

AI模板

Baileys Thea Starry



2016-10-16

NJUST

目录

(一) Readme 3

(二) 图论 3

1. R最短路Dijkastra 3

2. R最短路Floyd 4

3. R最短路SPFA 4

4. R最小生成树Kruskal 4

5. R最小生成树Prim 5

6. R强连通分量Tarjan 6

7. D连通分量Tarjan 7

8. D连通分量korasaju 8

9. R曼哈顿生成树 8

10. R欧拉路径输出 11

(三) 数据结构 13

1. 并查集相关 13

2. Drmq 13

3. T-RMQ 14

4. T-LCA 15

5. R堆 18

6. R平衡树avl 18

7. R平衡树sbt 20

8. D树状数组 22

9. R树状数组二维 22

10. R主席树 22

11. R平衡树 23

12. R树链剖分 24

(四) DP 28

1. 背包多重 28

2. 背包完全 29

3. 背包混合 29

4. 轮廓线DP 30

(五) 字符串 33

1. Istringstream 33

2. T-KMP 33

3. DKMP 34

4. R字符串KMP 34

5. R字符串EXKMP 35

6. R字符串Manacher 37

7. Dmanacher 38

8. D最小表示法 38

9. T最大最小表示法 39

10. T后缀数组 39

11. AC自动机 41

(六) 计算几何 42

1. 求线段交点 42

2. 判断直线与直线的位置关系 43

3. T三维凸包 44

(七) 数学 46

1. 公式 46

2. O(2√N+2√M)优化 49

3. 线性筛打表 49

4. 根号N求欧拉函数 50

5. 求逆元 50

6. Matrix 51

7. Gauss 52

8. 扩展欧几里得二元一次最小整数解 53

9. 中国剩余定理 53

10. BSGS 54

11. Miller-Robin测试 57

12. Pollard *ρ*整数分解法 58

13. n!中因子m的个数 59

14. T-polya 59

(八) 博弈 59

(九) 网络流 60

1. R二分图最大匹配 60

2. R二分图最佳匹配 61

3. R最大流sap 64

4. R最大流EK 66

5. R最大流Dinic 68

6. R费用流Spfa 70

7. R费用流Zkw 72

8. T无源汇有上下界最大流 75

9. T有源汇有上下界最大流 77

10. T有源汇有上下节最小流 80

(十) 其他 83

1. D莫队 83

2. R2-SAT 84

3. R高精度 86

4. R莫队分块 90

5. R莫队曼哈顿 91

6. Java大数 95

7. STL 96

# Readme

|  |
| --- |
| **WA了三次重新读题**  **初始化,可以为负的情况,位置错了**  **入坑容易退坑难**  **string很慢**  **读题时mark高精度**  **输出格式**  **赛场带vimrc配置**  **试机：递归深度，lld，PE，除以0，下标为负，重启机器测还原**  **^优先级竟然比<要低** |
| syntax on  set nu cindent sw=4 sts=4 ts=4  set mouse=a  nmap <F2> :w<CR>  nmap <F3> :make %< <CR>  nmap <F4> :vs %<.in <CR>  nmap <F5> :!./%< < %<.in <CR>  nmap <F6> :!./%< < %<.in > %<.out <CR> :vs %<.out <CR>  nmap <F7> ggVG "+y  nmap <F12> :wq<CR>  nmap <TAB> <SPACE><SPACE><SPACE><SPACE>  colorscheme delek  inoremap ( ()<ESC>i  inoremap [ []<ESC>i  inoremap " ""<ESC>i  inoremap ' ''<ESC>i  inoremap { {<CR>}<ESC>i |

# 图论

## R最短路Dijkastra

|  |
| --- |
| void Work()  {  int S=1, T=n, u=1, v, cur, cnt=0;  s[S]=0;  while(cnt++<n)  {  vis[u]=true;  for(int i=1; i<=n; i++)  {  if(vis[i]) continue;  if(s[i]>s[u]+d[u][i]) s[i]=s[u]+d[u][i];  if(s[i]<cur) cur=s[i], v=i;  }  u=v, cur=s[0];  }  printf("%d\n", s[T]);  return ;  } |

## R最短路Floyd

|  |
| --- |
| void Work()  {  for(int k=1; k<=n; k++)  for(int i=1; i<=n; i++)  for(int j=1; j<=n; j++)  if(i!=j&&j!=k&&i!=k)  d[i][j]=Min(d[i][j], d[i][k]+d[k][j]);  printf("%d\n", d[1][n]);  return ;  } |

## R最短路SPFA

|  |
| --- |
| void Work()  {  s[S=1]=0, T=n;  q.push(S), vis[S]=true;  while(!q.empty())  {  int u=q.front();  for(int i=1; i<=n; i++)  if(s[i]>s[u]+d[u][i])  {  s[i]=s[u]+d[u][i];  if(!vis[i]) q.push(i), vis[i]=true;  }    q.pop(), vis[u]=false;  }  printf("%d\n", s[T]);  return ;  } |

## R最小生成树Kruskal

|  |
| --- |
| void Read()  {  for(int i=1; i<=n; i++)  for(int j=1; j<=n; j++)  scanf("%d", &d[i][j]);  while(!q.empty()) q.pop();  for(int i=1; i<=n; i++)  for(int j=i+1; j<=n; j++)  {  p.u=i, p.v=j, p.c=d[i][j];  q.push(p);  }  return ;  }  int Rt(int x)  {  return (x==fa[x])?(x):(fa[x]=Rt(fa[x]));  }  void Work()  {  for(int i=1; i<=n; i++) fa[i]=i;  int ans=0, cnt=1;  while(cnt<n)  {  int u=Rt(q.top().u), v=Rt(q.top().v), c=q.top().c; //每次选最短边  q.pop();  if(u==v) continue; //如果端点已经在同一集合 不更新  ++cnt, ans+=c; //加入生成树  if(u<v) fa[v]=u; //合并端点  else fa[u]=v;  }  printf("%d\n", ans);  return ;  } |

## R最小生成树Prim

|  |
| --- |
| void Work()  {  memset(vis, false, sizeof vis);  memset(\_min, 0xf, sizeof \_min);  int u=1, v, cur=\_min[0], cnt=1, ans=0; //任意找一个点加入集合  while(cnt++<n)  {  vis[u]=true;  for(int i=1; i<=n; i++)  {  if(vis[i]||u==i) continue;  if(\_min[i]>d[u][i]) \_min[i]=d[u][i];//更新邻接点到目标集合的最短距离  if(\_min[i]<cur) cur=\_min[i], v=i; //取距离最近的点加入目标集合  }  u=v, ans+=cur, cur=\_min[0];  }  printf("%d\n", ans);  return ;  } |

## R强连通分量Tarjan

|  |
| --- |
| #include<iostream>  #include<cstring>  #include<cstdio>  #include<stack>  #define Min(a, b) ((a)<(b)?(a):(b))  #define Max(a, b) ((a)>(b)?(a):(b))  using namespace std;  const int maxn=100+10;  int n, ans1, ans2, tot, w[maxn];  int cur, fir[maxn], nxt[maxn\*maxn], ver[maxn\*maxn];  int idx, dfn[maxn], low[maxn];  bool ins[maxn], in[maxn], out[maxn];  stack<int> s;  void Add(int x, int y)  {  ver[++cur]=y, nxt[cur]=fir[x], fir[x]=cur;  return ;  }  void Tarjan(int u)  {  dfn[u]=low[u]=++idx;  ins[u]=true;  s.push(u);  for(int i=fir[u], v; i; i=nxt[i])  if(!dfn[v=ver[i]])  {  Tarjan(v);  low[u]=Min(low[u], low[v]);  }  else if(ins[v]) low[u]=Min(low[u], low[v]);  //low[u]=Min(low[u], dfn[v])所得结果不同 但dfn[v]和low[v]均小于low[u]  //算法只在乎dfn[u]==low[u]的节点，所以任何一种写法对整体结果都不影响  if(dfn[u]==low[u])  {  int v=s.top();  s.pop();  w[v]=++tot, ins[v]=false;  while(v!=u)  {  v=s.top();  s.pop();  w[v]=tot, ins[v]=false;  }  }  return ;  }  int main()  {  scanf("%d", &n);  for(int i=1, j; i<=n; i++) while(scanf("%d", &j)&&j) Add(i, j);  for(int i=1; i<=n; i++) if(!dfn[i]) Tarjan(i);  for(int i=1; i<=n; i++)  for(int k=fir[i], j; k; k=nxt[k])  {  if(w[i]==w[j=ver[k]]) continue;  in[w[j]]=out[w[i]]=true;  }  for(int i=1; i<=tot; i++) ans1+=(!in[i]), ans2+=(!out[i]);  if(tot==1) printf("1\n0\n");  else printf("%d\n%d\n", ans1, Max(ans1, ans2));  return 0;  } |

## D连通分量Tarjan

|  |
| --- |
| #define N 10100  std::vector<int> v[N];  int vis[N], dfn[N], low[N], stk[N], top, ins[N], clk;  void Tarjon(int x){  dfn[x] = low[x] = clk++;  stk[top++] = x;  vis[x] = ins[x] = 1;  for(int j = 0;j < v[x].size();j++){  int y = v[x][j];  if(!vis[y]){  Tarjon(y);  low[x] = min(low[x], low[y]);  } else if(ins[y]){  low[x] = min(low[x], dfn[y]);  }  }  if(low[x] == dfn[x]){  int sz = 0;  do {  ins[stk[top-1]] = 0;  top--;  sz++;  } while(stk[top] != x);  if(sz > 1) ans++;  }  } |

## D连通分量korasaju

|  |
| --- |
| 1. 在图G随便选择起点dfs，记录访问时间。  2. 在图G^T根据1得到的访问时间最大的点作为起点dfs，得到若干个树。  3. 每棵树在原图里都是一个强连通分量（由对称在转置图中同样也是）。  void dfs(int x){  vis[x] = 1;  a[clk++] = x;  for(int j = 0;j < v[x].size();j++)  if(!vis[v[x][j]])  dfs(v[x][j]);  a[clk++] = x;  }  void strong(int x, int f){  vis[x] = 1;  flag[x] = f;  tuan[f]++;  for(int j = 0;j < w[x].size();j++)  if(!vis[w[x][j]])  strong(w[x][j], f);  } |

## R曼哈顿生成树

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：曼哈顿最小生成树  题目：POJ 3241 Object Clustering  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<algorithm>  #define lowbit(x) ((x)&-(x))  using namespace std;  const int maxn=100000+10, INF=0x7f7f7f7f;  int n, m, cur, fa[maxn], a[maxn], b[maxn];  int val[maxn], pos[maxn], ans[maxn];  struct node  {  int x, y, id;  bool operator <(const node&t)const  {  return x==t.x?y<t.y:x<t.x;  }  }p[maxn], pp[maxn];  struct line  {  int u, v, d;  line(int \_u=0, int \_v=0, int \_d=0)  {  u=\_u, v=\_v, d=\_d;  }  bool operator <(const line&t)const  {  return d<t.d;  }  }s[maxn];  int Dist(node u, node v)  {  return abs(u.x-v.x)+abs(u.y-v.y);  }  void Add(int u, int v, int d)  {  s[++cur]=line(u, v, d);  return ;  }  void Update(int x, int \_val, int \_pos)  {  for(int i=x; i; i-=lowbit(i)) if(\_val<val[i])  {  val[i]=\_val;  pos[i]=\_pos;  }  return ;  }  int Query(int x, int N, int \_val=INF, int \_pos=0)  {  for(int i=x; i<=N; i+=lowbit(i)) if(val[i]<\_val)  {  \_val=val[i];  \_pos=pos[i];  }  return \_pos;  }  int Rt(int x)  {  return x==fa[x] ? x : fa[x]=Rt(fa[x]);  }  void Manhattan()  {  cur=0;  int a[maxn], b[maxn];  for(int d=0; d<4; d++)  {  if(d==1 || d==3)  {  for(int i=1; i<=n; i++) swap(p[i].x, p[i].y);  }  else if(d==2)  {  for(int i=1; i<=n; i++) p[i].x=-p[i].x;  }  sort(p+1, p+1+n);  for(int i=1; i<=n; i++) a[i]=b[i]=p[i].y-p[i].x;  sort(b+1, b+1+n);  int N=unique(b+1, b+1+n)-(b+1);  for(int i=1; i<=N; i++) val[i]=INF, pos[i]=0;  for(int i=n; i; i--)  {  a[i]=lower\_bound(b+1, b+1+N, a[i])-b;  int ret=Query(a[i], N);  if(ret) Add(p[i].id, p[ret].id, Dist(p[i], p[ret]));  Update(a[i], p[i].x+p[i].y, i);  }  }  sort(s+1, s+1+cur);  for(int i=1; i<=n; i++) fa[i]=i;  int cnt=1, i=0, u, v, d;  while(cnt<n)  {  while(++i<=cur)  {  u=Rt(s[i].u), v=Rt(s[i].v), d=s[i].d;  if(u!=v) break;  }  ans[cnt++]=d;  if(u<v) fa[v]=u;  else fa[u]=v;  }  printf("%d\n", ans[n-m]);  return ;  }  int main()  {  while(scanf("%d%d", &n, &m)!=EOF)  {  for(int i=1; i<=n; i++)  {  scanf("%d%d", &p[i].x, &p[i].y);  p[i].id=i, pp[i]=p[i];  }  Manhattan();  }  return 0;  } |

