**Emacs配置**

(global-hl-line-mode t)  
(ido-mode t)  
(set-background-color "gray20")  
(set-foreground-color "wheat")  
(tool-bar-mode -1)  
(scroll-bar-mode -1)

(blink-cursor-mode -1)  
(menu-bar-mode -1)  
(global-linum-mode t)

(global-font-lock-mode t);;高亮

;;;;;设置编译信息  
(defun compile-file ()  
  (interactive)  
  (compile (format "g++ -o %s %s -g -lm -Wall"  (file-name-sans-extension (buffer-name))(buffer-name))))  
(global-set-key (kbd "<f9>") 'compile-file)  
;;;;;设置一键调试  
(global-set-key (kbd "<f7>") 'gud-gdb)

;;;;;允许emacs和外部其他程序的粘贴  
(setq x-select-enable-clipboard t)

;;设置tab为2个空格的宽度  
(setq default-tab-width 2)  
(setq c-basic-offset 2)

;;;;;是用滚轴鼠标  
(mouse-wheel-mode t)

(show-paren-mode 1);;括号匹配

**对拍**

:1

make

shit

shit1

fc shit.out shit1.out ||pause

goto 1

**c++扩栈**

**就在开头加上一句**

**#pragma comment(linker,"/STACK:102400000,102400000")**

**G++扩栈**

**int size = 256 << 20; // 256MB**

**char \*p = (char\*)malloc(size) + size;**

**\_\_asm\_\_("movl %0, %%esp\n" :: "r"(p));**

**(asm两边各两个下划线)**

**make**

**#include<sys/timeb.h>**

**int main(){**

**freopen("1.out","w",stdout);**

**struct timeb t;ftime(&t);**

**srand(t.millitm+t.time\*1000);**

**printf("%d",rand());**

**return 0;**

**}**

**ios**

**ios::sync\_with\_stdio(false);**

**加上上面这句话就行了,不过不能用scanf和printf了**

**陈丹琪分治**

**const double eps= 1e-8;**

**const int maxn= 300001,oo= 1e8;**

**struct point{**

**double x,y;**

**};**

**typedef point vec;**

**int dcmp(double x){**

**if (fabs(x)<eps) return 0;**

**return x<0 ? -1:1;**

**}**

**point p[maxn],ch[maxn];**

**int c[maxn];**

**double f[maxn],a[maxn],b[maxn],r[maxn];**

**double s,ans;**

**int i,j,n;**

**vec operator +(vec a,vec b){a.x+= b.x;a.y+= b.y;return a;}**

**vec operator -(vec a,vec b){a.x-= b.x;a.y-= b.y;return a;}**

**vec operator \*(double t,vec a){a.x\*= t;a.y\*= t;return a;}**

**vec operator /(vec a,double t){a.x/= t;a.y/= t;return a;}**

**bool operator <(vec a,vec b){return dcmp(a.x-b.x)<0||(!dcmp(a.x-b.x)&&dcmp(a.y-b.y)<0);}**

**bool operator ==(vec a,vec b){return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y);}**

**double cj(vec a,vec b){return a.x\*b.y-a.y\*b.x;}**

**double dj(vec a,vec b){return a.x\*b.x+a.y\*b.y;}**

**double w(int i,point p){**

**return p.x\*a[i]+p.y\*b[i];**

**}**

**bool cmp(point a,point b){return a<b;}**

**bool cmp1(int i,int j){**

**return -a[i]/b[i]>-a[j]/b[j];**

**}**

**int convexHull(int n){**

**sort(p+1,p+n+1,cmp);**

**int i,w= 0,q= 1;**

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

**if (dcmp(p[i].y-p[q].y)>=0) q= i;**

**for (i= q;i<=n;i++){**

**while (w>1&&dcmp(cj(ch[w]-ch[w-1],p[i]-ch[w-1]))>=0) w--;**

**ch[++w]= p[i];**

**}return w;**

**}**

**void cdq(int L,int R){**

**if (L==R) return;**

**double x,y,Max= -oo;**

**int mid= (L+R)>>1,i,n,j,u;**

**cdq(L,mid);**

**sort(c+mid+1,c+R+1,cmp1);**

**for (i= L;i<=mid;i++){**

**x= f[i]/(r[i]\*a[i]+b[i]);**

**p[i-L+1]= (point){x\*r[i],x};**

**Max= max(f[i],Max);**

**}j= 1;**

**n= convexHull(mid-L+1);**

**for (i= mid+1;i<=R;i++){**

**u= c[i];y= w(u,ch[j]);**

**if (j<n){**

**while (j<n){**

**x= w(u,ch[j+1]);**

**if (x<y) break;**

**y= x;j++;**

**}**

**}**

**f[u]= max(f[u],Max);**

**f[u]= max(f[u],y);**

**}**

**sort(c+mid+1,c+R+1);**

**cdq(mid+1,R);**

**}**

**int main(){**

**scanf("%d %lf",&n, &s);**

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

**scanf("%lf %lf %lf",&a[i], &b[i], &r[i]);**

**c[i]= i;**

**}**

**f[1]= s;**

**cdq(1,n);**

**for (i= 1;i<=n;i++) ans= max(ans,f[i]);**

**printf("%.3lf",ans);**

**return 0;**

**}**

**动态规划**

**1:**

**const int maxn= 3001;**

**ll b[200][200],d[2][2000];**

**int n,m,cur;**

**void update(int a,int b){**

**if (b & (1<<m))**

**d[cur][b^(1<<m)]+= d[cur^1][a];**

**}**

**ll solve(int n,int m){**

**int i,j,k;cur= 0;**

**memset(d,0,sizeof(d));**

**d[0][(1<<m)-1]= 1;**

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

**for (j= 1;j<=m;j++){**

**cur^= 1;memset(d[cur],0,sizeof(d[cur]));**

**for (k= 0;k<(1<<m);k++){**

**update(k,k<<1);**

**if (i>1&&!(k&(1<<(m-1)))) update(k,(k<<1)^(1<<m)^1);**

**if (j>1&&!(k&1)) update(k,(k<<1)^3);**

**}**

**}return d[cur][(1<<m)-1];**

**}**

**int main(){**

**memset(b,-1,sizeof(b));**

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

**if (n<m) swap(n,m);**

**if (b[n][m]<0) b[n][m]= solve(n,m);**

**printf("%lld\n",b[n][m]);**

**}return 0;**

**}**

**2:**

**const int oo= 1e9;**

**int n,m,i,j,ans;**

**struct state{**

**int up[10];**

**int left;**

**int encode(){**

**int key= left,i;**

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

**key= key\*3+up[i];**

**return key;**

**}**

**bool next(int x,int y,int U,int R,int D,int L,state &T){**

**if (y==m&&R!=0) return 0;**

**if (x==n&&D!=0) return 0;**

**int must\_left= (y>1&&left!=0);**

**int must\_up= (x>1&&up[y]!=0);**

**if ((must\_left&&L!=left)||(!must\_left&&L!=0)) return 0;**

**if ((must\_up&&U!=up[y])||(!must\_up&&U!=0)) return 0;**

**if (must\_up&&must\_left&&up[y]!=left) return 0;**

**int i;**

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

**T.up[i]= up[i];**

**T.left= R;**

**T.up[y]= D;**

**return 1;**

**}**

**};**

**state init;**

**int f[10][10][60000],g[20][20];**

**int dfs(int x,int y,state S){**

**if (y>m){x++;y= 1;}**

**if (x>n) return 0;**

**int key= S.encode(),&res= f[x][y][key],i;**

**if (res!=-1) return res;**

**res= oo;**

**state T;**

**if (g[x][y]<=1){**

**if (S.next(x,y,0,0,0,0,T)) res= min(res,dfs(x,y+1,T));**

**if (g[x][y]==0){**

**for (i= 1;i<=2;i++){**

**if (S.next(x,y,i,i,0,0,T)) res= min(res,dfs(x,y+1,T)+2);**

**if (S.next(x,y,i,0,i,0,T)) res= min(res,dfs(x,y+1,T)+2);**

**if (S.next(x,y,i,0,0,i,T)) res= min(res,dfs(x,y+1,T)+2);**

**if (S.next(x,y,0,i,i,0,T)) res= min(res,dfs(x,y+1,T)+2);**

**if (S.next(x,y,0,i,0,i,T)) res= min(res,dfs(x,y+1,T)+2);**

**if (S.next(x,y,0,0,i,i,T)) res= min(res,dfs(x,y+1,T)+2);**

**}**

**}**

**}else {**

**i= g[x][y]-1;**

**if (S.next(x,y,i,0,0,0,T)) res= min(res,dfs(x,y+1,T)+1);**

**if (S.next(x,y,0,i,0,0,T)) res= min(res,dfs(x,y+1,T)+1);**

**if (S.next(x,y,0,0,i,0,T)) res= min(res,dfs(x,y+1,T)+1);**

**if (S.next(x,y,0,0,0,i,T)) res= min(res,dfs(x,y+1,T)+1);**

**}return res;**

**}**

**int main(){**

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

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

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

**scanf("%d",&g[i][j]);**

**memset(f,-1,sizeof(f));**

**ans= dfs(1,1,init);**

**if (ans!=oo) printf("%d\n",ans/2);**

**else printf("%d\n",0);**

**}return 0;**

**}**

**2判定性问题**

**const int maxn= 3001,base= 1000,oo= 1e8;**

**vector<int> g[maxn];**

**int b[maxn],X[maxn],stack[maxn];**

**int n,m,i,j,x,x1,y1,x2,y2,la;**

**void addedge(int x,int y){ g[x].push\_back(y); }**

**void link(int x1,int y1,int x2,int y2){**

**if (y1){**

**if (y2){**

**addedge(x1+base,x2);**

**addedge(x2+base,x1);**

**}else {**

**addedge(x1+base,x2+base);**

**addedge(x2,x1);**

**}**

**}else {**

**if (y2){**

**addedge(x1,x2);**

**addedge(x2+base,x1+base);**

**}else {**

**addedge(x1,x2+base);**

**addedge(x2,x1+base);**

**}**

**}**

**}**

**bool dfs(int o){**

**if (b[X[o]]) return 0;**

**if (b[o]) return 1;**

**b[o]= 1;stack[++la]= o;**

**int i,len= g[o].size();**

**for (i= 0;i<len;i++){**

**int v= g[o][i];**

**if (!dfs(v))**

**return 0;**

**}**

**return 1;**

**}**

**bool solve(int n){**

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

