Dijstra

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<queue>

using namespace std;

const int mod = (int) 1e9+7;

int n,m,dist[205],head[205],vis[205],now;

struct edge{

int to,next,val;

}e[2005];

void init(int x,int y,int v){

e[++now].to=y,e[now].val=v,e[now].next=head[x],head[x]=now;

e[++now].to=x,e[now].val=v,e[now].next=head[y],head[y]=now;

}

struct node{

int id,val;

node(int id,int val):id(id),val(val){} //构造方法

bool operator < (const node &x)const{ //重写运算符，val越小优先级越高

return val > x.val;

}

};

priority\_queue<node> q; //优先队列默认从大到小

void dij(int x){

dist[x]=0;

q.push(node(x,0));

while(!q.empty()){

int u=q.top().id; //得到优先队列里val最小的node结构体的id值

q.pop();

vis[u]=1; //如果当前结点已经被作为最近点去更新其他点的话，下次再访问到就直接跳过

for(int i=head[u];~i;i=e[i].next){ //i一开始是u的头结点，往后跳，当i==-1循环结束

int v=e[i].to; //v是取出下一个结点的编号

if(vis[v])continue;

if(dist[v]-e[i].val>dist[u]){ //如果距离被更新

dist[v]=dist[u]+e[i].val;

q.push(node(v,dist[v])); //把更新的结点放入优先队列

}

}

}

}

int main()

{

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

memset(dist,125,sizeof(dist));

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

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

now=0;

int x,y,z;

for(int i=0;i<m;i++){

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

init(x,y,z);

}

dij(n);

if(dist[1]>mod)printf("-1\n");

else printf("%d\n",dist[1]);

}

return 0;

}

Fload

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

const int mod = (int) 1e9+7;

int n,m,dis[205][205];

int main()

{

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

memset(dis,125,sizeof(dis)); //将数组初始化成一个很大的数

for(int i=0;i<n;i++)dis[i][i]=0; //自己到自己的距离为0

int x,y,z;

for(int i=0;i<m;i++){

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

dis[x][y]=dis[y][x]=min(dis[x][y],z); //双向边

}

for(int k=0;k<n;k++){ //选取一个点，把它尝试插入其他路径中

for(int i=0;i<n;i++){ //选取起点

for(int j=0;j<n;j++){ //选取一个终点

if(dis[i][j]-dis[k][j]>dis[i][k]){

dis[i][j]=dis[i][k]+dis[k][j];

}

}

}

}

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

if(dis[x][y]>mod)printf("-1\n");

else printf("%d\n",dis[x][y]);

}

return 0;

}

Spfa

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<queue>

using namespace std;

const int mod = (int) 1e9+7;

int n,m,dis[51][51],dist[51];

bool inq[51]; //inq[i]代表i是不是在队列中

void spfa(int x){

queue<int> q;

dist[x]=0;

q.push(x);

inq[x]=1;

while(!q.empty()){

int u=q.front();

q.pop();

inq[u]=0; //u点已经离开队列，就把inq[u]置为0

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

if(dist[i]-dis[u][i]>dist[u]){

dist[i]=dist[u]+dis[u][i]; //更新i点到起点的距离

if(inq[i])continue; //如果已经在队列里，跳过后续步骤

q.push(i); //把i点放入队列

inq[i]=1; //更新inq[i]

}

}

}

}

int main()

{

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

{

memset(dis,125,sizeof(dis));

memset(dist,125,sizeof(dist));

for(int i=0;i<=n;i++)dis[i][i]=0;

int x,y;

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

{

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

dis[x][y]=1;

}

spfa(1);

if(dist[n]>mod)printf("no\n");

else printf("yes\n");

}

return 0;

}

括号匹配

const int maxn = 105;

char str[maxn];

int dp[maxn][maxn];

bool ck(int i, int j) {

if ((str[i] == '(' && str[j] == ')') || (str[i] == '[' && str[j] == ']')) return true;

else return false;}

int main() {

while (~scanf("%s", str)) {

if (str[0] == 'e') break;

int len;

len = strlen(str);

memset(dp, 0, sizeof(dp));

for (int l = 1; l < len; l++) { //len = j - i 为当前区间长度

for (int i = 0, j = l; j < len; i++, j++) { // i++, j++

if (ck(i, j)) { // 匹配

dp[i][j] = dp[i + 1][j - 1] + 1;