## R欧拉路径输出

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：欧拉路径输出  题目：POJ2337 catenyms  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<algorithm>  #include<cstring>  #include<cstdio>  using namespace std;  const int maxn=1000+10, maxm=26+5;  #define Min(a, b) ((a)<(b)?(a):(b))  int T, n, in[maxm], out[maxm], fa[maxm], root, put[maxn];  string s[maxn];  bool vis[maxn];  int Rt(int x)  {  return (x==fa[x])?(x):(fa[x]=Rt(fa[x]));  }  void Init()  {  root=maxm;  memset(in, 0, sizeof in);  memset(out, 0, sizeof out);  memset(vis, false, sizeof vis);  for(int i=0; i<26; i++) fa[i]=i;  return ;  }  bool Read()  {  scanf("%d", &n);  int a, b;  for(int i=1; i<=n; i++)  {  cin>>s[i];  a=s[i][0]-'a', b=s[i][s[i].size()-1]-'a';  ++in[a], ++out[b];  root=Min(root, a);  fa[Rt(b)]=Rt(a);  }  a=Rt(a);  for(int i=0; i<26; i++)  if((in[i]||out[i])&&Rt(i)!=a)  return false;  return true;  }  bool Work()  {  int \_in=0, \_out=0;  for(int i=0; i<26; i++)  {  if(in[i]==out[i]) continue;  else if(in[i]==out[i]+1) ++\_in, root=i;  else if(in[i]+1==out[i]) ++\_out;  else return false;  }  return (\_in==\_out)&&(\_in<2);  }  bool Dfs(int x, int cnt)  {  for(int i=1; i<=n; i++)  if((!vis[i])&&s[i][0]-'a'==x)  {  vis[i]=true;  put[cnt]=i;  if(cnt==n||Dfs(s[i][s[i].size()-1]-'a', cnt+1)) return true;  vis[i]=false;  }  return false;  }  int main()  {  scanf("%d", &T);  while(T--)  {  Init();  if(Read()&&Work())  {  sort(s+1, s+1+n);  Dfs(root, 1);  for(int i=1; i<n; i++) cout<<s[put[i]]<<'.';  cout<<s[put[n]]<<endl;  }  else printf("\*\*\*\n");  }  return 0;  } |

# 数据结构

## 并查集相关

|  |
| --- |
| //p[i]: i所在堆的团长，初始为i a[i]: i底下有多少个积木，初始为0  int get(int x){  if(p[x] != x){  a[x] += a[p[x]];  p[x] = get(p[x]);  }  return p[x];  } |

## Drmq

|  |
| --- |
| void build(){  for(int i = 0;i < n;i++){  dp[i][0] = a[i];  }  for(int j = 1;(1<<j) <= n;j++){  for(int i = 0;i + (1<<j) - 1 < n;i++){  dp[i][j] = \_\_gcd(dp[i][j-1], dp[i+(1<<j-1)][j-1]);  }  }  }  int ask(int l, int r){  int k = log(1.0\*(r-l+1))/log(2);  return \_\_gcd(dp[l][k], dp[r-(1<<k)+1][k]);  } |

## T-RMQ

|  |
| --- |
| void init (){  int i;  for (i=1;i<=n;i++){  scanf ("%d",&a[i].x);  f[i][0]=1;  }  }  void pre (){  int i,j,l,last;  last=a[1].x;l=1;  a[n+1].x=a[n].x-1;  for (i=1;i<=n+1;i++){  if (a[i].x==last)continue;  else {  for (j=l;j<i;j++){  a[j].l=l;  a[j].r=i-1;  }  last=a[i].x;  l=i;  }  }    }  void work (){  int i,j,t,u,k,ll,rr;  k=int(log (double(n))/log(2.0));  for (j=1;j<=k;j++)  for (i=1;i<=n;i++){  u=int(i+pow (2.0,j)-1);  if (u>n)break;  t=int(i+pow (2.0,j-1));  f[i][j]=max(f[i][j-1],f[t][j-1]);  if (a[t-1].x==a[t].x){  ll=max (i,a[t].l);  rr=min (u,a[t].r);  if (rr-ll+1>f[i][j])f[i][j]=rr-ll+1;  }  }  } |

## T-LCA

|  |
| --- |
| /\*\*  LCA在线算法O(nlogn)  主函数调用：  init();  tot=0,dir[1]=0;  dfs(1,1);  ST(2\*n-1);  int lca=LCA(u,v);  \*/  #include <stdio.h>  #include <string.h>  #include <iostream>  #include <math.h>  using namespace std;  const int maxn=40010;///节点数目  const int maxm=25;    int \_pow[maxm],m,n;///预处理2的幂  int head[maxn],ip;  int ver[maxn\*2],R[maxn\*2],first[maxn],dir[maxn],dp[maxn\*2][maxm],tot;  ///数组依次表示第u位置对应的节点存储、第x位置节点深度存储、第一次访问时间、到根节点的距离，RNQ数组，tot点依次访问标记  bool vis[maxn];  void init()  {  memset(vis,false,sizeof(vis));  memset(head,-1,sizeof(head));  ip=0;  }  struct note  {  int v,w,next;  }edge[maxn\*2];  void addedge(int u,int v,int w)  {  edge[ip].v=v,edge[ip].w=w,edge[ip].next=head[u],head[u]=ip++;  }  void dfs(int u,int dep)  {  vis[u]=true;  ver[++tot]=u,first[u]=tot,R[tot]=dep;  for(int i=head[u];i!=-1;i=edge[i].next)  {  int v=edge[i].v;  int w=edge[i].w;  if(!vis[v])  {  dir[v]=dir[u]+w;  dfs(v,dep+1);  ver[++tot]=u,R[tot]=dep;  }  }  }  void ST(int len)///RMQ预处理  {  int k=(int)log((double)len)/(log(2.0));  for(int i=1;i<=len;i++)  {  dp[i][0]=i;  }  for(int j=1;j<=k;j++)  {  for(int i=1;i+\_pow[j]-1<=len;i++)  {  int a=dp[i][j-1],b=dp[i+\_pow[j-1]][j-1];  if(R[a]<r[b]) a="dp[x][k],b=dp[y-\_pow[k]+1][k];" else="" int="" k="(int)log((double)(y-x+1)/log(2.0));" lca="" return="" x="">y)swap(x,y);  int res=RMQ(x,y);  return ver[res];  }  /\*\*  LCA（离线算法）  主函数除建边外还应调用  init();  dir[1]=0;  tarjan(1);  \*/  #include <stdio.h>  #include <string.h>  #include <iostream>  using namespace std;  const int maxn=40010;    struct note  {  int u,v,w,lca,next;  }edge[maxn\*2],edge1[805];    int head[maxn],ip,head1[maxn],ip1;///需要建两次边。1，该树的边2，需要查询的两点  int m,n;  int father[maxn],vis[maxn],dir[maxn];  ///依次表示u点的祖先、标记是否访问过，到根节点的距离  void init()  {  memset(vis,0,sizeof(vis));  memset(dir,0,sizeof(dir));  memset(head,-1,sizeof(head));  memset(head1,-1,sizeof(head1));  ip=ip1=0;  }  void addedge(int u,int v,int w)  {  edge[ip].v=v,edge[ip].w=w,edge[ip].next=head[u],head[u]=ip++;  }  void addedge1(int u,int v)  {  edge1[ip1].u=u,edge1[ip1].v=v,edge1[ip1].lca=-1,edge1[ip1].next=head1[u],head1[u]=ip1++;  }  int Find(int x)  {  if(father[x]==x)  return x;  return father[x]=Find(father[x]);  }  void Union(int x,int y)  {  x=Find(x);  y=Find(y);  if(x!=y)  father[y]=x;  }  void tarjan(int u)  {  vis[u]=1;  father[u]=u;  for(int i=head[u];i!=-1;i=edge[i].next)  {  int v=edge[i].v;  int w=edge[i].w;  if(!vis[v])  {  dir[v]=dir[u]+w;  tarjan(v);  Union(u,v);  }  }  for(int i=head1[u];i!=-1;i=edge1[i].next)  {  int v=edge1[i].v;  if(vis[v])  {  edge1[i].lca=edge1[i^1].lca=father[Find(v)];  }  }  } |

## R堆

|  |
| --- |
| void Maintain(int k)//维护以k为根的子堆  {  while(kl<=n)  {  if(kr<=n&&a[kl]>a[kr]&&a[k]>a[kr]) swap(a[k], a[kr]), k=kr;  else if(a[k]>a[kl]) swap(a[k], a[kl]), k=kl;  else break;  }  return ;  }  void Build()  {  for(int i=n>>1; i; i--) Maintain(i);//叶子节点自然满足条件 i从n/2开始调整  return ;  }  void Delete()//删除操作只会出现在堆顶  {  swap(a[1], a[n]);  --n;  Maintain(1);  return ;  }  void Add(int val)//新元素加入到堆底 向下调整  {  a[++n]=val;  for(int k=n; k>1&&a[k]<a[k>>1]; k>>=1) swap(a[k], a[k>>1]);  return ;  } |

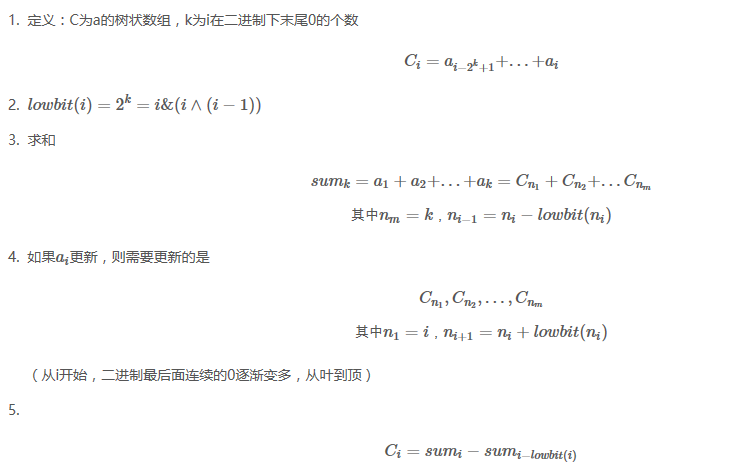
## R平衡树avl

|  |
| --- |
| int rt, cnt;  struct node  {  int ch[2], val, idx, h;  node(int v=0, int i=0){ val=v, idx=i; h=1, ch[0]=ch[1]=0; }  }tr[maxn];  void Rotate(int &k, int f)  {  int p=tr[k].ch[f];  tr[k].ch[f]=tr[p].ch[!f];  tr[p].ch[!f]=k;  tr[k].h=max(tr[tr[k].ch[0]].h, tr[tr[k].ch[1]].h)+1;  tr[p].h=max(tr[tr[p].ch[0]].h, tr[tr[p].ch[1]].h)+1;  k=p;  return ;  }  void dbRotate(int &k, int f)  {  Rotate(tr[k].ch[f], !f);  Rotate(k, f);  return ;  }  void Maintain(int &k)  {  if(tr[tr[k].ch[0]].h==tr[tr[k].ch[1]].h+2)  {  if(tr[tr[k].ch[0]].h==tr[tr[tr[k].ch[0]].ch[0]].h+1) Rotate(k, 0);  else dbRotate(k, 0);  }  else if(tr[tr[k].ch[0]].h+2==tr[tr[k].ch[1]].h)  {  if(tr[tr[k].ch[1]].h==tr[tr[tr[k].ch[1]].ch[1]].h+1) Rotate(k, 1);  else dbRotate(k, 1);  }  return ;  }  void Insert(int &k, int val, int idx)  {  if(!k) tr[k=++cnt]=node(val, idx);  else  {  Insert(tr[k].ch[val>tr[k].val], val, idx);  Maintain(k);  }  return ;  }  int Delete(int &k, int val)//若val不存在 删去第一个比它小的数  {  int ans=0;  if(val==tr[k].val || (val<tr[k].val&&!tr[k].ch[0]) || (val>tr[k].val&&!tr[k].ch[1]))  {  if(!tr[k].ch[0] || !tr[k].ch[1])  {  ans=tr[k].val;  k=tr[k].ch[0]+tr[k].ch[1];  return ans;  }  else ans=tr[k].val=Delete(tr[k].ch[0], val);  }  else ans=Delete(tr[k].ch[val>tr[k].val], val);  Maintain(k);  return ans;  }  int Query(int k, int f)  {  while(tr[k].ch[f]) k=tr[k].ch[f];  return k;  } |