**if (!b[i]&&!b[X[i]]){**

**la= 0;**

**if (!dfs(i)){**

**while (la) b[stack[la--]]= 0;**

**if (!dfs(X[i])) return 0;**

**}**

**}**

**return 1;**

**}**

**void init(int n){**

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

**g[i].clear(),g[i+base].clear(),X[i]= i+base,X[i+base]= i;**

**memset(b,0,sizeof(b));**

**}**

**int main(){**

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

**init(n);**

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

**scanf("%d %d %d %d",&x1, &y1, &x2, &y2);**

**link(x1,y1,x2,y2);**

**}**

**if (solve(n)) printf("yes!");**

**else printf("no!");**

**return 0;**

**}**

**二分图最佳完美匹配**

**const int maxn= 3001,oo= 1e8;**

**struct edge{**

**int v,w;**

**};**

**vector<edge> g[maxn];**

**int lx[maxn],ly[maxn],S[maxn],T[maxn],d[maxn];**

**int n,m,x,y,z,i,j,ans;**

**bool match(int o){**

**S[o]= 1;**

**int len= g[o].size(),i;**

**for (i= 0;i<len;i++){**

**int v= g[o][i].v,w= g[o][i].w;**

**if (lx[o]+ly[v]==w && !T[v]){**

**T[v]= 1;**

**if (!d[v] || match(d[v])){**

**d[v]= o;**

**return 1;**

**}**

**}**

**}return 0;**

**}**

**void update(){**

**int i,j,a= oo;**

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

**if (S[i]){**

**int len= g[i].size();**

**for (j= 0;j<len;j++){**

**int v= g[i][j].v;**

**if (!T[v])**

**a= min(a,lx[i]+ly[v]-g[i][j].w);**

**}**

**}**

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

**if (S[i]) lx[i]-= a;**

**if (T[i]) ly[i]+= a;**

**}**

**}**

**int main(){**

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

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

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

**g[x].push\_back((edge){y,z});**

**}**

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

**d[i]= ly[i]= 0;**

**lx[i]= -oo;int len= g[i].size();**

**for (j= 0;j<len;j++)**

**lx[i]= max(lx[i],g[i][j].w);**

**}**

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

**for (;;){**

**for (j= 1;j<=n;j++) S[j]= T[j]= 0;**

**if (match(i)) break;**

**else update();**

**}**

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

**ans+= lx[i]+ly[i];**

**printf("%d",ans);**

**return 0;**

**}**

**tarjan**

**const int maxn= 3001;**

**int head[maxn],next[maxn],des[maxn];**

**int pre[maxn],low[maxn],stack[maxn],sccno[maxn];**

**vector<int> scc[maxn];**

**int n,m,x,y,i,j,top,cnt,la,time;**

**void dfs(int o){**

**pre[o]= low[o]= ++time;**

**stack[++la]= o;**

**int p= head[o];**

**while (p){**

**int v= des[p];**

**if (!pre[v]){**

**dfs(v);**

**low[o]= min(low[o],low[v]);**

**}else if (!sccno[v])//反向边更新**

**low[o]= min(low[o],pre[v]);**

**p= next[p];**

**}**

**if (low[o]==pre[o]){**

**cnt++;**

**while (true){**

**int x= stack[la--];**

**scc[cnt].push\_back(x);**

**sccno[x]= cnt;**

**if (x==o) break;**

**}**

**}**

**}**

**int main(){**

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

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

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

**top++;**

**des[top]= y;**

**next[top]= head[x];**

**head[x]= top;**

**}**

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

**if (!pre[i]) dfs(i);**

**printf("%d\n",cnt);**

**for (i= 1;i<=cnt;i++){**

**int len= scc[i].size();**

**for (j= 0;j<len;j++)**

**printf("%d ",scc[i][j]);**

**printf("\n");**

**}**

**return 0;**

**}**

**最大流**

**const int maxn= 3001,oo= 99999999;**

**struct edge{**

**int from,to,cap,flow;**

**};**

**vector<edge> edges;**

**vector<int> g[maxn];**

**int cur[maxn],d[maxn],l[maxn];**

**bool vis[maxn];**

**int n,m,s,t,i,j,x,y,z;**

**long long flow;**

**void addedge(int from,int to,int cap){**

**edges.push\_back((edge){from,to,cap,0});**

**edges.push\_back((edge){to,from,0,0});**

**int m= edges.size();**

**g[from].push\_back(m-2);**

**g[to].push\_back(m-1);**

**}**

**bool bfs(){**

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

**int f= 0,r= 1;**

**l[r]= s;vis[s]= 1;**

**while (f<r){**

**int i,u= l[++f],len =g[u].size();**

**for (i= 0;i<len;i++){**

**edge &e= edges[g[u][i]];**

**if (!vis[e.to] && (e.cap-e.flow>0)){**

**vis[e.to]= 1;**

**d[e.to]= d[u]+1;**

**l[++r]= e.to;**

**}**

**}**

**}**

**return vis[t];**

**}**

**long long dfs(int o,int re){**

**if (o==t || re==0) return re;**

**int f,flow= 0,len= g[o].size();**

**for (int &i= cur[o];i<len;i++){**

**edge &e= edges[g[o][i]];**

**if (d[o]+1==d[e.to] && (f= dfs(e.to,min(re,e.cap-e.flow)))>0){**

**flow+= f;**

**re-= f;**

**e.flow+= f;**

**edges[g[o][i]^1].flow-= f;**

**if (!re) break;**

**}**

**}**

**return flow;**

**}**

**int main(){**

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

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

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

**addedge(x,y,z);**

**}**

**while (bfs()){**

**memset(cur,0,sizeof(cur));**

**flow+= dfs(s,oo);**

**}**

**printf("%lld",flow);**

**return 0;**

**}**

**最小费用最大流**

**const int maxn= 3001,oo= 99999;**

**struct edge{**

**int from,to,cap,flow,cost;**

**};**

**vector<edge> edges;**

**vector<int> g[maxn];**

**int l[10\*maxn],d[maxn],a[maxn],p[maxn];**

**bool vis[maxn];**

**int i,a1,a2,a3,a4,n,m,s,t,flow,cost;**

**void addedge(int from,int to,int cap,int cost){**

**edges.push\_back((edge){from,to,cap,0,cost});**

**edges.push\_back((edge){to,from,0,0,-cost});**

**int m= edges.size();**

**g[from].push\_back(m-2);**

**g[to].push\_back(m-1);**

**}**

**bool spfa(){**

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

**for (int i= 1;i<=n;i++) d[i]= oo;**

**d[s]= 0;vis[s]= 1;a[s]= oo;**

**int f= 0,r= 1;**

**l[1]= s;**

**while (f<r){**

**int u= l[++f],len= g[u].size();**

**vis[u]= 0;**

**for (int i= 0;i<len;i++){**

**edge& e= edges[g[u][i]];**

**if (d[u]+e.cost<d[e.to] && e.cap>e.flow){**

**d[e.to]= d[u]+e.cost;**

**a[e.to]= min(a[u],e.cap-e.flow);**

**p[e.to]= g[u][i];**

**if (!vis[e.to]){**

**vis[e.to]= 1;**

**l[++r]= e.to;**

**}**

**}**

**}**

**}**

**if (d[t]==oo) return 0;**

**flow+= a[t];**

**cost+= d[t]\*a[t];**

**int u= t;**

**while (u!=s){**

**edges[p[u]].flow+= a[t];**

**edges[p[u]^1].flow-= a[t];**

**u= edges[p[u]].from;**

**}**

**return 1;**

**}**

**int main(){**

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

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

**scanf("%d %d %d %d",&a1, &a2, &a3, &a4);**

**addedge(a1,a2,a3,a4);**

**}**

**while (spfa());**

**printf("%d %d",flow, cost);**

**}**

**差分约束系统**

如果一个系统由n个变量和m个约束条件组成，其中每个约束条件形如xj-xi<=bk(i,j∈[1,n],k∈[1,m]),则称其为差分约束系统(system of difference constraints)。亦即，差分约束系统是求解关于一组变量的特殊不等式组的方法。

求解差分约束系统，可以转化成图论的单源最短路径（或最长路径）问题。

观察xj-xi<=bk，会发现它类似最短路中的三角不等式d[v]<=d[u]+w[u,v]，即d[v]-d[u]<=w[u,v]。因此，以每个变量xi为结点，对于约束条件xj-xi<=bk，连接一条边(i,j)，边权为bk。我们再增加一个源点s,s与所有定点相连，边权均为0。对这个图，以s为源点运行Bellman-ford算法（或SPFA算法），最终{d[i]}即为一组可行解。

求最优值:

最大值求最长路，最小值求最短路

因为d[v]<=d[u]+w[u,v]的话,使得每个d[v]都尽量小了

d[v]>=d[u]+w[u,v]的话,使得每个d[v]都尽量大了

所以sigmad[v]分别最小最大...

