1]]=i;

next[i+1]=h[v[i]];h[v[i]]=i+1;

}

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

deep[k]=0;

dfs(k);

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

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

if (p[i][j-1]!=-1) p[i][j]=p[p[i][j-1]][j-1];

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

{scanf("%d%d",&A,&B);

printf("%d\n",lca(A,B));

}

}

kruskal

struct Edge{int u,v,w;}a[1000005];

int fa[100005],n,m,ans,cnt;

int compare(Edge t,Edge b){

return t.w<b.w;

}

int found(int x){

if (fa[x]==x) return x;

return fa[x]=found(fa[x]);

}

int main(){

scanf("%d%d",&n,&m);

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

scanf("%d%d%d",&a[i].u,&a[i].v,&a[i].w);

sort(a+1,a+m+1,compare);

for (int i=1;i<=n;i++) fa[i]=i;

cnt=n;

for (int i=1;i<=m && cnt>1;i++)

{int x=found(a[i].u);

int y=found(a[i].v);

if (x!=y) {fa[x]=y;ans+=a[i].w;cnt--;}

}

printf("%d",ans);

}

卡特兰数

int catalan[maxn];int n;

int main()

{

catalan[0]=catalan[1]=1;

scanf("%d",&n);

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

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

{catalan[i]+=catalan[k]\*catalan[i-k-1];}

printf("%d",catalan[n]);

}

exgcd

//求整数x,y 使得 ax+by=d 且 |x|+|y| 最小

//d=gcd(a,b)

void gcd(LL a, LL b, LL& d, LL& x, LL& y) {

if (!b) {d=a; x=1; y=0;}

else {gcd(b, a%b, d, y, x);y-=x\*(a/b);}

}

欧拉筛法

const int maxn=10000005;

int prime[maxn],n,m,x,tot;bool flag[maxn];

void euler(int n) {

memset(flag,0,sizeof flag);

tot=0;flag[1]=true;

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

{if (!flag[i]) prime[++tot]=i;

for (int j=1;i\*prime[j]<=n;j++)

{flag[i\*prime[j]]=true;

if (i%prime[j]==0) break;

}

}

}

int main(){

scanf("%d%d",&n,&m);

euler(n);

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

{scanf("%d",&x);

printf(flag[x] ? "No\n" : "Yes\n");

}

}

欧拉函数求和

int phi[10000005];int n;

long long sum\_phi(int x){

memset(phi,0,sizeof phi);

phi[1]=1;

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

if (!phi[i]) for (int j=i;j<=n;j+=i)

{if (!phi[j]) phi[j]=j;

phi[j]=phi[j]/i\*(i-1);

}

long long ans=0;

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

ans+=phi[i];

return ans;

}

int main(){

scanf("%d",&n);

cout<<sum\_phi(n);

}

欧拉函数

int euler\_phi(int n)

{

int m=(int)sqrt(n+0.5);

int ans=n;

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

if (n % i==0) {ans=ans/i\*(i-1);

while (n%i==0) n/=i;

}

if (n>1) ans=ans/n\*(n-1);

return ans;

}

int main()

{

int n;

scanf("%d",&n);

printf("%d", euler\_phi(n));

}

最小费用最大流

const int maxn=50010; const int maxm=500010;

const int inf=2000000010;

int sumflow,tot,mincost,n,m,s,t;

struct Edge{int u,v,cap,cost,next;}edge[maxm<<2];

int head[maxn],dis[maxn],pp[maxn],vis[maxn];

void addedge(int u,int v,int cap,int cost){

edge[tot].u=u;edge[tot].v=v;edge[tot].cap=cap;

edge[tot].cost=cost;edge[tot].next=head[u];

head[u]=tot++;

}

bool spfa(){

queue<int> q;int cur,x;

memset(vis,0,sizeof(vis));

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

for (int i=1;i<=n;i++) dis[i]=inf;

vis[s]=1;dis[s]=0;

q.push(s);

while (!q.empty()){

cur=q.front();q.pop();vis[cur]=0;

for (int i=head[cur];i!=-1;i=edge[i].next){

x=edge[i].v;

if(edge[i].cap&&dis[x]>dis[cur]+edge[i].cost){

dis[x]=dis[cur]+edge[i].cost;

pp[x]=i;

if (!vis[x]) {q.push(x);vis[x]=1;}

}

}

}

if (dis[t]==inf) return false;

return true;

}

void MCMF(){

int minflow,flow=0;

while (spfa()){

minflow=inf+1;

for (int i=pp[t];i!=-1;i=pp[edge[i].u])

minflow=min(minflow,edge[i].cap);

flow+=minflow;

for (int i=pp[t];i!=-1;i=pp[edge[i].u]){

edge[i].cap-=minflow;

edge[i^1].cap+=minflow;

}

mincost+=dis[t]\*minflow;

}

sumflow=flow;

}

int main(){

int u,v,cap,cost;

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

scanf("%d%d%d%d",&n,&m,&s,&t);

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

scanf("%d%d%d%d",&u,&v,&cap,&cost);

addedge(u,v,cap,cost);

addedge(v,u,0,-cost);

}

MCMF();

printf("%d %d",sumflow,mincost);

}

**KMP**

#define s1(i) s1[i-1]

#define s2(i) s2[i-1]

using namespace std;

void match(){

int j=0;

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

while (j && s2(j+1)!=s2(i)) j=p[j];

if (s2(j+1)==s2(i)) j++;

p[i]=j;

}

}

void kmp(){

int j=0;

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

while (j && s2(j+1)!=s1(i)) j=p[j];

if (s2(j+1)==s1(i)) j++;

if (j==m) {printf("%d\n",i-m+1);j=p[j];}

}

}

SPFA

#define maxn 2505

#define maxm 1000005

int v[maxm],next[maxm],w[maxm];

int h[maxn],dis[maxn];

int q[maxm\*2],head,tail;int n,m,t,p;

int main(){

scanf("%d%d",&n,&m);

for (int i=2;i<=n;i++) dis[i]=maxm;

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

scanf("%d%d%d",&t,&v[i],&w[i]);

next[i]=h[t];h[t]=i;

v[m+i]=t;w[m+i]=w[i];

next[m+i]=h[v[i]];h[v[i]]=m+i;

}

q[head]=1;tail++;

while (tail>head){

t=q[head++];p=h[t];

while (p){

if ((dis[t]+w[p])<dis[v[p]]){

dis[v[p]]=dis[t]+w[p];q[tail++]=v[p];

}

p=next[p];

}

}

printf("%d",dis[n]);

}

对拍

#include<cstdlib>

using namespace std;

int main(){

int cnt=0;

while (1) {

printf("Case %d:\n",++cnt);

system("make.exe");

system("std.exe");

system("baoli.exe");

if (system("fc baoli.out std.out")) break;

}

}

割顶

int dfs(int u, int fa) {

int lowu = pre[u] = ++dfs\_clock;

int child = 0;

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

int v = G[u][i];

if (!pre[v]) {

child++;int lowv = dfs(v, u);

lowu = min(lowu, lowv);

if (lowv >= lowu)

iscut[u] = true;

}

else if (pre[v] < pre[u] && v!=fa)

lowu = min(lowu, pre[v]);

}

if (fa < 0 && child == 1) iscut[u] = 0;

low[u] = lowu;

return lowu;

}

makedata

#include<cstdio>

#include<cstdlib>

#include<ctime>

#include<iostream>

using namespace std;

int main(){

freopen("make.in","w",stdout);

srand((unsigned)time(NULL));

int a=rand()%2010;int b=rand()%2010;

if(a<b)swap(a,b);

printf("%d %d",a,b);

}