## R平衡树sbt

|  |
| --- |
| struct node  {  int ch[2], val, idx, siz;  node(int v=0, int i=0){ val=v, idx=i; siz=1, ch[0]=ch[1]=0; }  }tr[maxn];  void Rotate(int &k, int f)  {  int p=tr[k].ch[f];  tr[k].ch[f]=tr[p].ch[!f];  tr[p].ch[!f]=k;  tr[k].siz=tr[tr[k].ch[0]].siz+tr[tr[k].ch[1]].siz+1;  tr[p].siz=tr[tr[p].ch[0]].siz+tr[tr[p].ch[1]].siz+1;  k=p;  return ;  }  void dbRotate(int &k, int f)  {  Rotate(tr[k].ch[f], !f);  Rotate(k, f);  return ;  }  void Maintain(int &k, int f)  {  if(tr[tr[tr[k].ch[f]].ch[f]].siz>tr[tr[k].ch[!f]].siz) Rotate(k, f);  else if(tr[tr[tr[k].ch[f]].ch[!f]].siz>tr[tr[k].ch[!f]].siz) dbRotate(k, f);  else return ;  Maintain(tr[k].ch[0], 0);  Maintain(tr[k].ch[1], 1);  Maintain(k, 0);  Maintain(k, 1);  return ;  }  void Insert(int &k, int val, int idx)  {  if(!k) tr[k=++cnt]=node(val, idx);  else  {  Insert(tr[k].ch[val>tr[k].val], val, idx);  ++tr[k].siz;  Maintain(k, val>tr[k].val);  }  return ;  }  int Delete(int &k, int val)//若val不存在 删去第一个比它小的数  {  int ans=0;  if(val==tr[k].val || (val<tr[k].val&&!tr[k].ch[0]) || (val>tr[k].val&&!tr[k].ch[1]))  {  if(!tr[k].ch[0] || !tr[k].ch[1])  {  ans=tr[k].val;  k=tr[k].ch[0]+tr[k].ch[1];  return ans;  }  else ans=tr[k].val=Delete(tr[k].ch[0], val);  }  else ans=Delete(tr[k].ch[val>tr[k].val], val);  --tr[k].siz;  return ans;  }  int Query(int k, int f)  {  while(tr[k].ch[f]) k=tr[k].ch[f];  return k;  } |

## D树状数组



## R树状数组二维

|  |
| --- |
| void Update(int x, int y, lint val)  {  for(int i=x; i<=n; i+=lowbit(i))  for(int j=y; j<=n; j+=lowbit(j))  p[x&1][y&1][i][j]^=val;  return ;  }  lint Query(int x, int y, lint ans=0ll)  {  for(int i=x; i; i-=lowbit(i))  for(int j=y; j; j-=lowbit(j))  ans^=p[x&1][y&1][i][j];  return ans;  } |

## R主席树

|  |
| --- |
| int T, n, m, a[maxn], tot, id[maxn];  struct node  {  int ch[2], val;  }tr[maxn\*50];  map<int, int> mmp;  void Build(int &k, int l=1, int r=n)  {  k=++tot;  tr[k].ch[0]=tr[k].ch[1]=tr[k].val=0;  if(l==r) return ;  Build(tr[k].ch[0], l, mid);  Build(tr[k].ch[1], mid+1, r);  return ;  }  void Insert(int &k, int p, int x, int v, int l=1, int r=n)  {  tr[k=++tot]=tr[p], tr[k].val+=v;  if(l==r) return ;  if(x<=mid) Insert(tr[k].ch[0], tr[p].ch[0], x, v, l, mid);  else Insert(tr[k].ch[1], tr[p].ch[1], x, v, mid+1, r);  return ;  }  int Query1(int k, int x, int l=1, int r=n)  {  if(l==r) return tr[k].val;  if(x<=mid) return Query1(tr[k].ch[0], x, l, mid);  return tr[tr[k].ch[0]].val+Query1(tr[k].ch[1], x, mid+1, r);  }  int Query2(int k, int cnt, int l=1, int r=n)  {  if(l==r) return l;  if(cnt<=tr[tr[k].ch[0]].val) return Query2(tr[k].ch[0], cnt, l, mid);  return Query2(tr[k].ch[1], cnt-tr[tr[k].ch[0]].val, mid+1, r);  } |

## R平衡树

|  |
| --- |
| int rt, cnt;  struct node  {  int ch[2], val, idx, rval;  node(int v=0, int i=0, int r=0){ val=v, idx=i, rval=r; ch[0]=ch[1]=0; }  }tr[maxn];  void Rotate(int &k, int f)  {  int p=tr[k].ch[f];  tr[k].ch[f]=tr[p].ch[!f];  tr[p].ch[!f]=k;  k=p;  return ;  }  void Insert(int &k, int val, int idx)  {  if(!k) tr[k=++cnt]=node(val, idx, rand());  else  {  int f=(val>tr[k].val);  Insert(tr[k].ch[f], val, idx);  if(tr[k].rval<tr[tr[k].ch[f]].rval) Rotate(k, f);  }  return ;  }  void Delete(int &k, int val)  {  if(tr[k].val==val)  {  if(tr[k].ch[0]&&tr[k].ch[1])  {  int f=tr[tr[k].ch[1]].rval>tr[tr[k].ch[0]].rval;  Rotate(k, f);  Delete(tr[k].ch[f], val);  }  else k=tr[k].ch[!tr[k].ch[0]];  }  else Delete(tr[k].ch[val>tr[k].val], val);  return ;  }  int Query(int k, int f)  {  while(tr[k].ch[f]) k=tr[k].ch[f];  return k;  } |

## R树链剖分

|  |
| --- |
| int T, n;  int cur, fir[maxn], ver[maxn<<1], nxt[maxn<<1], cst[maxn<<1], eid[maxn<<1];  int tot, siz[maxn], son[maxn], dep[maxn], fa[maxn], fid[maxn], top[maxn], fw[maxn], idx[maxn], from[maxn];  struct node  {  int mx, mn;  bool f;  }tr[maxn<<2];  void Add(int u, int v, int c, int i)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cst[cur]=c, eid[cur]=i;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cst[cur]=c, eid[cur]=i;  return ;  }  void Read()  {  cur=0;  memset(fir, 0, sizeof fir);  scanf("%d", &n);  for(int i=1, u, v, c; i<n; i++)  {  scanf("%d%d%d", &u, &v, &c);  Add(u, v, c, i);  }  return ;  }  void Dfs1(int u, int f, int d)  {  fa[u]=f, dep[u]=d, siz[u]=1;  for(int i=fir[u], v, tmp=0; i; i=nxt[i]) if((v=ver[i])!=f)  {  fw[v]=cst[i], from[v]=eid[i];  Dfs1(v, u, d+1);  siz[u]+=siz[v];  if(!son[u] || siz[v]>tmp) son[u]=v, tmp=siz[v];  }  return ;  }  void Dfs2(int u, int tp)  {  fid[u]=++tot, top[u]=tp;  if(!son[u]) return ;  Dfs2(son[u], tp);  for(int i=fir[u], v; i; i=nxt[i]) if((v=ver[i])!=fa[u] && v!=son[u]) Dfs2(v, v);  return ;  }  void Build(int k=1, int l=1, int r=n)  {  tr[k].mx=-INF, tr[k].mn=INF, tr[k].f=false;  if(l==r) return ;  Build(kl, l, mid);  Build(kr, mid+1, r);  return ;  }  void Init()  {  memset(son, 0, sizeof son);  Dfs1(1, 0, 1);  tot=0;  Dfs2(1, 1);  return ;  }  inline void Ne(int k)  {  tr[k].mx=-tr[k].mx, tr[k].mn=-tr[k].mn, tr[k].f^=1;  swap(tr[k].mx, tr[k].mn);  return ;  }  void Push\_down(int k)  {  if(tr[k].f)  {  Ne(kl);  Ne(kr);  tr[k].f=false;  }  return ;  }  void Push\_up(int k)  {  tr[k].mx=max(tr[kl].mx, tr[kr].mx);  tr[k].mn=min(tr[kl].mn, tr[kr].mn);  return ;  }  void Update(int x, int v, int k=1, int l=1, int r=n)  {  if(l==r)  {  tr[k].mx=tr[k].mn=v, tr[k].f=false;  return ;  }  Push\_down(k);  if(x<=mid) Update(x, v, kl, l, mid);  else Update(x, v, kr, mid+1, r);  Push\_up(k);  return ;  }  int Query(int ll, int rr, int k=1, int l=1, int r=n)  {  if(rr< l||r< ll) return -INF;  if(ll<=l&&r<=rr) return tr[k].mx;  Push\_down(k);  int tl=Query(ll, rr, kl, l, mid), tr=Query(ll, rr, kr, mid+1, r);  Push\_up(k);  return max(tl, tr);  }  void Negate(int ll, int rr, int k=1, int l=1, int r=n)  {  if(rr< l||r< ll) return ;  if(ll<=l&&r<=rr)  {  Ne(k);  return ;  }  Push\_down(k);  Negate(ll, rr, kl, l, mid);  Negate(ll, rr, kr, mid+1, r);  Push\_up(k);  return ;  }  int Find(int u, int v)  {  int ans=-INF, tu=top[u], tv=top[v], tmp;  while(tu!=tv)  {  if(dep[tu]<dep[tv]) swap(tu, tv), swap(u, v);  tmp=Query(fid[tu], fid[u]);  ans=max(ans, tmp);  u=fa[tu], tu=top[u];  }  if(u==v) return ans;  if(dep[u]>dep[v]) swap(u, v);  tmp=Query(fid[son[u]], fid[v]);  return max(ans, tmp);  }  void Change(int u, int v)  {  int tu=top[u], tv=top[v];  while(tu!=tv)  {  if(dep[tu]<dep[tv]) swap(tu, tv), swap(u, v);  Negate(fid[tu], fid[u]);  u=fa[tu], tu=top[u];  }  if(u==v) return ;  if(dep[u]>dep[v]) swap(u, v);  Negate(fid[son[u]], fid[v]);  return ;  }  void Work()  {  Build();  for(int i=2; i<=n; i++) Update(idx[from[i]]=fid[i], fw[i]);  char s[10];  while(scanf("%s", s)!=EOF&&s[0]!='D')  {  int a, b;  scanf("%d%d", &a, &b);  if(s[0]=='C') Update(idx[a], b);  else if(s[0]=='Q') printf("%d\n", Find(a, b));  else Change(a, b);  }  return ;  } |

# DP

## 背包多重

|  |
| --- |
| void Zero(int i){  for(int j=m; j>=v[i]; j--) f[j]=max(f[j], f[j-v[i]]+v[i]);  return ;  }  void Com(int i){  for(int j=v[i]; j<=m; j++) f[j]=max(f[j], f[j-v[i]]+v[i]);  return ;  }  void Multi(int i){  for(int j=0; j<v[i]; j++){  fir=tail=1;  for(int k=j, h=0; k<=m; k+=v[i], h++){  while(fir!=tail&&k-p[fir]>v[i]\*c[i]) ++fir;//p[]记录队列中元素在原数组中的下标 保证队列区间在范围内  int tmp=f[k]-h\*v[i]; //将f[i-1][k]放入队列  while(fir!=tail&&q[tail-1]<tmp) --tail;  q[tail++]=tmp, p[tail]=k;  f[k]=q[fir]+h\*v[i];  }  }  return ;  }  int main(){  while(scanf("%d%d", &n, &m)!=EOF){  memset(f, 0, sizeof f);  for(int i=1; i<=n; i++) scanf("%d", &v[i]);  for(int i=1; i<=n; i++) scanf("%d", &c[i]);  for(int i=1; i<=n; i++){  if(c[i]==1) Zero(i);  else if(c[i]\*v[i]>=m) Com(i);  else Multi(i);  }  int ans=0;  for(int i=1; i<=m; i++) ans+=(f[i]==i);  cout<<ans<<endl;  }  return 0;  } |

## **背包完全**

|  |
| --- |
| int n=5, v[10]={0, 50, 25, 10, 5, 1}, m, f[maxm];  int main(){  while(scanf("%d", &m)!=EOF) {  memset(f, 0, sizeof f);  f[0]=1;  for(int i=1; i<=n; i++)  for(int j=v[i]; j<=m; j++)//f[i][j]=f[i-1][j]+f[i][j-v[i]];  f[j]=f[j]+f[j-v[i]];  cout<<f[m]<<endl;  }  return 0;  } |

## **背包混合**

|  |
| --- |
| int main()  {  while(scanf("%d%d", &n, &m)!=EOF)  {  for(int i=1; i<=n; i++)  {  scanf("%d", &v[i]);  v1[i]=-v[i];  }  for(int i=1; i<=n; i++) scanf("%d", &c[i]);  N=0;  for(int i=1; i<=n; i++)  for(int sum=0, tmp=1; sum<=c[i]; tmp<<=1)//二进制拆分  if(sum+tmp<c[i]) v2[++N]=tmp\*v[i], w[N]=tmp, sum+=tmp;  else  {  v2[++N]=(c[i]-sum)\*v[i], w[N]=(c[i]-sum);  break;  }  memset(f, 0x7f, sizeof f);  f[0]=0;  for(int i=1; i<=N; i++)  for(int j=p; j>=v2[i]; j--)  if(f[j-v2[i]]!=INF) f[j]=min(f[j], f[j-v2[i]]+w[i]);  for(int i=1; i<=n; i++)  for(int j=p+v1[i]; j>=0; j--)  if(f[j-v1[i]]!=INF) f[j]=min(f[j], f[j-v1[i]]+1);  if(f[m]!=INF) cout<<f[m]<<endl;  else cout<<-1<<endl;  }  return 0;  } |