求最短路时如果路径上有负权值，则要判负环

求最长路时如果路径上有正权环，则要判正环

如果是x-y<k的情况要变成x-y<=k-1的标准形式

不等号的方向要统一

d[]值看情况初始化

**ac 自动机**

**const int maxn= 3001,sig= 30;**

**struct node{**

**int ch[sig+1];**

**int v;**

**};**

**node a[maxn];**

**char s[maxn][120],st[maxn];**

**int F[maxn],last[maxn],d[maxn];**

**int n,i,j,len,now,top,L;**

**int idx(char c){return c-'a';}**

**void getFail(){**

**queue<int> Q;**

**int i;F[0]= 0;**

**for (i= 0;i<sig;i++){**

**int v= a[0].ch[i];**

**if (v){**

**Q.push(v);**

**F[v]= last[v]= 0;**

**}**

**}**

**while (!Q.empty()){**

**int u= Q.front();**

**Q.pop();**

**for (i= 0;i<sig;i++){**

**int v= a[u].ch[i];**

**if (v){**

**Q.push(v);**

**int j= F[u];**

**while (j && !a[j].ch[i]) j= F[j];**

**j= a[j].ch[i];**

**F[v]= j;**

**last[v]= a[j].v? j:last[j];**

**}**

**}**

**}**

**}**

**void print(int j){**

**while (j){**

**d[a[j].v]++;**

**j= last[j];**

**}**

**}**

**void find(char \*t){**

**int len= strlen(t+1),i,j= 0;**

**for (i= 1;i<=len;i++){**

**int c= idx(t[i]);**

**while (j && !a[j].ch[c]) j= F[j];**

**j= a[j].ch[c];**

**if (a[j].v) print(j);**

**else print(last[j]);**

**}**

**}**

**int main(){**

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

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

**scanf("%s",s[i]+1);**

**len= strlen(s[i]+1);**

**now= 0;**

**for (j= 1;j<=len;j++){**

**int c= idx(s[i][j]);**

**if (a[now].ch[c]) now= a[now].ch[c];**

**else {**

**a[now].ch[c]= ++top;**

**now= top;**

**}**

**}**

**a[now].v= i;**

**}**

**getFail();**

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

**find(st);**

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

**if (d[i])**

**printf("%s\n",s[i]+1);**

**return 0;**

**}**

**Kmp算法**

**const int maxn= 3001;**

**int f[maxn];**

**char p[maxn],t[maxn];**

**int i,j,lent,lenp;**

**void kmp(){**

**f[1]= 0;**

**int j= 0,i;**

**for (i= 2;i<=lenp;i++){**

**while (j>0 && p[j+1]!=p[i])**

**j= f[j];**

**if (p[j+1]==p[i]) j++;**

**f[i]= j;**

**}**

**}**

**int main(){**

**bool bl= 0;**

**scanf("%s%s",t+1, p+1);**

**lent= strlen(t+1);**

**lenp= strlen(p+1);**

**kmp();**

**for (i= 1;i<=lent;i++){**

**while (j>0 && p[j+1]!=t[i])**

**j= f[j];**

**if (p[j+1]==t[i]) j++;**

**if (j==lenp){**

**printf("%d ",i-j+1);**

**bl =1;**

**j= f[j];**

**}**

**}**

**if (!bl) printf("Oh shit !");**

**return 0;**

**}**

**manacher**

**const int maxn= 3001;**

**int rad[maxn];**

**char s[maxn],st[maxn];**

**int id,mx,len,n,i,ans;**

**int main(){**

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

**len= strlen(st+1);**

**s[0]='$';**

**for (i= 1;i<=len;i++){**

**s[++n]= '#';**

**s[++n]= st[i];**

**}**

**s[++n]= '#';**

**len= n;**

**for (i= 1;i<=len;i++){**

**if (mx>i) rad[i]= min(rad[2\*id-i],mx-i+1);**

**else rad[i]= 1;**

**while (s[i-rad[i]]==s[i+rad[i]]) rad[i]++;**

**if (i+rad[i]-1>mx){**

**mx= i+rad[i]-1;**

**id= i;**

**}**

**}**

**for (i= 1;i<=len;i++)**

**ans= max(ans,rad[i]-1);**

**printf("%d",ans);**

**return 0;**

**}**

**后缀数组**

**const int maxn= 3001;**

**char s[maxn],st[maxn];**

**int c[maxn],X[maxn],sa[maxn],Y[maxn];**

**int n,m,len,i;**

**void build(int m){**

**int i;**

**for (i= 0;i<=m;i++) c[i]= 0;**

**for (i= 1;i<=len;i++) c[X[i]]++;**

**for (i= 1;i<=m;i++) c[i]+= c[i-1];**

**for (i= len;i>0;i--) sa[c[X[i]]--]= i;**

**for (int k= 1;k<=len;k<<= 1){**

**int p= 0;**

**for (i= len-k+1;i<=len;i++) Y[++p]= i;**

**for (i= 1;i<=len;i++)**

**if (sa[i]>k)**

**Y[++p]= sa[i]-k;**

**for (i= 0;i<=m;i++) c[i]= 0;**

**for (i= 1;i<=len;i++) c[X[Y[i]]]++;**

**for (i= 1;i<=m;i++) c[i]+= c[i-1];**

**for (i= len;i>0;i--) sa[c[X[Y[i]]]--]= Y[i];**

**swap(X,Y);**

**p= 1;X[sa[1]]= 1;**

**for (i= 2;i<=len;i++)**

**if (Y[sa[i-1]]==Y[sa[i]] && Y[sa[i-1]+k]==Y[sa[i]+k])**

**X[sa[i]]= p;**

**else X[sa[i]]= ++p;**

**if (p==len) break;**

**m= p;**

**}**

**}**

**int com(char \*p,int mid){**

**return strncmp(p+1,st+sa[mid],strlen(p+1));**

**}**

**int find(char\* s){**

**if (com(s,1)<0) return 0;**

**if (com(s,len)>0) return 0;**

**int l= 1,r= len;**

**while (l<=r){**

**int mid= (l+r)>>1;**

**int rt= com(s,mid);**

**if (!rt) return 1;**

**if (rt<0) r= mid-1;**

**if (rt>0) l= mid+1;**

**}**

**return 0;**

**}**

**int main(){**

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

**len= strlen(st+1);**

**for (i= 1;i<=len;i++)**

**X[i]= st[i]-'a'+1;**

**build(28);**

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

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

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

**int ans= find(s);**

**if (ans==1) printf("yes\n");**

**else printf("no\n");**

**}**

**return 0;**

**}**

**后缀自动机**

**const int maxn= 3001,sig= 30;**

**struct node{**

**node \*par,\*ch[sig];**

**int v;**

**node(int x):**

**par(0),v(x){memset(ch,0,sizeof(ch));}**

**};**

**node \*root,\*last;**

**char s[maxn];**

**int len;**

**void extend(int x){**

**node \*p= last,\*np= new node(p->v+1);**

**while (p && !p->ch[x]){**

**p->ch[x]= np;p= p->par;**

**}**

**if (!p)**

**np->par= root;**

**else {**

**node \*q= p->ch[x];**

**if (q->v==p->v+1)**

**np->par= q;**

**else {**

**node \*nq= new node(p->v+1);**

**memcpy(nq->ch,q->ch,sizeof(q->ch));**

**nq->par= q->par;**

**q->par= nq;**

**np->par= nq;**

**while (p && p->ch[x]==q){**

**p->ch[x]= nq;p= p->par;**

**}**

**}**

**}**

**last= np;**

**}**

**int idx(char c){return c-'a';}**

**void build(char \*s){**

**root= last= new node(0);**

**for (int i= 1;i<=len;i++)**

**extend(idx(s[i]));**

**}**

**int main(){**

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

**len= strlen(s+1);**

**build(s);**

**return 0;**

**}**

**树**

**最小树形图**

**O(ve)**

**const int maxn= 300,oo= 1e5;**

**int w[maxn][maxn],vis[maxn];**

**int pre[maxn],iw[maxn],cid[maxn],removed[maxn];**

**int ans,s,i,j,x,y,z,n,m,max\_cid;**

**int dfs(int o){**

**vis[o]= 1;int re= 1,i;**

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

**if (!vis[i]&&w[o][i]!=oo)**

**re+= dfs(i);**

**return re;**

**}**

**void update(int u){**

**iw[u]= oo;**

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