}// 讨论区间合并情况，求最大值

for (int pos = i; pos < j; pos++) {

dp[i][j] = max(dp[i][j], dp[i][pos] + dp[pos + 1][j]);

}

}

}

printf("%d\n", dp[0][len - 1]);

}

return 0;

}

石子归并

const int maxn = 210;

int dp[maxn][maxn];

int sum[maxn];

int a[maxn];

int main() {

int n;

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

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

scanf("%d", &a[i]);

sum[i] = sum[i - 1] + a[i];

}

for (int len = 1; len < n; len++) { //操作区间的长度

for (int i = 1, j = len + 1; j <= n; i++, j++) { //始末

//检查是否匹配（非必须）

dp[i][j] = INF;

for (int s = i; s < j; s++) {

dp[i][j] = min(dp[i][j], dp[i][s] + dp[s + 1][j] + sum[j] - sum[i - 1]);

}

}

}

printf("%d\n", dp[1][n]);

}

return 0;

}

整数划分

long long a[20][20];

long long dp[25][25];

int main() {

int T,m;

scanf("%d",&T);

getchar();

while(T--){

char s[22];

scanf("%s",s+1);

scanf("%d",&m);

int l=strlen(s),ok=1;

memset(a,0,sizeof(a));

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

{

if(s[i]=='0')

ok=0;

for(int j=i;j<l;j++)

{

a[i][j]=a[i][j-1]\*10+(s[j]-'0');

}

}

if(ok==0&&l-1==m||l-1<m) {

printf("0\n");continue;

}

long long x,ans;

memset(dp,0,sizeof(dp));

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

dp[i][1]=a[1][i];

ans=0;

if(m==1) ans=dp[l-1][1];

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

for(int i=j;i<l;i++){

ans=a[i][i];

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

dp[i][j]=MAX(dp[i][j],dp[k][j-1]\*a[k+1][i]);

}

}

}

printf("%lld\n",dp[l-1][m]);

}

return 0;

}

Lis

int dp[500],a[500];

int main()

{

int n;

while(~scanf("%d",&n)&&n)

{

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

{

scanf("%d",&a[i]);

dp[i]=INF;

}

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

{

\*lower\_bound(dp,dp+n,a[i])=a[i];

}

cout<<lower\_bound(dp,dp+n,INF)-dp<<endl;

}

Lcs

int n,m,dp[maxn][maxn];

char a[maxn],b[maxn];

int main()

{

while(~scanf("%s%s",a,b))

{

n=strlen(a);

m=strlen(b);

memset(dp,0,sizeof(dp));

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

{

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

{

if(a[k-1]==b[j-1])

dp[k][j]=dp[k-1][j-1]+1;

else

dp[k][j]=max(dp[k-1][j],dp[k][j-1]);

}

}

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

}

return 0;

} }

Kmp

#include<iostream>

#define maxn 1000010

using namespace std;

int \_next[maxn];

char t[maxn],s[maxn];

int slen,tlen;

void getnext()

{

int k=-1,j=0;

\_next[0]=-1;

while(j<tlen)

{

if(k==-1||t[k]==t[j])\_next[++j]=++k;

else k=\_next[k];

}

}

int kmp()

{

int j=0,i=0;

getnext();

while(i<slen&&j<tlen)

{

if(j==-1||s[i]==t[j])

{

j++;

i++;

}

else j=\_next[j];

}

if(j==tlen) return i-tlen+1;//返回主串中第一个字符的位置

else return 0;

}

int main()

{

scanf("%s%s",s,t);

slen=strlen(s);

tlen=strlen(t);

int temp=kmp();

cout<<(temp==-1 ? 0:temp)<<endl;

}

树状数组

int sum[1000010];

int n;

struct node

{

int x,pos;

}a[1000010];

bool cmp(node a,node b)

{

return a.x<b.x;

}

void updata(int pos,int val)

{

while(pos<=n)

{

sum[pos]+=val;

pos+=(pos&-pos);//末尾的1进位

}

}

int query(int pos)