## **轮廓线DP**

|  |
| --- |
| #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<algorithm>  using namespace std;  const int N=12+5, \_hash=30007, maxn=1000000+10;  typedef long long lint;  int n, m, x, y, code[N], ch[N];  char mmp[N][N];  struct hashmap{  int cur, fir[\_hash], nxt[maxn];  lint st[maxn], f[maxn];  void clear(){  cur=0;  memset(fir, 0, sizeof fir);  return ;  }  void push(lint s, lint ans) {  int u=s%\_hash;//取余构造哈希函数  for(int i=fir[u]; i; i=nxt[i]) if(st[i]==s){  f[i]+=ans;  return ;  }  nxt[++cur]=fir[u], fir[u]=cur, st[cur]=s, f[cur]=ans;  return ;  }  }p[2];//滚动  int Read(int cnt=0){  for(int i=1; i<=n; i++){  scanf("%s", mmp[i]+1);  for(int j=1; j<=m; j++) if(mmp[i][j]=='.') x=i, y=j, ++cnt;//(x, y) 终点坐标  }  return cnt;  }  void Decode(lint st) {//解密  for(int i=m; i>=0; i--){  code[i]=st&7;//n=12 两两配对最多6条不相交线段  st>>=3;  }  return ;  }  lint Encode(){//加密  int cnt=0;  lint st=0;  memset(ch, -1, sizeof ch);  ch[0]=0;  for(int i=0; i<=m; i++){  if(ch[code[i]]==-1) ch[code[i]]=++cnt;  code[i]=ch[code[i]];//最小表示法 回路不交叉  st<<=3;  st|=code[i];  }  return st;  }  void Shift(){  for(int i=m; i; i--) code[i]=code[i-1];  code[0]=0;  return ;  }  void DP\_blank(int i, int j, int u){  for(int k=1, t; k<=p[u].cur; k++){  Decode(p[u].st[k]);  int left=code[j-1], up=code[j];  if(left&&up){  if(left==up){  if(i==x&&j==y) {//终点  code[j-1]=code[j]=0;  if(j==m) Shift();//转移到新一行  p[u^1].push(Encode(), p[u].f[k]);  }  }  else {  code[j-1]=code[j]=0;  for(int h=0; h<=m; h++) if(code[h]==up) code[h]=left;//左线段右插头和右线段左插头合并  if(j==m) Shift();  p[u^1].push(Encode(), p[u].f[k]);  }  }  else if((left&&(!up)) || ((!left)&&up)){  t=(left ? left : up);  if(mmp[i][j+1]=='.') {//插头向右转移  code[j-1]=0, code[j]=t;  p[u^1].push(Encode(), p[u].f[k]);  }  if(mmp[i+1][j]=='.') {//插头向下转移  code[j-1]=t, code[j]=0;  if(j==m) Shift();  p[u^1].push(Encode(), p[u].f[k]);  }  }  else {  if(mmp[i][j+1]=='.'&&mmp[i+1][j]=='.') {//构建新的联通块  code[j-1]=code[j]=6;//旧联通块最多5种  p[u^1].push(Encode(), p[u].f[k]);  }  }  }  return ;  }  void DP\_block(int i, int j, int u){  for(int k=1; k<=p[u].cur; k++) {  Decode(p[u].st[k]);  code[j-1]=code[j]=0;  if(j==m) Shift();  p[u^1].push(Encode(), p[u].f[k]);  }  return ;  }  lint Work(){  int u=0;  lint ans=0;  p[u].clear();  p[u].push(0, 1);  for(int i=1; i<=n; i++)  for(int j=1; j<=m; j++){  p[u^1].clear();  if(mmp[i][j]=='.') DP\_blank(i, j, u);  else DP\_block(i, j, u);  u^=1;  }  for(int i=1; i<=p[u].cur; i++) ans+=p[u].f[i];  return ans;  }  int main(){  while(scanf("%d%d", &n, &m)!=EOF){  if(!Read()) cout<<0<<endl;  else cout<<Work()<<endl;  }  return 0;  } |

# 字符串

## Istringstream

|  |
| --- |
| #include <iostream>  #include <sstream>  using namespace std;  int main(){  string line, str;  while(getline(cin, line)){  istringstream sm(line);  while(sm >> str)  cout << str << endl;  }  return 0;  } |

## T-KMP

|  |
| --- |
| #define MAXN 1000010  int next[MAXN];  // 传入的字符串下标需要以1开头  void getNext(int m, char \*str) {  next[1]=0; //枚举模式串的每个位置，判断以当前字符结尾能够匹配到的最大前缀  for(int j=0,i=2;i<=m;i++) {  while(j>0&& str[i]!=str[j+1])  j=next[j];// 在str[i-j i-1]和str[1j] 完全匹配的前提下判断str[i]和str[j+1]是否相等,如果不相等，则减小j的值，直到匹配到完全相等位置  if(str[i]==str[j+1])  j++;// 如果能够找到以i结尾的后缀和以j+1结尾的前缀完全匹配，j自增1。  // 这里j有两种情况：  // j = 0 以i结尾的后缀找不到一个前缀和它完全匹配  // j > 0 以i结尾的后缀和以j结尾的前缀完全匹配，更新Next函数的值  next[i]=s;  }  }  // S[1n] 目标串  // T[1m] 匹配串  int KMP(int n, char \*S, int m, char \*T) {  int cnt=0;  for(int j=0,i=1;i<=n;i++) {  while(j>0&&S[i]!=T[j+1])j=next[j];  if(S[i]==T[j+1])j++;  if(j==m) {  cnt++;  j=next[j];  }  }  return cnt;  } |

## DKMP

|  |
| --- |
| #define M 100000  int pre[M], n, m;  void set(string p)  {  memset(pre, 0, sizeof(pre));  pre[0] = -1;  int m = p.length();  for(int i = 1;i < m;i++)  {  int j = pre[i-1];  while(j >= 0 && p[j+1] != p[i])  j = pre[j];  pre[i] = p[j+1] == p[i] ? j+1:-1;  }  }  int KMP(string s, string p)  {  set(p);  int j = -1, r = 0;  int n = s.length(), m = p.length();  for(int i = 0;i < n;i++)  {  while(j >= 0 && s[i] != p[j+1])  j = pre[j];  if(s[i] == p[j+1])  j++;  if(j == m-1)  {  r++;  j = pre[j];  }  }  return r;  } |

## R字符串KMP

|  |
| --- |
| int T, n, m, a[maxn], b[maxn];  int nx[maxn], ex[maxn];  int Kmp()  {  nx[0]=0;  for(int i=1, j=0; i<m; i++)  {  while(j&&b[j]!=b[i]) j=nx[j-1];  if(b[j]==b[i]) ++j;  nx[i]=j;  }    for(int i=0, j=0; i<n; i++)  {  while(j&&b[j]!=a[i]) j=nx[j-1];  if(b[j]==a[i]) ++j;  if(j==m) return i-m+2;  }  return -1;  } |

## R字符串EXKMP

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：EXKMP  题目：HDU4333 Revolving Digits  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdio>  using namespace std;  const int maxn=200000+10;  #define Max(a, b) ((a)>(b)?(a):(b))  int T, n, m, nx[maxn], ex[maxn];  char t[maxn], s[maxn];  void Read()  {  scanf("%s", t);  n=strlen(t);  for(int i=0; i<n; i++) s[i]=s[n+i]=t[i];  m=n+n;    memset(nx, 0, sizeof nx);  memset(ex, 0, sizeof ex);  return ;  }  void Exkmp()  {  nx[0]=n;  for(int i=1; i<n; i++)  {  if(t[i]!=t[i-1]) break;  nx[1]=i;  }  for(int i=2, j, k=1, p, w; i<n; i++)  {  p=k+nx[k]-1, w=nx[i-k];  if(i+w<p+1) nx[i]=w;  else  {  j=Max(p-i+1, 0);  while(i+j<n && t[i+j]==t[j]) ++j;  nx[i]=j, k=i;  }  }  for(int i=0; i<n&&i<m; i++)  {  if(s[i]!=t[i]) break;  ex[0]=i+1;  }  for(int i=1, j, k=0, p, w; i<m; i++)  {  p=k+ex[k]-1, w=nx[i-k];  if(i+w<p+1) ex[i]=w;  else  {  j=Max(p-i+1, 0);  while(i+j<m && j<n && s[i+j]==t[j]) ++j;  ex[i]=j, k=i;  }  }  return ;  }  int Kmp()  {  nx[0]=0;  for(int j=0, i=1; i<n; i++)  {  while(j && t[j]!=t[i]) j=nx[j-1];  if(t[j]==t[i]) ++j;  nx[i]=j;  }  return nx[n-1];  }  void Work()  {  int le=0, eq=0, gr=0;  Exkmp();  for(int i=0; i<n; i++)  {  if(ex[i]==n) ++eq;  else  {  if(s[i+ex[i]]>t[ex[i]]) ++gr;  else ++le;  }  }  int tmp=n-Kmp();  int cnt=(n%tmp)?(1):(n/tmp);  printf(" %d %d %d\n", le/cnt, eq/cnt, gr/cnt);  return ;  }  int main()  {  scanf("%d", &T);  for(int i=1; i<=T; i++)  {  printf("Case %d:", i);  Read(), Work();  }  return 0;  } |

## R字符串Manacher

|  |
| --- |
| int main()  {  int T=0;  string t;  while(cin>>t)  {  if(t[0]=='E') break;  string s="!\*";  for(int i=0, j=t.size(); i<j; i++) s+=t[i], s+='\*';    int f[maxn]={0}, n=s.size();  int id=0, mx=0, ans=0;  for(int i=1; i<n; i++)  {  f[i]= (mx>i)? Min(f[id\*2-i], mx-i): 1;  while(s[i-f[i]]==s[i+f[i]]) ++f[i];  ans=Max(ans, f[i]);  if(i+f[i]>mx) mx=i+f[i], id=i;  }  printf("Case %d: %d\n", ++T, ans-1);  }  return 0;  } |

## Dmanacher

|  |
| --- |
| int id, m, ans;  id = m = ans = 0;  for(int i = 1;i < len;i++){  if(i < id) p[i] = 1;  else p[i] = min(p[2\*id-i], m-i);  while(a[i+p[i]] == a[i-p[i]])  p[i]++;  ans = max(ans, p[i]);  if(i > id){  id = i;  m = p[i] + i;  }  } |

## D最小表示法

|  |
| --- |
| char str[N];  int solve()  {  int len = strlen(str), i, j, k;  for(int i = len;i < len\*2;i++)  str[i] = str[i-len];  for(i = 0, j = 1;j < len;)  {  for(k = 0;k < len;k++)  if(str[i+k] != str[j+k])  break;  if(k >= len)  return i+1;  if(str[i+k] < str[j+k])  j += k+1;  else if(str[i+k] > str[j+k])  {  int t = j;  j = max(i+k, j)+1;  i = t;  }  }  return i+1;  } |

## T最大最小表示法

|  |
| --- |
| int find\_min\_max(bool ifmin){//1：min 0：max s开始下标=1 返回字符串下标  int i,j,k;  i=1;j=2;k=0;  while (j<=l&&i<=l&&k<l){  if (s[i+k]==s[j+k]){ // 注意这里可以自身扩展，或者用mod  k++;  if (k==l)break;  }  else if (s[i+k]<s[j+k]){  if (ifmin)  j+=k+1;  else i+=k+1;  k=0;  }  else {  if (ifmin)  i+=k+1;  else j+=k+1;  k=0;  }  if (i==j){  j++;  }  }  return min (i,j);  } |

## T后缀数组

|  |
| --- |
| //倍增 O(nlgn)  int wa[maxn],wb[maxn],wv[maxn],ws[maxn];  int cmp(int \*r,int a,int b,int l){  return r[a]==r[b]&&r[a+l]==r[b+l];  }  void da(int \*r,int \*sa,int n,int m){  int i,j,p,\*x=wa,\*y=wb,\*t;  for(i=0;i<m;i++) ws[i]=0;  for(i=0;i<n;i++) ws[x[i]=r[i]]++;  for(i=1;i<m;i++) ws[i]+=ws[i-1];  for(i=n-1;i>=0;i--) sa[--ws[x[i]]]=i;  for(j=1,p=1;p<n;j\*=2,m=p){  for(p=0,i=n-j;i<n;i++) y[p++]=i;  for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;  for(i=0;i<n;i++) wv[i]=x[y[i]];  for(i=0;i<m;i++) ws[i]=0;  for(i=0;i<n;i++) ws[wv[i]]++;  for(i=1;i<m;i++) ws[i]+=ws[i-1];  for(i=n-1;i>=0;i--) sa[--ws[wv[i]]]=y[i];  for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)  x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;  }  return;  }  //DC3 O(n)  #define F(x) ((x)/3+((x)%3==1?0:tb))  #define G(x) ((x)<tb?(x)\*3+1:((x)-tb)\*3+2)  int wa[maxn],wb[maxn],wv[maxn],ws[maxn];  int c0(int \*r,int a,int b){  return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];  }  int c12(int k,int \*r,int a,int b){  if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);  else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];}  void sort(int \*r,int \*a,int \*b,int n,int m){  int i;  for(i=0;i<n;i++) wv[i]=r[a[i]];  for(i=0;i<m;i++) ws[i]=0;  for(i=0;i<n;i++) ws[wv[i]]++;  for(i=1;i<m;i++) ws[i]+=ws[i-1];  for(i=n-1;i>=0;i--) b[--ws[wv[i]]]=a[i];  return;  }  void dc3(int \*r,int \*sa,int n,int m){  int i,j,\*rn=r+n,\*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;  r[n]=r[n+1]=0;  for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;  sort(r+2,wa,wb,tbc,m);  sort(r+1,wb,wa,tbc,m);  sort(r,wa,wb,tbc,m);  for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)  rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;  if(p<tbc) dc3(rn,san,tbc,p);  else for(i=0;i<tbc;i++) san[rn[i]]=i;  for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]\*3;  if(n%3==1) wb[ta++]=n-1;  sort(r,wb,wa,ta,m);  for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;  for(i=0,j=0,p=0;i<ta && j<tbc;p++)  sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];  for(;i<ta;p++) sa[p]=wa[i++];  for(;j<tbc;p++) sa[p]=wb[j++];  return;  } |