**if (!removed[i]&&iw[u]>w[i][u])**

**iw[u]= w[i][u],pre[u]= i;**

**}**

**bool cycle(int u){**

**max\_cid++;**

**int v= u;**

**while (cid[v]!=max\_cid){cid[v]= max\_cid;v= pre[v];};**

**return u==v;**

**}**

**bool get(int s){**

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

**if (dfs(s)!=n) return 0;**

**memset(cid,0,sizeof(cid));**

**memset(removed,0,sizeof(removed));**

**int i,u;**

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

**update(i);**

**iw[s]= 0;pre[s]= s;**

**ans= max\_cid= 0;**

**for (;;){**

**int bl= 0;**

**for (u= 1;u<=n;u++)**

**if (u!=s&&!removed[u]&&cycle(u)){**

**int v= u;bl= 1;**

**do {**

**if (u!=v) removed[v]= 1;**

**ans+= iw[v];**

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

**if (!removed[i]&&cid[i]!=cid[u]){**

**if (w[i][v]!=oo)**

**w[i][u]= min(w[i][u],w[i][v]-iw[v]);**

**w[u][i]= min(w[u][i],w[v][i]);**

**if (pre[i]==v) pre[i]= u;**

**}**

**v= pre[v];**

**}while (u!=v);**

**update(u);**

**break;**

**}**

**if (!bl) break;**

**}**

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

**if (!removed[i])**

**ans+= iw[i];**

**return 1;**

**}**

**int main()**

**{**

**freopen("1.in","r",stdin);**

**freopen("1.out","w",stdout);**

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

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

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

**w[i][j]= oo;**

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

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

**w[x][y]= z;**

**}**

**if (dfs(1)!=n) printf("no solution!");**

**else {**

**get(1);**

**printf("%d",ans);**

**}**

**return 0;**

**}**

**splay 树**

**const int maxn= 6e5,oo= 1e9+76543221;**

**struct node \*null;**

**struct node{**

**node \*ch[2];**

**int v,s,flip,sum,z,l,m,r;**

**int cmp(int k){**

**int d= ch[0]->s;**

**if (d+1==k) return -1;**

**return k>d+1;**

**}**

**void mt(){**

**s= ch[0]->s+ch[1]->s+1;**

**int i,S[2]= {0},L[2],M[2],R[2];**

**for (i= 0;i<=1;i++)**

**L[i]= M[i]= R[i]= -oo;**

**for (i= 0;i<=1;i++){**

**if (ch[i]->z==oo){**

**L[i]= ch[i]->l;**

**M[i]= ch[i]->m;**

**R[i]= ch[i]->r;**

**S[i]+= ch[i]->sum;**

**}else {**

**int x= max(ch[i]->z,ch[i]->z\*ch[i]->s);**

**L[i]= M[i]= R[i]= x;**

**S[i]+= ch[i]->z\*ch[i]->s;**

**}**

**if (ch[i]->flip) swap(L[i],R[i]);**

**}**

**l= max(L[0],S[0]+v+max(0,L[1]));**

**m= max(M[0],M[1]);**

**m= max(m,max(0,R[0])+v+max(0,L[1]));**

**r= max(R[1],max(0,R[0])+v+S[1]);**

**sum= S[0]+S[1]+v;**

**}**

**void pushdown(){**

**if (z!=oo){**

**v= z;**

**sum= s\*z;**

**l= m= r= max(z,sum);**

**if (ch[0]!=null) ch[0]->z= z;**

**if (ch[1]!=null) ch[1]->z= z;**

**z= oo;**

**}**

**if (flip){**

**flip= 0;**

**swap(ch[0],ch[1]);**

**if (ch[0]!=null) ch[0]->flip^= 1;**

**if (ch[1]!=null) ch[1]->flip^= 1;**

**}**

**}**

**void clear(){**

**v= s= flip= sum= 0;**

**z= oo;**

**l= m= r= -oo;**

**ch[0]= ch[1]= null;**

**}**

**node(int x):v(x),s(0),flip(0),sum(0),z(oo),l(-oo),m(-oo),r(-oo){**

**ch[0]= ch[1]= null;**

**}**

**};**

**node \*stack[maxn];**

**char s[20];**

**int c[maxn];**

**node \*root,\*left,\*mid,\*right;**

**int n,m,i,j,x,y,z,la,top;**

**void rotate(node \*&o,int d){**

**node \*k= o->ch[d^1];o->ch[d^1]= k->ch[d];k->ch[d]= o;**

**o->mt();k->mt();o= k;**

**}**

**void splay(node \*&o,int k){**

**o->pushdown();**

**int d= o->cmp(k);**

**if (d!=-1){**

**if (d) k-= o->ch[0]->s+1;**

**node \*p= o->ch[d];**

**p->pushdown();**

**int d2= p->cmp(k);**

**if (d2!=-1){**

**if (d2) k-= p->ch[0]->s+1;**

**splay(p->ch[d2],k);**

**if (d==d2) rotate(o,d^1);**

**else rotate(o->ch[d],d);**

**}rotate(o,d^1);**

**}**

**}**

**node \*merge(node \*left,node \*right){**

**splay(left,left->s);**

**left->ch[1]= right;**

**left->mt();**

**return left;**

**}**

**void split(node \*o,int k,node \*&left,node \*&right){**

**splay(o,k);**

**left= o;**

**right= o->ch[1];**

**o->ch[1]= null;**

**o->mt();**

**}**

**node \*build(int sz){**

**if (!sz) return null;**

**node \*l= build(sz/2);**

**node \*o;**

**if (la){**

**o= stack[la--];**

**o->clear();**

**o->v= c[top++];**

**}else o= new node(c[top++]);**

**o->ch[0]= l;**

**o->ch[1]= build(sz-sz/2-1);**

**o->mt();**

**return o;**

**}**

**void recover(node \*o){**

**if (o==null) return;**

**recover(o->ch[0]);**

**stack[++la]= o;**

**recover(o->ch[1]);**

**}**

**int main(){**

**null= new node(0);**

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

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

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

**root= build(n+1);**

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

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

**if (s[1]=='I'){//insert**

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

**if (y==0) continue;**

**top= 1;**

**for (j= 1;j<=y;j++)**

**scanf("%d",&c[j]);**

**mid= build(y);**

**split(root,x+1,left,right);**

**root= merge(merge(left,mid),right);**

**}else if (s[1]=='D'){//delete**

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

**if (y==0) continue;**

**split(root,x,left,mid);**

**split(mid,y,mid,right);**

**recover(mid);**

**root= merge(left,right);**

**}else if (s[1]=='R'){//reverse**

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

**if (y==0) continue;**

**split(root,x,left,mid);**

**split(mid,y,mid,right);**

**mid->flip^= 1;**

**root= merge(merge(left,mid),right);**

**}else if (s[1]=='G'){//get-sum**

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

**if (y==0){**

**printf("0\n");**

**continue;**

**}**

**split(root,x,left,mid);**

**split(mid,y,mid,right);**

**printf("%d\n",mid->sum);**

**root= merge(merge(left,mid),right);**

**}else if (s[3]=='K'){//make-same**

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