{

int ans=0;

while(pos>0)

{

ans+=sum[pos];

pos-=(pos&-pos);//去掉末尾的1

}

return ans;

}

int main()

{

int nxs=0;

scanf("%d",&n);

for(int i=1,x;i<=n;i++)//当数字是1-n时

{

scanf("%d",&x);

updata(x,1);

int temp=query(x);

nxs+=x-temp;

}

/\*

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

{

scanf("%d",&a[i].x);

a[i].pos=i;

}

sort(a+1,a+1+n,cmp);

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

{

updata(a[i].pos,1);

int temp=query(a[i].pos);

nxs+=i-temp;

}//当数字不是按照1-n时排序，弄一个小hash

\*/

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

return 0;

}

最小生成树

struct node{

int u,v,val;

bool operator<(const node& a)const{

return val<a.val;

}

}edge[200005];

int fa[200005];

int findfa(int t){

return fa[t]==t?fa[t]:fa[t]=findfa(fa[t]);

}

int main(){

int n,m,i,j,ans=0;

while(scanf("%d",&n)!=EOF){

if(n==0) break;

scanf("%d",&m);

if(m==0) cout<<0<<endl;

else

{ans=0;

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

for(i=1;i<=m;i++)cin>>edge[i].u>>edge[i].v>>edge[i].val;

sort(edge+1,edge+1+m);

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

int u=edge[i].u,v=edge[i].v;

if(findfa(u)!=findfa(v)){

ans+=edge[i].val;

fa[findfa(u)]=findfa(v);

}

}

cout<<ans<<endl;

}

}

return 0;

}

链式前向星添边

struct node{

int to,nex,val;

}edge[200050];

int head[200050],cnt;

void add(int u,int v)

{

edge[cnt].to=v;

edge[cnt].nex=head[u];

head[u]=cnt;

cnt++;

}

。。int i=head[t];~i;i=edge[i].nex

线段树

#include<iostream>

#include<cstdio>

#define lson l+r<<1

#define rson l+r<<1|1

#define ll long long

using namespace std;

ll a[100050\*4],f[100050\*4];

void build(ll l,ll r,ll root)

{

if(l==r)

{

scanf("%lld",&a[root]);

return;

}

ll mid=l+r>>1;

build(l,mid,lson);

build(mid+1,r,rson);

a[root]=a[lson]+a[rson];

}

void pushdown(ll l,ll r,ll root)

{

if(f[root]==0) return;

f[lson]+=f[root];

f[rson]+=f[root];

ll mid=l+r>>1;

a[lson]+=((mid-l+1)\*f[root]);

a[rson]+=((r-mid)\*f[root]);

f[root]=0;

}

void updata(ll l,ll r,ll root,ll x,ll y,ll z)

{

if(l==x&&r==y)

{

a[root]+=(r-l+1)\*z;

f[root]+=z;

return;

}

pushdown(l,r,root);

ll mid=l+r>>1;

if(y<=mid)updata(l,mid,lson,x,y,z);

else if(x>mid) updata(mid+1,r,rson,x,y,z);

else updata(l,mid,lson,x,mid,z),updata(mid+1,r,rson,mid+1,y,z);

a[root]=a[rson]+a[lson];

}

ll query(ll l,ll r,ll root,ll x,ll y)

{

if(l==x&&r==y)

return a[root];

pushdown(l,r,root);

ll mid=l+r>>1;

if(y<=mid)return query(l,mid,lson,x,y);

else if(x>mid)return query(mid+1,r,rson,x,y);

else return query(l,mid,lson,x,mid)+query(mid+1,r,rson,mid+1,y);

}

int main()

{

ll t,n;

cin>>t>>n;

build(1,t,1);

char ff[2];

while(n--)

{

scanf("%s",ff);

if(ff[0]=='Q')

{

ll x,y;

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

printf("%lld\n",query(1,t,1,x,y));

}

if(ff[0]=='C')

{

ll x,y,z;

scanf("%lld%lld%lld",&x,&y,&z);

updata(1,t,1,x,y,z);

}

}

return 0;

}

卡特兰数

C(n) = C(1)\*C(n-1) + C(2)\*C(n-2) + ... + C(n-1)C(1)，n>=2