## AC自动机

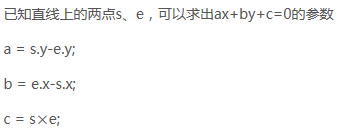
|  |
| --- |
| struct Trie  {  int next[WN][AL], fail[WN], end[WN];  int root, L;  int newnode()  {  for(int i = 0;i < AL;i++)  next[L][i] = -1;  end[L++] = 0;  return L-1;  }  void init()  {  L = 0;  root = newnode();  }  void insert(char buf[], int x)  {  int len = strlen(buf);  int now = root;  for(int i = 0;i < len;i++)  {  if(next[now][buf[i]-'A'] == -1)  next[now][buf[i]-'A'] = newnode();  now = next[now][buf[i]-'A'];  }  end[now] = x;  }  void build()  {  queue<int> Q;  fail[root] = root;  for(int i = 0;i < AL;i++)  if(next[root][i] == -1)  next[root][i] = root;  else  {  fail[next[root][i]] = root;  Q.push(next[root][i]);  }  while(!Q.empty())  {  int now = Q.front();  Q.pop();  for(int i = 0;i < AL;i++)  if(next[now][i] == -1)  next[now][i] = next[fail[now]][i];  else  {  fail[next[now][i]] = next[fail[now]][i];  Q.push(next[now][i]);  }  }  }  int query(char buf[])  {  int len = strlen(buf);  int now = root;  int res = 0;  for(int i = 0;i < len;i++)  {  now = next[now][buf[i]-'A'];  int temp = now;  while( temp != root )  {  res += end[temp];  end[temp] = 0;  temp = fail[temp];  }  }  return res;  }  }ac; |

# 计算几何

## 求线段交点

|  |
| --- |
| int delta(double x){  return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);  }  struct pt  {  double x, y;  pt(){}  pt(double x, double y): x(x), y(y){}  bool eq(pt a){  return !delta(x-a.x) && !delta(y-a.y);  }  double operator \* (const pt &b){  return x\*b.y-y\*b.x;  }  pt operator - (const pt &b){  return pt(x-b.x, y-b.y);  }  }p[N], q[N\*N];  struct seg  {  pt a, b;  seg(){}  seg(pt a, pt b) : a(a), b(b){}  int jo(seg C, pt &ans){  pt c = C.a, d = C.b;  if(max(a.x,b.x) < min(c.x,d.x)||  max(a.y,b.y) < min(c.y,d.y)||  max(c.x,d.x) < min(a.x,b.x)||  max(c.y,d.y) < min(a.y,b.y))  return 0;  int x = delta((b-a)\*(c-a)), y = delta((b-a)\*(d-a));  if(x == 0 && y == 0) return -1;  if(x == 0) ans = c;  else if(y == 0) ans = d;  else{  ans.x =-(c\*d\*(a.x-b.x)-a\*b\*(c.x-d.x))/((a-b)\*(c-d));  ans.y =(a\*b\*(c.y-d.y)-c\*d\*(a.y-b.y))/((a-b)\*(c-d));  }  if(delta(ans.x-max(a.x,b.x)) > 0 || delta(ans.x-min(a.x,b.x))<0)  return 0;  return 1;  }  }s[N]; |

## 判断直线与直线的位置关系



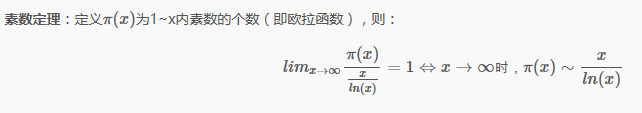
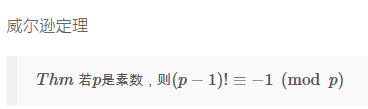
1. 容易理解的是利用性质2，得到直线方程*a*1*x*+*b*1*y*+*c*1=0与*a*2*x*+*b*2*y*+*c*2=0 。当*a*1*b*2=*a*2*b*1时 ，直线平行（特判重合的情况）；否则两直线相交，直接解得
2. 分别在直线上取两点，类似于应用2，将其中一条作为基准直线，得到两个叉积结果。若都为0，则两直线重合。若两者不为0且值相等，则两直线平行。否则相交。   
   用叉积，而不是求出斜率，这样能避免斜率不存在的情况。

## T三维凸包

|  |
| --- |
| #include<iostream>  #include<cmath>  #include<cstring>  #include<cstdio>  using namespace std;  const int N=505;  const double eps=0.000001;  struct Point  {  double x,y,z;  Point(){}  Point(double \_x,double \_y,double \_z)  {  x=\_x; y=\_y; z=\_z;  }  Point operator-(Point t1)//向量减法  {  return Point(x-t1.x,y-t1.y,z-t1.z);  }  Point operator\*(Point t2)//叉积  {  return Point(y\*t2.z-t2.y\*z,z\*t2.x-x\*t2.z,x\*t2.y-y\*t2.x);  }  double operator^(Point t3)//点积  {  return x\*t3.x+y\*t3.y+z\*t3.z;  }  }point[N];  struct Plane  {  int a,b,c;//a,b,c为三个点的编号，a,b,c要满足从凸包外面看成右手系  bool in;//表示该平面是否在凸包内  }plane[N\*10];  void Swap(Point &a,Point &b)  {  Point c;  c=a; a=b; b=c;  }  int n,plen;//计算过的面的个数  int edge[N][N];  void dfs(int p,int t);  double vol(Point p,Plane f)//p与平面abc组成的四面体的有向体积的6倍  {  Point a=point[f.a]-p,b=point[f.b]-p,c=point[f.c]-p;  return (a\*b)^c;  }  double vlen(Point a)//求向量a的模  {  return sqrt(a.x\*a.x+a.y\*a.y+a.z\*a.z);  }  void deal(int p,int t1,int t2)  {  int t=edge[t1][t2];//搜索与该边相邻的另外一个平面  if(plane[t].in)  if(vol(point[p],plane[t])>eps) dfs(p,t);  else  {  Plane add;  add.a=t2,add.b=t1,add.c=p,add.in=true;//这里注意顺序,就可以不用Swap了,add.a,add.b,add.c要成右手系  edge[add.a][add.b]=edge[add.b][add.c]=edge[add.c][add.a]=plen;  plane[plen++]=add;  }  }  void dfs(int p,int t)//递归搜索所有应该从凸包内删除的面  {  plane[t].in=false;  deal(p,plane[t].b,plane[t].a);//注意:a和b的顺序刚好跟下面的相反,为的就是搜索与边(point[plane[t].a],point[plane[t].b])相邻的另外一个平面  deal(p,plane[t].c,plane[t].b);  deal(p,plane[t].a,plane[t].c);  }  int solve(int n)//增量法,有n个点,返回计算过的平面个数,若无法构成凸包,则返回-1  {  if(n<4)//如果点数小于,则无法构成凸包,若已保证点数大于或等于4,可略去  return -1;  plen=0;//计算过的面的个数  Plane add;  for(int i=0;i<4;i++)  {  add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.in=true;  if(vol(point[i],add)>0) swap(add.a,add.b);  edge[add.a][add.b]=edge[add.b][add.c]=edge[add.c][add.a]=plen;//记录与该边相邻的其中一个面,并且该顺序在该面内成右手系  plane[plen++]=add;  }  for(int i=4;i<n;i++)  for(int j=0;j<plen;j++)  if(plane[j].in && vol(point[i],plane[j])>eps)  {  dfs(i,j);  break;  }  return plen;  }  double area(Plane a)//求平面三角形的面积  {  return vlen((point[a.b]-point[a.a])\*(point[a.c]-point[a.a]))/2.0;  }  int main()  {  cin>>n;  for(int i=0;i<n;i++) cin>>point[i].x>>point[i].y>>point[i].z;  int m=solve(n);  if(m==-1) printf("0.000\n");  else  {  double ans=0.0;  for(int i=0;i<m;i++) if(plane[i].in) ans+=area(plane[i]);  printf("%.3lf\n",ans);  }  return 0;  } |

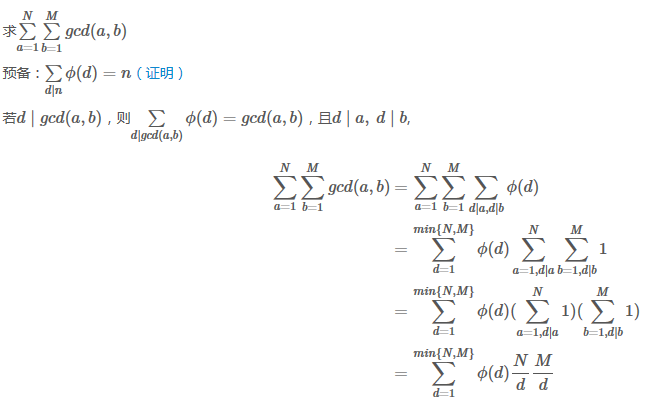
# 数学

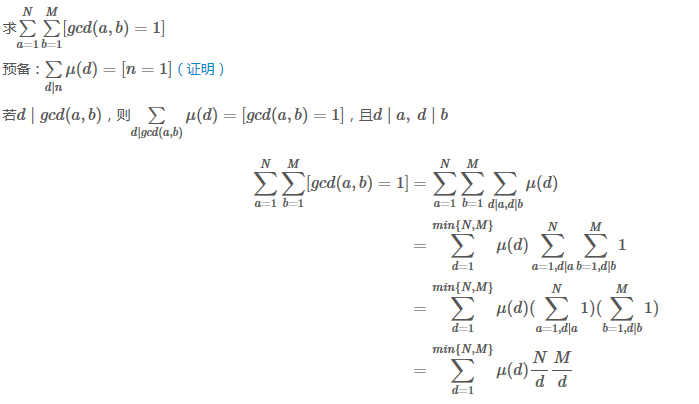
## 公式

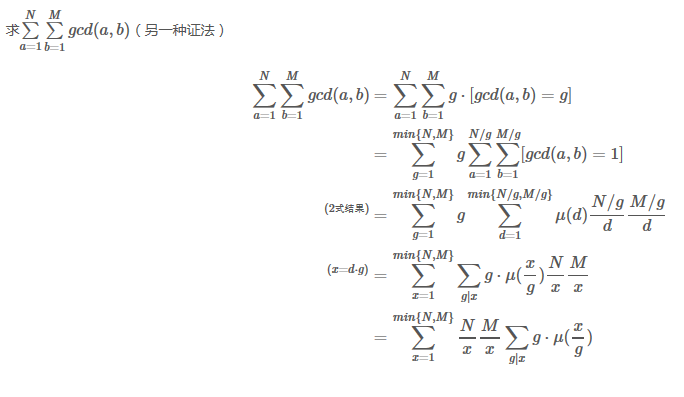
1. 
2. 
3. 若素数p满足，则
4. 若素数p满足
5. 当*n*>1 时， 1到n中与n互质的整数和为*nϕ*(*n*)/2   
   与n不互质的整数成对出现，平均值为*n/*2 ，可证。
6. 当(*n*,*m*)=1时,*ϕ*(*nm*)=*ϕ*(*n*)*ϕ*(*m*)