**if (y==0) continue;**

**split(root,x,left,mid);**

**split(mid,y,mid,right);**

**mid->z= z;**

**root= merge(merge(left,mid),right);**

**}else{//max-sum**

**split(root,1,left,right);**

**printf("%d\n",right->m);**

**root= merge(left,right);**

**}**

**}return 0;**

**}**

**主席树**

**const int maxn= 3001,L= -1000,R= 1001;**

**int a[maxn],left[maxn],right[maxn],size[maxn];**

**int n,m,i,x,y,z,tot,ans;**

**int change(int o,int l,int r,int v,int w){**

**int u= ++tot;**

**size[u]= size[o]+w;**

**if (l+1==r){**

**left[u]= right[u]= 0;**

**return u;**

**}**

**int mid= (l+r)>>1;**

**if (v<mid){**

**right[u]= right[o];**

**left[u]= change(left[o],l,mid,v,w);**

**}else {**

**left[u]= left[o];**

**right[u]= change(right[o],mid,r,v,w);**

**}return u;**

**}**

**int ask(int x,int y,int k){**

**if (y-x+1<k) return 0;**

**int l= L,r= R;**

**x= a[x-1];y= a[y];**

**while (l+1<r){**

**int mid= (l+r)>>1;**

**if (size[left[y]]-size[left[x]]>=k){**

**x= left[x];**

**y= left[y];**

**r= mid;**

**}else {**

**k-= size[left[y]]-size[left[x]];**

**x= right[x];**

**y= right[y];**

**l= mid;**

**}**

**}return l;**

**}**

**int main(){**

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

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

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

**a[i]= change(a[i-1],L,R,x,1);**

**}**

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

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

**ans= ask(x,y,z);**

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

**}return 0;**

**}**

**Kd树**

**const int maxn= 3e5;**

**struct point{**

**int d[3],w;**

**};**

**struct node{**

**int d[3],w,M;**

**int split,L[3],R[3],ch[2];**

**};**

**point P[maxn];**

**node a[maxn];**

**int last[maxn],next[maxn],A[maxn],B[maxn],range[3][2];**

**int n,m,i,x,y,z,root,tot,D,ans;**

**bool operator <(const point &a,const point &b){**

**return a.d[D]<b.d[D];**

**}**

**void update(int o){**

**int ch[2]= {a[o].ch[0],a[o].ch[1]};**

**for (int i= 0;i<2;i++)**

**if (ch[i])**

**for (int j= 0;j<3;j++){**

**a[o].L[j]= min(a[o].L[j],a[ch[i]].L[j]);**

**a[o].R[j]= max(a[o].R[j],a[ch[i]].R[j]);**

**a[o].M= max(a[o].M,a[ch[i]].M);**

**}**

**}**

**double F(int l,int r){**

**double x= 0,sum= 0;**

**int i;**

**for (i= l;i<=r;i++)**

**x+= P[i].d[D];**

**x/= r-l+1;**

**for (i= l;i<=r;i++)**

**sum+= (P[i].d[D]-x)\*(P[i].d[D]-x);**

**return sum/= r-l+1;**

**}**

**void build(int &o,int l,int r){**

**if (!o) o= ++tot;**

**int i,mid= (l+r)>>1;**

**double f[3];i= 0;**

**for (D= 0;D<3;D++){**

**f[D]= F(l,r);**

**i= f[D]>f[i]?D:i;**

**}**

**D= i;**

**sort(P+l,P+r+1);**

**for (i= 0;i<3;i++)**

**a[o].d[i]= a[o].L[i]= a[o].R[i]= P[mid].d[i];**

**a[o].split= D;a[o].w= a[o].M= P[mid].w;**

**if (l<mid) build(a[o].ch[0],l,mid-1);**

**if (mid<r) build(a[o].ch[1],mid+1,r);**

**update(o);**

**}**

**bool ok(int o){**

**if (!o) return 0;**

**for (int i= 0;i<3;i++)**

**if (range[i][1]<a[o].L[i]||range[i][0]>a[o].R[i]) return 0;**

**return 1;**

**}**

**void query(int o){**

**if (a[o].M<=ans) return;**

**bool bl= 1;int i;**

**for (i= 0;i<3;i++)**

**if (a[o].L[i]<range[i][0]||a[o].R[i]>range[i][1]){**

**bl= 0;**

**break;**

**}**

**if (bl){**

**ans= max(ans,a[o].M);**

**return;**

**}**

**bl= 1;**

**for (i= 0;i<3;i++)**

**if (a[o].d[i]<range[i][0]||a[o].d[i]>range[i][1]){**

**bl= 0;**

**break;**

**}**

**if (bl) ans= max(ans,a[o].w);**

**for (i= 0;i<2;i++)**

**if (ok(a[o].ch[i]))**

**query(a[o].ch[i]);**

**}**

**int main(){**

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

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

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

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

**last[i]= B[A[i]];**

**B[A[i]]= i;**

**}**

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

**for (i= n;i>0;i--){**

**next[i]= B[A[i]];**

**B[A[i]]= i;**

**}**

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

**P[i]= (point){i,last[i],next[i],A[i]};**

**build(root,1,n);**

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

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

**x= (x+ans)%n+1;**

**y= (y+ans)%n+1;**

**if (x>y) swap(x,y);**

**range[0][0]= x;**

**range[0][1]= y;**

**range[1][0]= 0;**

**range[1][1]= x-1;**

**range[2][0]= y+1;**

**range[2][1]= n+1;**

**ans= 0;**

**query(root);**

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

**}**

**return 0;**

**}**

**树分治**

**链**

**const int maxn= 3e5,M= 15,oo= 1e9;**

**vector<int> g[maxn];**

**char s[20];**

**int c[maxn],f[maxn][20],par[maxn],fa[maxn],per[maxn];**

**int next[maxn],d[maxn],X[maxn],Y[maxn],root[maxn];**

**int left[maxn],right[maxn],sum[maxn],Max[maxn],L[maxn];**

**int R[maxn],size[maxn];**

**int n,i,j,x,y,ans,top,tot,q;**

**void dfs(int o,int depth){**

**d[o]= depth;**

**size[o]= 1;**

**int len= g[o].size(),i,p= 0;**

**for (i= 0;i<len;i++){**

**int v= g[o][i];**

**if (!d[v]){**

**f[v][0]= fa[v]= o;**

**dfs(v,depth+1);**

**size[o]+= size[v];**

**p= size[p]>size[v]?p:v;**

**}**

**}next[o]= p;**

**}**

**bool cmp(int a,int b){**

**return d[a]<d[b];**

**}**

**void dfs1(int o){**

**if (!o) return;**

**X[o]= ++tot;Y[tot]= o;**

**root[o]= top;par[o]= per[i];**

**dfs1(next[o]);**

**}**

**void update(int o){**

**sum[o]= sum[left[o]]+sum[right[o]];**

**Max[o]= max(Max[left[o]],Max[right[o]]);**

**}**

**void build(int &o,int l,int r){**

**if (!o) o= ++top;**

**L[o]= l,R[o]= r;**

**if (l+1==r){**

**sum[o]= Max[o]= c[Y[l]];**

**return;**

**}**

**int mid= (l+r)>>1;**

**build(left[o],l,mid);**

**build(right[o],mid,r);**

**update(o);**

**}**

**void change(int o,int v,int x){**

**if (L[o]+1==R[o]){**

**sum[o]= Max[o]= x;**

**return;**

**}**

**int mid= (L[o]+R[o])>>1;**

**if (v<mid) change(left[o],v,x);**

**else change(right[o],v,x);**

**update(o);**

**}**

**int lca(int x,int y){**

**if (d[x]<d[y]) swap(x,y);**

**int i;**

**for (i= M;i>=0;i--)**

**if (d[f[x][i]]>=d[y])**

**x= f[x][i];**

**if (x==y) return x;**

**for (i= M;i>=0;i--)**

**if (f[x][i]!=f[y][i])**

**x= f[x][i],y= f[y][i];**

**return f[x][0];**

**}**

**void askMax(int o,int l,int r,int &re){**

**if (l<=L[o]&&R[o]<=r){**

**re= max(re,Max[o]);**

**return;**

**}**

**int mid= (L[o]+R[o])>>1;**

**if (l<mid) askMax(left[o],l,r,re);**

**if (mid<r) askMax(right[o],l,r,re);**

**}**

**void askSum(int o,int l,int r,int &re){**

**if (l<=L[o]&&R[o]<=r){**

**re+= sum[o];**

**return;**

**}**

**int mid= (L[o]+R[o])>>1;**

**if (l<mid) askSum(left[o],l,r,re);**

**if (mid<r) askSum(right[o],l,r,re);**

**}**

**int queryMax(int x,int y){**

**int a= lca(x,y),re= -oo;**

**for (int i= 1;i<=2;i++){**

**while (d[x]>=d[a]){**

**if (d[par[x]]<=d[a]){**

**askMax(root[x],X[a],X[x]+1,re);**

**x= fa[a];**

**}else {**

**askMax(root[x],X[par[x]],X[x]+1,re);**

**x= fa[par[x]];**

**}**

**}**

**x= y;**

**}return re;**

**}**

**int querySum(int x,int y){**

**int a= lca(x,y),re= 0;**

**for (int i= 1;i<=2;i++){**

**while (d[x]>=d[a]){**

**if (d[par[x]]<=d[a]){**

**askSum(root[x],X[a],X[x]+1,re);**

**x= fa[a];**

**}else {**

**askSum(root[x],X[par[x]],X[x]+1,re);**

**x= fa[par[x]];**

**}**

**}**

**x= y;**

**}**

**return re-c[a];**

**}**

**int main(){**

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