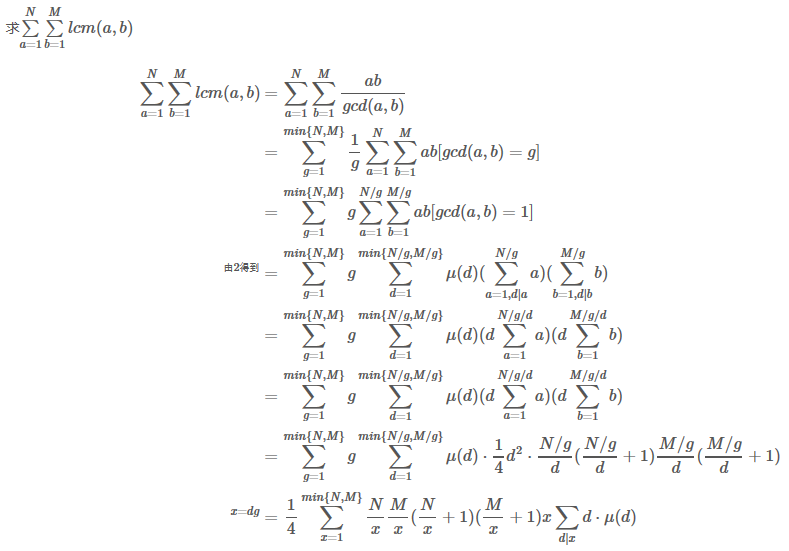
| **莫比乌斯** | **容斥** |
| --- | --- |
| d是x的约数 | y是x的倍数 |
| *μ*(*x/d*) 自变量是跳跃的 | *μ*(*y/x*) 自变量是连续的 |
| *f*(*d*) 为有限项 | *f*(*y*) 为无限项（除非规定了一个范围） |

1. 推导









## O(2√N+2√M)优化

|  |
| --- |
| /\*∑*ϕ*(*x*)(*N/x)(M/x)*对欧拉函数求前缀和，使查询更快。\*/  LL gao(int n, int m){  LL re = 0;  if(n > m) swap(n, m);  int last;  for(int i = 1;i <= n;i = last+1){  last = min(n/(n/i), m/(m/i));  re += 1LL\*(n/i)\*(m/i)\*(sum[last]-sum[i-1]);  }  return re;  } |

## 线性筛打表

|  |
| --- |
| #define N 100100  #define LL long long  int num[N], prim[N], phi[N] = {1,1}, mob[N]={1,1};  int pn = 0;  void table(){  memset(num, -1, sizeof(num));  for(int i = 2;i < N;i++){  if(num[i]) {  prim[pn++] = i;  phi[i] = i-1;  mob[i] = -1;  }  for(int j = 0;j < pn && 1LL\*i\*prim[j] < N;j++){  if(i % prim[j] == 0){  phi[i\*prim[j]] = phi[i] \* prim[j];  num[i\*prim[j]] = 0;  mob[i\*prim[j]] = 0;  break;  }  phi[i\*prim[j]] = phi[i] \* (prim[j]-1);  num[i\*prim[j]] = 0;  mob[i\*prim[j]] = -mob[i];  }  }  } |

## 根号N求欧拉函数

|  |
| --- |
| int phi(int n)  {  int ans = n;  for(int i = 2;i\*i <= n;i++)if(n % i == 0)  {  ans -= ans/i;  while(n % i == 0)  n /= i;  }  if(n != 1)  ans -= ans/n;  return ans;  } |

## 求逆元

|  |
| --- |
| /\*1. 扩展欧几里得\*/  LL extend\_Euclid(LL a, LL b, LL &x, LL &y){  if(b == 0){  x = 1; y = 0;  return a;  }  LL r = extend\_Euclid(b, a%b, y, x);  y -= a/b\*x;  return r;  }  /\*2. 费马小定理\*/  #define LL long long  LL poo(LL a, int k, int m){  LL res = 1;  while(k){  if(k & 1LL)  res = res \* a % m;  k >>= 1;  a = a\*a%m;  }  return res;  }  /\*3. 打表\*/  int rev[N];  void get\_rev(){  rev[1] = 1;  for(int n = 2;n < N;n++){  int k = P % n, t = P / n;  rev[n] = 1LL\*(P-t)\*rev[k]%P;  }  } |

## Matrix

|  |
| --- |
| #define M 10  struct Matrix{  int n, m;  int a[M][M];  Matrix(){}  Matrix(int p[M][M], int x, int y) {  memcpy(a, p, sizeof(a));  n = x, m = y;  }  void clear(){  n = m = 0;  memset(a, 0, sizeof(a));  }  void pr() {  for(int i = 0;i < n;i++)  for(int j = 0;j < m;j++)  printf("%d%c", a[i][j], " \n"[j==m-1]);  printf("\n");  }  void set(int p[M][M], int x, int y) {  n = x, m = y;  memcpy(a, p, sizeof(a));  }  Matrix operator + (const Matrix &b) const{  Matrix tmp;  tmp.n = n; tmp.m = m;  for(int i = 0;i < n;i++)  for(int j = 0;j < m;j++)  tmp.a[i][j] = a[i][j] + b.a[i][j];  return tmp;  }  Matrix operator - (const Matrix &b) const{  Matrix tmp;  tmp.n = n; tmp.m = m;  for(int i = 0;i < n;i++)  for(int j = 0;j < m;j++)  tmp.a[i][j] = a[i][j] - b.a[i][j];  return tmp;  }  Matrix operator \* (const Matrix &b) const{  Matrix tmp;  tmp.clear();  tmp.n = n; tmp.m = b.m;  for(int i = 0;i < n;i++)  for(int j = 0;j < b.m;j++)  for(int k = 0;k < m;k++) if(a[i][k] && b.a[k][j]) //avoid TLE  tmp.a[i][j] += a[i][k] \* b.a[k][j];  return tmp;  }  }; |

## Gauss

|  |
| --- |
| int gauss() {  int i, j, k, id;  for(i = 0, j = 0; i < m,j < n;) {  id = i;  for(k = i+1; k < m; k++)  if(abs(a[k][j]) > abs(a[id][j]))  id = k;  if(id != i) {  for(k = j; k < n; k++)  swap(a[i][k],a[id][k]);  }//使a[id][k](i<=k<=n)是最 大的,并把这行移到第i行  if(a[i][j] == 0) { j++; continue; }//最大的a[id][k]=0  for(k = i+1; k < m; k++) {//线性变换化0  if(a[k][j] == 0) continue;  for(int l = j; l < n; l++)  a[k][l] = a[k][l] ^ a[i][l];  }  i++,j++;  }  for(int k = i; k < m; k++) if(a[k][n] != 0) {  return -1; //根据前面说的，无解  }  return n-i;  } |

## 扩展欧几里得二元一次最小整数解

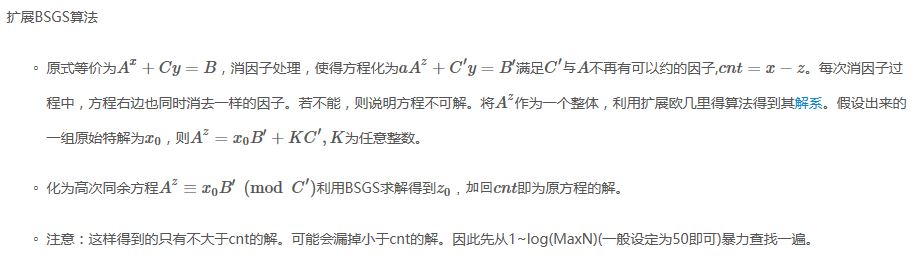
|  |
| --- |
| LL extend\_Euclid(LL a, LL b, LL &x, LL &y){  if(b==0){  x = 1; y = 0;  return a;  }  LL r = extend\_Euclid(b, a%b, y, x);  y -= a/b\*x;  return r;  }  LL equation(LL a, LL b, LL c, LL &x, LL &y)  {  LL g = extend\_Euclid(a, b, x, y);  if(c % g) return -1;  LL ran = b / g;  x \*= c/g;  if(ran < 0) ran = -ran;  x = (x%ran + ran) % ran;  return 0;  } |

## 中国剩余定理

|  |
| --- |
| /\*令Ni=M/mi,则(Ni,mi)=1,故存在xi,满足Nixi+miyi=1,  即Nixi≡1(mod mi) 则sum(aiNixi) ≡ai(mod mi)  \*/  //解方程组x=ai(mod mi) mi之间两两互质  int China(int r)  {  int M = 1, ans = 0;  for (int i = 0; i < r; ++i)  M \*= m[i];  for(int i = 0;i < r;i++)  {  int N = M/m[i];  int x, y;  extend\_Euclid(N, m[i], x, y);  ans = (ans+a[i]\*N\*x)%M;  }  ans = (ans - a[r])%M;  return (ans+M)%M;  } |

## BSGS

|  |
| --- |
| /\*--------------------------------------------  \* File Name: POJ 3696  \* Author: Danliwoo  \* Mail: Danliwoo@outlook.com  \* Created Time: 2015-10-02 22:16:59  --------------------------------------------\*/  #include <cstdio>  #include <iostream>  #include <cstring>  #include <queue>  #include <algorithm>  #include <cmath>  using namespace std;  #define N 44725  #define LL long long  #define MOD 150000 //是sqrt(1999999999\*9)要求那么大的  LL gcd(LL a, LL b)  {  return b == 0?a:gcd(b, a%b);  }  LL hs[MOD],head[MOD],next[MOD],id[MOD],top;  void insert(LL x, LL y)  {  LL k = x%MOD;  hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top++;  }  LL find(LL x)  {  LL k = x%MOD;  for(LL i = head[k]; i != -1; i = next[i])  if(hs[i] == x)  return id[i];  return -1;  }  LL mult(LL a, LL b, LL c)  {  a %= c; b %= c;  LL ret = 0, tmp = a;  while(b)  {  if(b & 1LL)  {  ret += tmp;  if(ret > c) ret -= c;  }  tmp <<= 1;  if(tmp > c) tmp -= c;  b >>= 1;  }  return ret%c;  }  LL BSGS(LL a,LL b,LL c) //a^x = b(mod c)  {  memset(head, -1, sizeof(head));  top = 1;  LL m = sqrt(c\*1.0), j;  long long x = 1, p = 1;  for(LL i = 0; i < m; ++i, p = p\*a%c)  insert(p\*b%c, i);//存的是(a^j\*b, j)  for(long long i = m; ;i += m)  {  if( (j = find(x = mult(x,p,c))) != -1 )//注意x要用mult()函数否则超LL  return i-j; //a^(ms-j)=b(mod c)  if(i > c)  break;  }  return 0;  }  int main()  {  LL o = 0;  LL n;  while(scanf("%lld", &n), n)  {  printf("Case %d: ", ++o);  LL m = n/gcd(n, 8)\*9;  if(gcd(m,10) != 1)  {  printf("0\n");  continue;  }  printf("%lld\n", BSGS(10,1,m));  }  return 0;  } |



|  |
| --- |
| LL xiao(LL &a, LL &b, LL &c)  {  LL r = 0, d, x, y, a1 = 1;  while((d = extend\_Euclid(a, c, x, y)) != 1)  {  if(b % d)  return -1;  c /= d;  b /= d;  a1 = a1\*(a/d)%c;  r++;  }  if(r == 0)  return 0;  extend\_Euclid(a1, c, x, y);  b = (b\*x%c+c)%c;  return r;  }  LL extend\_BSGS(LL a,LL b,LL c)  {  a %= c; //简化运算  LL r = 1;  for(int i = 0;i < 50;i++)  {  if((r-b) % c == 0)  return i;  r \*= a; r %= c;  }  memset(head, -1, sizeof(head));  LL cnt = xiao(a, b, c);  if(cnt == -1)  return -1;  top = 1;  if(b == 1)  return cnt;  LL m = ceil(sqrt(c\*1.0)), j;  LL x = 1, p = 1;  for(LL i = 0; i < m; ++i, p = p\*a%c)  insert(p\*b%c, i);  for(LL i = m; ;i += m)  {  if( (j = find(x = x\*p%c)) != -1 )  return i-j+cnt;  if(i > c)  break;  }  return -1;  } |

## Miller-Robin测试

判断n是否为素数

* + 由[费马小定理](http://blog.csdn.net/danliwoo/article/details/48827813#fermat)，取与n互质的a作为底数，验证*an*−1≡1(mod*n*)是否成立。
  + 当上式成立时，n是基于a的伪素数。
  + 多取几个小于n的不同的a，反复验证，若都成立，则认为n是素数。取5次即可保证比较高的正确率。
  + 不适用范围：卡麦克尔数。
* 二次探测   
  对卡麦尔数（伪素数）进行测试

二次探测定理：对素数*p*,满足*x^*2≡1(mod*p*)的小于p的正整数解*x*只有1或*p*−1

|  |
| --- |
| //从n-1的最大奇因子为指数开始判定  bool recheck(LL a, LL n, LL m, LL j)  {  LL d = po(a, m, n);//对大数运算时里面的乘法要另外算mult\_mod  if(d == 1 || d == n-1)  return true;  for(int i = 0;i < j;i++)  {  d = d\*d % n;  if(d == 1 || d == n-1)  return true;  }  return false;  }  bool Miller\_Rabin(LL n)  {  if( n < 2)return false;  if( n == 2)return true;  if( (n&1LL) == 0)return false;//偶数  srand((unsigned) time (NULL));  LL m = n-1, j = 0;  while( !(m & 1LL) )  {  m >>= 1;  j++;  }  for(int i = 0;i < 5;i++)  {  LL a = rand()%(n-2) + 2;  if( !recheck(a,n,m,j) )  return false;  }  return true;  } |

## Pollard *ρ*整数分解法

|  |
| --- |
| /\* 1. 生成两个整数a、b，计算p=(a-b,n)，直到p不为1或a,b出现循环为止。  2. 若p=n，则n为质数；否则为n的一个约数。  分解n的具体步骤如下：  1. 选取一个小的随机数*x*1 ，迭代生成*xi*=*xi*−1^2+*k* ，取k=1，若序列出现循环则退出。  2. 计算*p*=*gcd*(*xi*−1−*xi*,*n*) 直到p>1,否则返回上一步。  3. 若p=n,则n为素数。否则p为n的约数，继续分解p和n/p \*/  //找出一个因子  LL pollard\_rho(LL x,LL c)  {  LL i = 1, k = 2;  srand(time(NULL));  LL x0 = rand()%(x-1) + 1;  LL y = x0;  while(1)  {  i++;  x0 = (mult\_mod(x0,x0,x) + c)%x;//迭代公式为(x0\*x0+c)%x  LL d = gcd(y - x0,x);//gcd返回一个绝对值  if( d != 1 && d != x)  return d;  if(y == x0)  return x;  if(i == k)  {  y = x0;  k += k;  }  }  }  //对n进行素因子分解，存入factor. k设置为107左右即可  void findfac(LL n,int k)  {  if(n == 1)  return;  if(Miller\_Rabin(n))  {  factor[tol++] = n;  return;  }  LL p = n;  int c = k;  while( p >= n)  p = pollard\_rho(p,c--);//c即k表示最大搜索次数？  //值变化，防止死循环k  findfac(p,k);  findfac(n/p,k);  } |

## n!中因子m的个数

|  |
| --- |
| for (s=0;n;s+=n/=m);  //→\_→ |

## T-polya

|  |
| --- |
| int ksm(int x,int y)  {  int i,j=x,k=1;  while(y)  {  if(y%2) k=k\*j;  j=j\*j;  y/=2;  }  return k;  }  void work(void)  {  int i;  ans=0;  for(i=1;i<=m;i++) ans+=ksm(n,gcd(i,m));  if(m%2==0)  {  ans+=m/2\*ksm(n,(m+1)/2);  ans+=m/2\*ksm(n,m/2+1);  ans/=m\*2;  }  else  {  ans+=m\*ksm(n,(m+1)/2);  ans/=m\*2;  }  cout<<ans<<endl;  } |

# 博弈

能够经过一步操作**存在**必败态的状态，就一定是**必胜态**。

经过一步操作**全是**必胜态的状态，一定是**必败态**。

*a*1⨁*a*2⨁...⨁*an*=1 为必胜态   
*a*1⨁*a*2⨁...⨁*an*=0 为必败态

MEX的运算对象为一个集合S，MEX(S)即最小的不属于S的非负整数。

对于一个给定的有向无环图，图上每个点表示一种局面或状态。   
定义关于图的每个顶点的SG函数如下

*SG*(*x*)=*MEX*{*SG*(*y*)|*x*→*y*}

当∃*y*,*SG*(*y*)=0,则*SG*(*x*)≠0   
当∀*y*,*SG*(*y*)≠0,则*SG*(*x*)=0

这个描述和上面的必胜必败态完全一致，可见当*SG*(*x*)=0 时，x即为一个必败态。

若有若干个平行局面共同组成了一个游戏，则这个父局面的状态由子局面的SG值决定

*SG*(*a*1)⨁*SG*(*a*2)⨁...⨁*SG*(*an*)=0为必败态

# 网络流

## R二分图最大匹配

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：二分图最大匹配  题目：POJ1274 The Perfect Stall    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<algorithm>  using namespace std;  const int maxn=200+10;  int n, m, ans, from[maxn];  int cur, fir[maxn], ver[maxn\*maxn], nxt[maxn\*maxn];  bool vis[maxn];  void Add(int u, int v)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur;  return ;  }  bool Dfs(int u)  {  for(int i=fir[u], v; i; i=nxt[i])  {  if(vis[v=ver[i]]) continue;  vis[v]=true;//保证y部每个结点不被重复访问  if((!from[v])||Dfs(from[v]))//找到新结点 或从已匹配结点出发原路返回能找到新结点 则原路径反向 长度一定增加  {  from[v]=u;//记录y部每个结点与x部结点配对情况  return true;  }  }  return false;  }  int main()  {  while(scanf("%d%d", &n, &m)!=EOF)  {  cur=0, ans=0;  memset(fir, 0, sizeof fir);  memset(from, 0, sizeof from);  for(int u=1, v, k; u<=n; u++)  {  scanf("%d", &k);  for(int i=1; i<=k; i++)  {  scanf("%d", &v);  Add(u, v);  }  }  for(int i=1; i<=n; i++)  {  memset(vis, 0, sizeof vis);  ans+=Dfs(i);//寻找增广路，保证每次增广成功时路径长度加一  }  cout<<ans<<endl;  }  return 0;  } |

## R二分图最佳匹配

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：二分图最佳匹配  题目：POJ2195 Going Home    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<algorithm>  using namespace std;  const int maxn=100+10;  const int INF=0x7f7f7f7f;  int n, m, ch, cm;  int lx[maxn\*maxn], ly[maxn], w[maxn\*maxn][maxn], from[maxn], del;  bool vx[maxn\*maxn], vy[maxn];  struct node  {  int x, y;  node(int a=0, int b=0){ x=a, y=b; }  }ph[maxn], pm[maxn\*maxn];  int Dis(int a, int b)  {  return abs(pm[a].x-ph[b].x)+abs(pm[a].y-ph[b].y);  }  bool Dfs(int x)  {  vx[x]=true;  for(int y=1; y<=ch; y++)  {  if(vy[y]) continue;  int t=lx[x]+ly[y]-w[x][y];//对于每一条匹配中的边(u, v)，满足lx[u]+ly[v]>=w[u][v]  if(!t)  {  vy[y]=true;  if((!from[y])||Dfs(from[y]))  {  from[y]=x;  return true;  }  }  else if(del>t) del=t;  }  return false;  }  int KM()  {  for(int i=1; i<=cm; i++)//x部  {  while(true)  {  memset(vx, 0, sizeof vx);  memset(vy, 0, sizeof vy);  del=INF;  if(Dfs(i)) break;  for(int j=1; j<=cm; j++) if(vx[j]) lx[j]-=del;  for(int j=1; j<=ch; j++) if(vy[j]) ly[j]+=del;//每一次修改lx[]、ly[]时  //端点均在/不在现有子图中的边不受影响  //x端在子图中的边，lx[u]'+ly[v]<lx[u]+ly[v]<w[u][v]，仍然不进入子图  //y端在子图中的边，lx[u]+ly[v]'>lx[u]+ly[v]，有可能进入子图  }  }  int ans=0;  for(int i=1; i<=ch; i++) if(from[i]) ans+=w[from[i]][i];  return ans;  }  int main()  {  int T=0;  while(scanf("%d%d", &n, &m)!=EOF&&n+m)  {  ch=cm=0;  memset(ly, 0, sizeof ly);  memset(from, 0, sizeof from);  char s[maxn];  for(int i=1; i<=n; i++)  {  scanf("%s", s+1);  for(int j=1; j<=m; j++)  {  if(s[j]=='H') ph[++ch]=node(i, j);  else if(s[j]=='m') pm[++cm]=node(i, j);  }  }  for(int i=1; i<=cm; i++)  {  lx[i]=-INF;  for(int j=1; j<=ch; j++)  {  w[i][j]=-Dis(i, j);  lx[i]=max(lx[i], w[i][j]);//lx[i]：i结点相连边的最大权值  }  }  cout<<-KM()<<endl;  }  return 0;  } |

## R最大流sap

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：最大流SAP  题目：HDU3549 Flow Problem    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdio>  #define Min(a, b) ((a)<(b)?(a):(b))  using namespace std;  const int maxn=15+5, maxm=1000+10, INF=0x7fffffff;  int T, n, m, st, des, ans, d[maxn], vd[maxn];  int cur, fir[maxn], ver[maxm<<1], nxt[maxm<<1], cap[maxm<<1];  void Add(int u, int v, int c)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cap[cur]=c;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cap[cur]=0;  return ;  }  void Init()  {  cur=1;  memset(d, 0, sizeof d);  memset(vd, 0, sizeof vd);  memset(fir, 0, sizeof fir);  return ;  }  void Read()  {  scanf("%d%d", &n, &m);  for(int i=1, u, v, c; i<=m; i++)  {  scanf("%d%d%d", &u, &v, &c);  Add(u, v, c);  }  st=1, des=n;  return ;  }  int Aug(int u, int flow)  {  if(u==des) return flow;  int rest=flow, mind=n-1, del;  for(int i=fir[u], v; i; i=nxt[i])  if(cap[i]>0)  {  if(d[u]==d[v=ver[i]]+1)  {  del=Aug(v, Min(rest, cap[i]));  cap[i]-=del, cap[i^1]+=del, rest-=del;  if(d[st]>=n) return flow-rest;  if(!rest) break;  }  mind=Min(mind, d[v]);  }  if(rest==flow)  {  if(!--vd[d[u]]) d[st]=n;  ++vd[d[u]=mind+1];  }  return flow-rest;  }  int Sap()  {  ans=0, vd[0]=n;  while(d[st]<n) ans+=Aug(st, INF);  return ans;  }  int main()  {  scanf("%d", &T);  for(int i=1; i<=T; i++)  {  Init(), Read();  printf("Case %d: %d\n", i, Sap());  }  return 0;  } |

## R最大流EK

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：最大流EK  题目：HDU3549 Flow Problem    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdio>  #include<queue>  #define Min(a, b) ((a)<(b)?(a):(b))  using namespace std;  const int maxn=15+5, maxm=1000+10, INF=0x7fffffff;  int T, n, m, st, des;  int cur, fir[maxn], ver[maxm<<1], nxt[maxm<<1], cap[maxm<<1];  int pre[maxn], rec[maxn], flow[maxn];  queue<int> q;  void Add(int u, int v, int c)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cap[cur]=c;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cap[cur]=0;  return ;  }  void Init()  {  cur=1;  memset(fir, 0, sizeof fir);  return ;  }  void Read()  {  scanf("%d%d", &n, &m);  for(int i=1, u, v, c; i<=m; i++)  {  scanf("%d%d%d", &u, &v, &c);  Add(u, v, c);  }  st=1, des=n;  return ;  }  int Bfs()  {  while(!q.empty()) q.pop();  memset(pre, -1, sizeof pre);  pre[st]=0, flow[st]=INF;  q.push(st);  while(!q.empty())  {  int u=q.front();  q.pop();  if(u==des) break;    for(int i=fir[u], v; i; i=nxt[i])  if(pre[v=ver[i]]==-1&&cap[i]>0)  {  pre[v]=u;  rec[v]=i;  flow[v]=Min(flow[u], cap[i]);  q.push(v);  }  }  return (pre[des]==-1)?(pre[des]):(flow[des]);  }  int Work()  {  int ans=0, del=0;  while((del=Bfs())!=-1)  {  for(int i=des; i!=st; i=pre[i]) cap[rec[i]]-=del, cap[rec[i]^1]+=del;  ans+=del;  }  return ans;  }  int main()  {  scanf("%d", &T);  for(int i=1; i<=T; i++)  {  Init(), Read();  printf("Case %d: %d\n", i, Work());  }  return 0;  } |

## R最大流Dinic

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：最大流Dinic  题目：HDU3549 Flow Problem    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdio>  #include<queue>  #define Min(a, b) ((a)<(b)?(a):(b))  using namespace std;  const int maxn=15+5, maxm=1000+10, INF=0x7fffffff;  int T, n, m, st, des, rec[maxn], dis[maxn];  int cur, fir[maxn], ver[maxm<<1], nxt[maxm<<1], cap[maxm<<1];  bool vis[maxn];  void Add(int u, int v, int c)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cap[cur]=c;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cap[cur]=0;  return ;  }  void Init()  {  cur=1;  memset(fir, 0, sizeof fir);  return ;  }  void Read()  {  scanf("%d%d", &n, &m);  for(int i=1, u, v, c; i<=m; i++)  {  scanf("%d%d%d", &u, &v, &c);  Add(u, v, c);  }  st=1, des=n;  return ;  }  bool Bfs()  {  queue<int> q;  memset(dis, 0, sizeof dis);  memset(vis, false, sizeof vis);  dis[st]=0, vis[st]=true;  q.push(st);  while(!q.empty())  {  int u=q.front();  q.pop();    for(int i=fir[u], v; i; i=nxt[i])  if(cap[i]>0&&!vis[v=ver[i]])  {  dis[v]=dis[u]+1, vis[v]=true;  q.push(v);  if(v==des) return true;  }  }  return false;  }  int Dfs(int u, int flow)  {  if(u==des||!flow) return flow;  int rest=flow, del;  for(int i=rec[u], v; i; i=nxt[i])  if(dis[u]+1==dis[v=ver[i]]&&(del=Dfs(v, Min(rest, cap[i])))>0)  {  cap[i]-=del, cap[i^1]+=del, rest-=del;  if(!rest) break;  }    return flow-rest;  }  int Dinic()  {  int ans=0;  while(Bfs())  {  memcpy(rec, fir, sizeof fir);  ans+=Dfs(st, INF);  }  return ans;  }  int main()  {  scanf("%d", &T);  for(int i=1; i<=T; i++)  {  Init(), Read();  printf("Case %d: %d\n", i, Dinic());  }  return 0;  } |