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

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

**g[x].push\_back(y);**

**g[y].push\_back(x);**

**}**

**dfs(1,1);**

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

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

**per[i]= i;**

**}**

**sort(per+1,per+n+1,cmp);**

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

**if (!root[per[i]]){**

**top++;x= tot+1;**

**dfs1(per[i]);**

**build(root[per[i]],x,tot+1);**

**}**

**for (i= 1;i<=M;i++)**

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

**f[j][i]= f[f[j][i-1]][i-1];**

**scanf("%d",&q);**

**for (i= 1;i<=q;i++){**

**scanf("%s%d%d",s+1, &x, &y);**

**if (s[2]=='H'){//change**

**change(root[x],X[x],y);**

**c[x]= y;**

**}else if (s[2]=='M'){//qmax**

**ans= queryMax(x,y);**

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

**}else {//qsum**

**ans= querySum(x,y);**

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

**}**

**}return 0;**

**}**

**点**

**const int maxn= 3e5,mo= 1e6+3,oo= 1e9,main\_stack=16;**

**struct node{**

**int v,pre;**

**};**

**vector<int> g[maxn];**

**vector<node> H[mo+10];**

**//char my\_stack[128<<20];**

**int sta[maxn],ans[2];**

**int inv[mo+10],c[maxn],b[maxn],vis[maxn];**

**int n,x,y,z,i,j,k,K,bl,tim,sz,now,cnt,bc,la;**

**void ex\_gcd(int a,int b,int &x,int &y){**

**if (!b){**

**x= 1;y= 0;**

**return;**

**}**

**ex\_gcd(b,a%b,x,y);**

**int t= x;**

**x= y;**

**y= t-a/b\*y;**

**}**

**void get\_ans(int x,int y){**

**if (x>y) swap(x,y);**

**int bl= 0;**

**if (!ans[0]) bl= 1;**

**else if (ans[0]>x||(ans[0]==x&&ans[1]>y)) bl= 1;**

**if (bl) ans[0]= x,ans[1]= y;**

**}**

**void get\_sz(int o,int fa){**

**sz++;**

**int len= g[o].size(),i;**

**for (i= 0;i<len;i++){**

**int v= g[o][i];**

**if (v==fa||b[v]) continue;**

**get\_sz(v,o);**

**}**

**}**

**int dfs(int o){**

**vis[o]= tim;**

**int len= g[o].size(),i,x= 0,y,sum= 1;**

**for (i= 0;i<len;i++){**

**int v= g[o][i];**

**if (vis[v]==tim||b[v]) continue;**

**y= dfs(v);**

**sum+= y;**

**x= max(x,y);**

**}**

**x= max(x,sz-sum);**

**if (x<now){**

**now= x;**

**bc= o;**

**}return sum;**

**}**

**void go(int o,int fa,int u,int k){**

**int len= g[o].size(),i;**

**k= (ll)k\*c[o]%mo;**

**int L= H[k].size();**

**if (!L) sta[++la]= k;**

**if (L<2) H[k].push\_back((node){o,u});**

**else {**

**int p= H[k][0].v>H[k][1].v?0:1;**

**if (H[k][p].v>o){**

**H[k][p].v= o;**

**H[k][p].pre= u;**

**}**

**}**

**for (i= 0;i<len;i++){**

**int v= g[o][i];**

**if (b[v]||v==fa) continue;**

**go(v,o,u,k);**

**}**

**}**

**void init(){**

**memset(b,0,sizeof(b));**

**memset(ans,0,sizeof(ans));**

**cnt= 0;**

**for (int i= 1;i<=n;i++) g[i].clear();**

**}**

**void gi(int &re){**

**char c= getchar();re= 0;**

**while (c<'0'||c>'9') c= getchar();**

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

**}**

**void MAIN(){**

**for (i= 1;i<mo;i++){**

**ex\_gcd(i,mo,inv[i],x);**

**inv[i]= (inv[i]+mo)%mo;**

**}**

**while (scanf("%d%d",&n, &K)>0){**

**init();**

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

**//scanf("%d",&c[i]);**

**gi(c[i]);**

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

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

**gi(x);gi(y);**

**g[x].push\_back(y);**

**g[y].push\_back(x);**

**}**

**do{**

**bl= 0;tim++;cnt++;**

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

**if (!b[i]&&vis[i]!=tim){**

**bl= 1;**

**sz= 0;**

**get\_sz(i,0);**

**now= oo;**

**dfs(i);**

**b[bc]= cnt;**

**int len= g[bc].size();**

**for (j= 0;j<len;j++){**

**int v= g[bc][j];**

**if (b[v]) continue;**

**go(v,bc,v,1);**

**}**

**x= (ll)K\*inv[c[bc]]%mo;**

**len= H[x].size();**

**for (j= 0;j<len;j++)**

**get\_ans(H[x][j].v,bc);**

**for (j= 1;j<=la;j++){**

**len= H[sta[j]].size();**

**x= (ll)K\*inv[(ll)c[bc]\*sta[j]%mo]%mo;**

**int len2= H[x].size();**

**for (k= 0;k<len;k++)**

**for (z= 0;z<len2;z++){**

**if (H[sta[j]][k].pre==H[x][z].pre) continue;**

**get\_ans(H[sta[j]][k].v,H[x][z].v);**

**}**

**}**

**while (la)**

**H[sta[la--]].clear();**

**}**

**}while (bl);**

**if (ans[0]) printf("%d %d\n",ans[0], ans[1]);**

**else printf("No solution\n");**

**}**

**}**

**int main()**

**{**

**/\*\_\_asm\_\_("movl %%esp, (%%eax);\n"::"a"(my\_stack):"memory");**

**\_\_asm\_\_("movl %%eax, %%esp;\n"::"a"(my\_stack+sizeof(my\_stack)-main\_stack):"%esp");\*/**

**MAIN();**

**/\*\_\_asm\_\_("movl (%%eax), %%esp;\n"::"a"(my\_stack):"%esp");\*/**

**return 0;**

**}**

**扩展欧几里德**

**int a,b,x,y;**

**void gcd(int a,int b){**

**if (!b){x= 1;y= 0;return;}**

**gcd(b,a%b);**

**int t= x;**

**x= y;**

**y= t-a/b\*y;**

**}**

**快速傅里叶变换**

**#include<complex>**

**const double PI= acos(-1),eps= 1e-8;**

**typedef complex<double> CD;**

**vector<double> a,b;**

**int n,m,i,x;**

**void FFT(vector<CD> &a,bool inv){**

**int n= a.size(),i,j,k,step,Ek;**

**for (i= j= 0;i<n;i++){**

**if (j>i) swap(a[i],a[j]);**

**k= n;**

**while (j & (k>>= 1)) j&= ~k;**

**j|= k;**

**}**

**double pi= inv ? -PI:PI;**

**for (step= 1;step<n;step<<= 1){**

**double alpha= pi/step;**

**for (k= 0;k<step;k++){**

**CD om= exp(CD(0,alpha\*k));**

**for (Ek= k;Ek<n;Ek+= step<<1){**

**int Ok= Ek+step;**

**CD t= om\*a[Ok];**

**a[Ok]= a[Ek]-t;**

**a[Ek]+= t;**

**}**

**}**

**}**

**if (inv)**

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

**a[i]/= n;**

**}**

**vector<double> operator \*(vector<double> &v1,vector<double> &v2){**

**int len1= v1.size(),len2= v2.size(),len= 2,i;**

**while (len<len1+len2) len<<= 1;**

**vector<CD> a(len,0),b(len,0);**

**for (i= 0;i<len1;i++) a[i]= v1[i];**

**for (i= 0;i<len2;i++) b[i]= v2[i];**

**FFT(a,0);**

**FFT(b,0);**

**for (i= 0;i<len;i++) a[i]\*= b[i];**

**FFT(a,1);**

**vector<double> res(len1+len2-1);**

**for (i= 0;i<len1+len2-1;i++) res[i]= a[i].real();**

**return res;**

**}**

**int main(){**

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

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

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

**a.push\_back(x);**

**}**

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

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

**b.push\_back(x);**

**}**

**a= a\*b;**

**for (i= 0;i<a.size();i++)**

**printf("%.0lf ",a[i]);**

**return 0;**

**}**

**高斯消元**

**const double eps= 1e-8;**

**const int maxn= 300;**

**double a[maxn][maxn];**

**int n,i,j;**

**bool gauss(int n){**

**int i,j,r,k;double f;**

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

**r= i;**

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

**if (fabs(a[r][i])<fabs(a[j][i])) r= j;**

**if (r!=i) for (j= 1;j<=n+1;j++) swap(a[r][j],a[i][j]);**

**if (fabs(a[i][i])<eps) return 0;**

**for (j= i+1;j<=n;j++){**

**f= a[j][i]/a[i][i];**

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

**a[j][k]-= a[i][k]\*f;**

**}**

**}**

**for (i= n;i>0;i--){**

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

**a[i][n+1]-= a[j][n+1]\*a[i][j];**

**a[i][n+1]/= a[i][i];**

**}return 1;**

**}**

**int main(){**

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

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

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

**scanf("%lf",&a[i][j]);**

**if (gauss(n))**

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

**printf("%.3lf ",a[i][n+1]);**

**else printf("Oh my God!");**

**return 0;**

**}**

**矩阵快速幂**

**const int maxn= 30;**

**typedef int matrix[maxn][maxn];**

**typedef int vec[maxn];**

**matrix a;**

**int A[maxn],F[maxn];**

**int d,n,mo,i;**

**void mul(matrix a,matrix b,matrix res){**

**matrix c= {0};int i,j,k;**

**for (i= 1;i<=d;i++)**

**for (j= 1;j<=d;j++)**

**for (k= 1;k<=d;k++)**

**c[i][j]= (c[i][j]+a[i][k]\*b[k][j]%mo)%mo;**

**memcpy(res,c,sizeof(c));**

**}**

**void ksm(matrix a,int b,matrix res){**

**matrix c= {0};**

**int i;**

**for (i= 1;i<=d;i++)**

**c[i][i]= 1;**

**while (b){**

**if (b&1) mul(c,a,c);**

**b>>= 1;mul(a,a,a);**

**}**

**memcpy(res,c,sizeof(c));**

**}**

**void trans(vec f,matrix a,vec res){**

**vec c= {0};**

**int i,j;**

**for (i= 1;i<=d;i++)**

**for (j= 1;j<=d;j++)**

**c[i]= (c[i]+f[j]\*a[j][i]%mo)%mo;**

**memcpy(res,c,sizeof(c));**

**}**

**void init(){**

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

**}**

**int main(){**

**while (scanf("%d%d%d",&d, &n, &mo)>0){**

**if (!d) break;**

**init();**

**for (i= 1;i<=d;i++){**

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

**A[i]%= mo;**

**}**

**for (i= 1;i<=d;i++){**

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

**F[i]%= mo;**

**}**

**for (i= 2;i<=d;i++) a[i][i-1]= 1;**

**for (i= 1;i<=d;i++) a[i][d]= A[d-i+1];**

**ksm(a,n-1,a);**

**trans(F,a,F);**

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

**}return 0;**

**}**

**中国剩余定理**

**const int maxn= 3001;**

**int a[maxn],m[maxn];**

**int i,n,ans;**

**void gcd(int a,int b,int &x,int &y){**

**if (!b){x= 1;y= 0;return;}**

**gcd(b,a%b,x,y);**

**int t= x;**

**x= y;**

**y= t-a/b\*y;**

**}**

**int china(int \*a,int \*m,int n){**

**int M= 1,i,x= 0,p,q;**

**for (i= 1;i<=n;i++) M\*= m[i];**

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

**int w= M/m[i];**

**gcd(w,m[i],p,q);**

**x= (x+w\*p%M\*a[i])%M;**

**}return (x+M)%M;**

**}**

**int main()**

**{**

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

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

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

**ans= china(a,m,n);**

**printf("%d",ans);**

**return 0;**

**}**

**卢卡斯**

**const int maxn= 3001;**

**int c[maxn];**

**int p,n,m,ans,i;**

**void gcd(int a,int b,int &x,int &y){**

**if (!b){x= 1;y= 0;return;}**

**gcd(b,a%b,x,y);**

**int t= x;**

**x= y;**

**y= t-a/b\*y;**

**}**

**int lucas(int m,int n,int p){**

**int re= 1,x,y;**

**while (n||m){**

**if (n%p>m%p) return 0;**

**re= re\*c[m%p]%p;**

**gcd(c[m%p-n%p]\*c[n%p]%p,p,x,y);**

**re= re\*x%p;**

**n/= p;m/= p;**

**}return (re+p)%p;**

**}**

**int main()**

**{**

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

**c[0]= 1;**

**for (i= 1;i<=p;i++)**

**c[i]= c[i-1]\*i%p;**

**ans= lucas(m,n,p);**

**printf("%d",ans);**

**return 0;**

**}**

**大步小步算法**

**ll a,b,n,ans;**

**void gcd(ll a,ll b,ll &x,ll &y){**

**if (!b){x= 1;y= 0;return;}**

**gcd(b,a%b,x,y);**

**ll t= x;**

**x= y;**

**y= t-a/b\*y;**

**}**

**ll ksm(ll a,ll b,ll mo){**

**ll t= 1;**

**while (b){**

**if (b&1) t= t\*a%mo;**

**b>>= 1;a= a\*a%mo;**

**}return t;**

**}**

**ll logMod(ll a,ll b,ll n){**

**ll v,m,i,e= 1,y;**

**map<ll,ll> X;**

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

**gcd(ksm(a,m,n),n,v,y);**

**X[1]= 0;**

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

**e= e\*a%n;**

**if (!X.count(e)) X[e]= i;**

**}**

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

**if (X.count(b)) return i\*m+X[b];**

**b= b\*v%n;**

**}return -1;**

**}**

**int main(){**

**scanf("%lld %lld %lld",&a, &b, &n);**

**ans= logMod(a,b,n);**

**printf("%lld",(ans+n)%n);**

**return 0;**

**}**

**辛普森**

**const double eps= 1e-8,oo= 1e8;**

**struct point{**

**double x,y;**

**};**

**typedef point vec;**

**struct circle{**

**point c;**

**double r;**

**point getpoint(double a){**

**return (point){c.x+r\*cos(a),c.y+r\*sin(a)};**

**}**

**};**

**struct node{**

**int k;**

**double x,y,a,b;**

**double count(double x){**

**if (k) return sqrt(b\*b-(x-a)\*(x-a));**

**else return a\*x+b;**

**}**

**};**

**const int maxn= 3001;**

**vector<point> sol;**

**circle C[maxn];**

**node a[maxn];**

**double h[maxn],r[maxn];**

**double alpha,ans,cnt,L,R;**

**int n,i,tot;**

**int dcmp(double x){**

**if (fabs(x)<eps) return 0;**

**return x<0 ? -1:1;**

**}**

**vec operator +(vec a,vec b){a.x+= b.x;a.y+= b.y;return a;}**

**vec operator -(vec a,vec b){a.x-= b.x;a.y-= b.y;return a;}**

**vec operator \*(double t,vec a){a.x\*= t;a.y\*= t;return a;}**

**vec operator /(vec a,double t){a.x/= t;a.y/= t;return a;}**

**bool operator <(vec a,vec b){return dcmp(a.x-b.x)<0||(!dcmp(a.x-b.x)&&dcmp(a.y-b.y)<0);}**

**bool operator ==(vec a,vec b){return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y);}**

**double cj(vec a,vec b){return a.x\*b.y-a.y\*b.x;}**

**double dj(vec a,vec b){return a.x\*b.x+a.y\*b.y;}**

**double length(vec a){return sqrt(dj(a,a));}**

**double polarAngle(vec a){return atan2(a.y,a.x);}**

**bool cmp(node a,node b){return a.x<b.x||(a.x==b.x&&a.y<b.y);}**

**int tangents(circle c1,circle c2){**

**sol.clear();**

**if (c1.r<c2.r) swap(c1,c2);**

**point o1= c1.c,o2= c2.c;**

**double r1= c1.r,r2= c2.r,d= length(o1-o2);**

**if (!dcmp(d)) return 0;**

**double da= acos((r1-r2)/d),base= polarAngle(o2-o1);**

**sol.push\_back(c1.getpoint(da+base));**

**sol.push\_back(c2.getpoint(da+base));**

**return 1;**

**}**

**void prepare(){**

**a[++tot]= (node){1,-C[1].r,C[1].r,0,C[1].r};**

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

**a[++tot]= (node){1,C[i].c.x-C[i].r,C[i].c.x+C[i].r,C[i].c.x,C[i].r};**

**if (tangents(C[i-1],C[i])){**

**point p= sol[0],q= sol[1];**

**p.y= fabs(p.y);q.y= fabs(q.y);**

**if (q<p) swap(p,q);**

**a[++tot]= (node){0,p.x,q.x,(q.y-p.y)/(q.x-p.x),p.y-p.x\*(q.y-p.y)/(q.x-p.x)};**

**}**

**}**

**sort(a+1,a+tot+1,cmp);**

**}**

**double F(double x){**

**int i;double re= -oo;**

**for (i= 1;i<=tot;i++)**

**if (dcmp(x-a[i].x)>=0&&dcmp(x-a[i].y)<=0)**

**re= max(re,a[i].count(x));**

**return re;**

**}**

**double simpson(double a,double b,double l,double m,double r){**

**return (l+4\*m+r)\*(b-a)/6;**

**}**

**double asr(double a,double b,double l,double m,double r,double eps,double A){**

**double c= (a+b)/2,m1= F((a+c)/2),m2= F((c+b)/2);**

**double L= simpson(a,c,l,m1,m),R= simpson(c,b,m,m2,r);**

**if (fabs(L+R-A)<eps) return L+R;**

**return asr(a,c,l,m1,m,eps,L)+asr(c,b,m,m2,r,eps,R);**

**}**

**int main(){**

**scanf("%d %lf",&n, &alpha);**

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

**scanf("%lf",&h[i]);**

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

**scanf("%lf",&r[i]);**

**L= oo;R= -oo;**

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

**C[i].c= (point){cnt,0};**

**C[i].r= r[i];**

**cnt+= h[i]/tan(alpha);**

**L= min(L,C[i].c.x-r[i]);**

**R= max(R,C[i].c.x+r[i]);**

**}**

**prepare();**

**double fl,fr,fm;**

**fl= F(L);fr= F(R);fm= F((L+R)/2);**

**ans= 2\*asr(L,R,fl,fm,fr,1e-6,simpson(L,R,fl,fm,fr));**

**printf("%.2lf",ans);**

**return 0;**

**}**

**高精度**

**const int MAXN = 410;**

**struct bign{**

**int len, s[MAXN];**

**bign (){**

**memset(s, 0, sizeof(s));**

**len = 1;**

**}**

**bign (int num) { \*this = num; }**

**bign (const char \*num) { \*this = num; }**

**void clean(){**

**while(len > 1 && !s[len-1]) len--;**

**}**

**bign operator = (const int num){**

**char s[MAXN];**

**sprintf(s, "%d", num);**

**\*this = s;**

**return \*this;**

**}**

**bign operator = (const char \*num){**

**for(int i = 0; num[i] == '0'; num++) ; //去前导0**

**len = strlen(num);**

**for(int i = 0; i < len; i++) s[i] = num[len - i - 1] - '0';**

**return \*this;**

**}**

**bign operator + (const bign &b) const{**

**bign c;**

**c.len = max(len, b.len);**

**for(int i = 0; i < c.len; i++){**

**c.s[i] += s[i] + b.s[i];**

**if (c.s[i] >= 10) c.s[i] -= 10, c.s[i + 1]++;**

**}**

**if (c.s[c.len + 1]) c.len++;**

**return c;**

**}**

**void operator += (const bign &b){**

**\*this = \*this + b;**

**}**

**bign operator \* (const bign &b){**

**bign c;**

**c.len = len + b.len;**

**for(int i = 0; i < len; i++)**

**for(int j = 0; j < b.len; j++)**

**c.s[i + j] += s[i] \* b.s[j];**

**for(int i = 0; i < c.len; i++){**

**c.s[i + 1] += c.s[i] / 10;**

**c.s[i] %= 10;**

**}**

**c.clean();**

**return c;**

**}**

**void operator \*= (const bign &b){**

**\*this = \*this \* b;**

**}**

**bign operator - (const bign &b){**

**bign c;**

**c.len = len;**

**for(int i = 0; i < len; i++){**

**c.s[i] += s[i] - b.s[i];**

**if (c.s[i] < 0) c.s[i] += 10, c.s[i + 1]--;**

**}**

**c.clean();**

**return c;**

**}**

**void operator -= (const bign &b){**

**\*this = \*this - b;**

**}**

**bign operator / (const bign &b){**

**bign c, f = 0;**

**for(int i = len-1; i >= 0; i--){**

**f = f \* 10;**

**f.s[0] = s[i];**

**while(f >= b){**

**f -= b;**

**c.s[i]++;**

**}**

**}**

**c.len = len;**

**c.clean();**

**return c;**

**}**

**void operator /= (const bign &b){**

**\*this = \*this / b;**

**}**

**bign operator % (const bign &b){**

**bign r = \*this / b;**

**r = \*this - r \* b;**

**return r;**

**}**

**void operator %= (const bign &b){**

**\*this = \*this % b;**

**}**

**bool operator < (const bign &b){**

**if(len != b.len) return len < b.len;**

**for(int i = len - 1; i >= 0; i--){**

**if(s[i] != b.s[i]) return s[i] < b.s[i];**

**}**

**return false;**

**}**

**bool operator > (const bign &b){**

**if(len != b.len) return len > b.len;**

**for(int i = len - 1; i >= 0; i--){**

**if(s[i] != b.s[i]) return s[i] > b.s[i];**

**}**

**return false;**

**}**

**bool operator == (const bign &b){**

**return !(\*this > b) && !(\*this < b);**

**}**

**bool operator != (const bign &b){**

**return !(\*this == b);**

**}**

**bool operator <= (const bign &b){**

**return \*this < b || \*this == b;**

**}**

**bool operator >= (const bign &b){**

**return \*this > b || \*this == b;**

**}**

**string str() const{**

**string res = "";**

**for(int i = 0; i < len; i++) res = char(s[i]+'0') + res;**

**return res;**

**}**

**};**

**istream& operator >> (istream &in, bign &x){**

**string s;**

**in >> s;**

**x = s.c\_str();**

**return in;**

**}**

**ostream& operator << (ostream &out, const bign &x){**

**out << x.str();**

**return out;**

**}**