## R费用流Spfa

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：费用流Spfa  题目：POJ2195 Going Home    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<queue>  #include<algorithm>  using namespace std;  const int maxn=100+10, maxm=20000+10, INF=0x7f7f7f7f;  int n, m, S, T, dis[maxn<<1], pre[maxn<<1], fro[maxn<<1];  int h, hx[maxn], hy[maxn], pp, px[maxn], py[maxn];  int cur, fir[maxn<<1], ver[maxm<<1], nxt[maxm<<1], cap[maxm<<1], prc[maxm<<1];  bool vis[maxn<<1];  char s[maxn][maxn];  int Dis(int u, int v)  {  return abs(hx[u]-px[v])+abs(hy[u]-py[v]);  }  void Add(int u, int v, int c, int p)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cap[cur]=c, prc[cur]=p;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cap[cur]=0, prc[cur]=-p;  return ;  }  void Init()  {  h=pp=0, cur=1;  memset(fir, 0, sizeof fir);  memset(pre, 0, sizeof pre);  memset(fro, 0, sizeof fro);  return ;  }  void Read()  {  for(int i=1; i<=n; i++) scanf("%s", s[i]+1);  for(int i=1; i<=n; i++)  for(int j=1; j<=m; j++)  {  if(s[i][j]=='H') hx[++h]=i, hy[h]=j;  else if(s[i][j]=='m') px[++pp]=i, py[pp]=j;  }    S=0, T=h+pp+1;  for(int i=1; i<=h; i++) Add(S, i, 1, 0);  for(int i=1; i<=h; i++)  for(int j=1; j<=pp; j++)  Add(i, j+h, 1, Dis(i, j));  for(int i=1; i<=pp; i++) Add(i+h, T, 1, 0);  return ;  }  bool Spfa()  {  queue<int> q;  memset(vis, false, sizeof vis);  memset(dis, 0x7f, sizeof dis);  dis[S]=0;  q.push(S);  vis[S]=true;  while(!q.empty())  {  int u=q.front();  q.pop();  vis[u]=false;  for(int i=fir[u], v; i; i=nxt[i])  if(cap[i]>0&&dis[v=ver[i]]>dis[u]+prc[i])  {  dis[v]=dis[u]+prc[i];  pre[v]=u, fro[v]=i;  if(!vis[v])  {  vis[v]=true;  q.push(v);  }  }  }  return dis[T]!=INF;  }  void Work()  {  int ans=0, delta=0;  while(Spfa())  {  delta=INF;  for(int u=T; u!=S; u=pre[u]) delta=min(delta, cap[fro[u]]);  for(int u=T; u!=S; u=pre[u]) cap[fro[u]]-=delta, cap[fro[u]^1]+=delta;  ans+=dis[T]\*delta;  }  cout<<ans<<endl;  return ;  }  int main()  {  while(scanf("%d%d", &n, &m)!=EOF&&n+m) Init(), Read(), Work();  return 0;  } |

## R费用流Zkw

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  模版：费用流Zkw  题目：POJ2195 Going Home    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  #include<cstring>  #include<cstdlib>  #include<cstdio>  #include<cmath>  #include<queue>  #include<algorithm>  using namespace std;  const int maxn=100+10, maxm=20000+10, INF=0x7f7f7f7f;  int n, m, S, T, ans, dis[maxn<<1], pre[maxn<<1], fro[maxn<<1];  int h, hx[maxn], hy[maxn], pp, px[maxn], py[maxn];  int cur, fir[maxn<<1], ver[maxm<<1], nxt[maxm<<1], cap[maxm<<1], prc[maxm<<1];  bool vis[maxn<<1];  char s[maxn][maxn];  int Dis(int u, int v)  {  return abs(hx[u]-px[v])+abs(hy[u]-py[v]);  }  void Add(int u, int v, int c, int p)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur, cap[cur]=c, prc[cur]=p;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur, cap[cur]=0, prc[cur]=-p;  return ;  }  void Init()  {  ans=h=pp=0, cur=1;  memset(fir, 0, sizeof fir);  memset(pre, 0, sizeof pre);  memset(fro, 0, sizeof fro);  memset(dis, 0, sizeof dis);  return ;  }  void Read()  {  for(int i=1; i<=n; i++) scanf("%s", s[i]+1);  for(int i=1; i<=n; i++)  for(int j=1; j<=m; j++)  {  if(s[i][j]=='H') hx[++h]=i, hy[h]=j;  else if(s[i][j]=='m') px[++pp]=i, py[pp]=j;  }  S=0, T=h+pp+1;  for(int i=1; i<=h; i++) Add(S, i, 1, 0);  for(int i=1; i<=h; i++)  for(int j=1; j<=pp; j++)  Add(i, j+h, 1, Dis(i, j));  for(int i=1; i<=pp; i++) Add(i+h, T, 1, 0);  return ;  }  int Aug(int u, int flow)  {  if(u==T)  {  ans+=flow\*dis[S];  return flow;  }  vis[u]=true;  int rest=flow, delta=0;  for(int i=fir[u], v; i; i=nxt[i])  if(!vis[v=ver[i]]&&cap[i]>0&&dis[u]==dis[v]+prc[i])  {  delta=Aug(v, min(cap[i], rest));  cap[i]-=delta, cap[i^1]+=delta, rest-=delta;  if(!rest) break;  }  return flow-rest;  }  bool Update()  {  int delta=INF;  for(int u=S; u<=T; u++) if(vis[u])  for(int i=fir[u], v; i; i=nxt[i])  if(cap[i]>0&&!vis[v=ver[i]])  delta=min(delta, dis[v]-dis[u]+prc[i]);  if(delta==INF) return false;  for(int u=S; u<=T; u++) if(vis[u]) dis[u]+=delta;  return true;  }  void Zkw()  {  do  {  do{ memset(vis, false, sizeof vis); }while(Aug(S, INF));  }while(Update());  cout<<ans<<endl;  return ;  }  int main()  {  while(scanf("%d%d", &n, &m)!=EOF&&n+m) Init(), Read(), Zkw();  return 0;  } |

## T无源汇有上下界最大流

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cmath>  #include <algorithm>  #include <cstring>  using namespace std;  const int mn=22222;  const int mm=1000000;  const int oo=0x3fffffff;  int node, st, sd, edge, Edge;  int reach[mm], flow[mm], next[mm];  int head[mn], work[mn], dis[mn], que[mn];  int du[mm], ans[mm], id[mm], dn[mm];  inline void init(int \_node, int \_st, int \_sd)  {  node=\_node, st=\_st, sd=\_sd;  for(int i=0; i<node; i++)  head[i]=-1;  edge=0;  }  inline void addedge(int u, int v, int c1, int c2, int ID)  {  id[edge]=ID, reach[edge]=v, flow[edge]=c1, next[edge]=head[u],head[u]=edge++;  id[edge]=0, reach[edge]=u, flow[edge]=c2, next[edge]=head[v],head[v]=edge++;  }  bool bfs()  {  int u, v, l=0, h=0;  for(int i=0; i<node; i++) dis[i]=-1;  que[l++]=st;  dis[st]=0;  while(l!=h)  {  u=que[h++];  if(h==mn) h=0;  for(int i=head[u]; i>=0; i=next[i])  {  v=reach[i];  if(flow[i]&&dis[v]<0)  {  dis[v]=dis[u]+1;  que[l++]=v;  if(l==mn) l=0;  if(v==sd) return true;  }  }  }  return false;  }  int dfs(int u, int exp)  {  if(u==sd) return exp;  for(int &i=work[u]; i>=0; i=next[i])  {  int v=reach[i], tp;  if(flow[i]&&dis[v]==dis[u]+1&&(tp=dfs(v,min(flow[i],exp)))>0)  {  flow[i]-=tp;  flow[i^1]+=tp;  return tp;  }  }  return 0;  }  void Dinic()  {  while(bfs())  {  for(int i=0; i<node; i++) work[i]=head[i];  while(dfs(st,oo));  }  }  int main()  {  int n,m;  while(~scanf("%d%d",&n,&m))  {  init(n+2,0,n+1);  for(int i=1; i<=m; i++)  {  int u, v, down, up;  scanf("%d%d%d%d",&u,&v,&down,&up);  addedge(u,v,up-down,0,i);  du[u]-=down;  du[v]+=down;  dn[i]=down;  }  Edge=edge;  for(int i=1; i<=n; i++)  {  if(du[i]>0) addedge(st,i,du[i],0,0);  if(du[i]<0) addedge(i,sd,-du[i],0,0);  }  Dinic();  bool flag=true;  for(int i=head[st]; i>=0; i=next[i])  if(flow[i]>0)  {  flag=false;  break;  }  if(!flag) puts("NO");  else  {  puts("YES");  for(int i=0; i<Edge; i++) ans[id[i]]=flow[i^1];  for(int i=1; i<=m; i++)  printf("%d\n",ans[i]+dn[i]);  }  }  return 0;  } |

## T有源汇有上下界最大流

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cstring>  #include <queue>  #include <algorithm>  using namespace std;  const int mn=2222;  const int mm=1000000;  const int oo=100000000;  int node, st, sd, edge;  int reach[mm], flow[mm], next[mm];  int head[mn], work[mn], dis[mn], que[mn];  int du[mn], dn[444][1111], id[444][1111];  inline void init(int \_node, int \_st, int \_sd)  {  node=\_node, st=\_st, sd=\_sd;  for(int i=0; i<node; i++)  head[i]=-1, du[i]=0;  edge=0;  }  inline void addedge(int u, int v, int c1, int c2)  {  reach[edge]=v, flow[edge]=c1, next[edge]=head[u],head[u]=edge++;  reach[edge]=u, flow[edge]=c2, next[edge]=head[v],head[v]=edge++;  }  bool bfs()  {  int u, v, l=0, h=0;  for(int i=0; i<node; i++) dis[i]=-1;  que[l++]=st;  dis[st]=0;  while(l!=h)  {  u=que[h++];  if(h==mn) h=0;  for(int i=head[u]; i>=0; i=next[i])  {  v=reach[i];  if(flow[i]&&dis[v]<0)  {  dis[v]=dis[u]+1;  que[l++]=v;  if(l==mn) l=0;  if(v==sd) return true;  }  }  }  return false;  }  int dfs(int u, int exp)  {  if(u==sd) return exp;  for(int &i=work[u]; i>=0; i=next[i])  {  int v=reach[i], tp;  if(flow[i]&&dis[v]==dis[u]+1&&(tp=dfs(v,min(flow[i],exp)))>0)  {  flow[i]-=tp;  flow[i^1]+=tp;  return tp;  }  }  return 0;  }  int Dinic()  {  int max\_flow=0, flow;  while(bfs())  {  for(int i=0; i<node; i++) work[i]=head[i];  while(flow=dfs(st,oo)) max\_flow+=flow;  }  return max\_flow;  }  int main()  {  int n, m, g, l, r, c, d;  while(scanf("%d%d",&n,&m)!=EOF)  {  init(n+m+2,n+m,n+m+1);  for(int i=0; i<m; i++)  {  scanf("%d",&g);  du[sd]+=g;  du[i+n]-=g;  addedge(i+n,sd,oo-g,0);  }  memset(id,0,sizeof(id));  for(int i=0; i<n; i++)  {  scanf("%d%d",&c,&d);  addedge(st,i,d,0);  for(int j=0; j<c; j++)  {  scanf("%d%d%d",&g,&l,&r);  du[i]-=l;  du[g+n]+=l;  dn[i][g]=l;  addedge(i,g+n,r-l,0);  id[i][g]=edge-1;  }  }  addedge(sd,st,oo,0);  st=node, sd=node+1, node+=2;  head[st]=head[sd]=-1; ///!!  int sum=0;  for(int i=0; i<node-2; i++)  {  if(du[i]>0) sum+=du[i], addedge(st,i,du[i],0);  else addedge(i,sd,-du[i],0);  }  int maxflow=Dinic();  if(maxflow!=sum) puts("-1");  else  {  head[st]=head[sd]=-1, node-=2;  st=node-2, sd=node-1;  maxflow=Dinic();  printf("%d\n",maxflow);  for(int i=0; i<n; i++)  {  for(int j=0; j<m; j++)  if(id[i][j])  printf("%d\n",flow[id[i][j]]+dn[i][j]);  }  }  puts("");  }  return 0;  } |

## T有源汇有上下节最小流

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cmath>  #include <algorithm>  #include <cstring>  using namespace std;  const int mn=111;  const int mm=11111;  const int oo=0x3fffffff;  int node, st, sd, edge, St, Sd, Edge;  int reach[mm], flow[mm], next[mm];  int head[mn], work[mn], dis[mn], que[mn];  int du[mm], ans[mm], id[mm], dn[mm];  inline void init(int \_node, int \_st, int \_sd)  {  node=\_node, st=\_st, sd=\_sd;  for(int i=0; i<node; i++)  head[i]=-1, du[i]=0;  edge=0;  }  inline void addedge(int u, int v, int c1, int c2, int ID)  {  id[edge]=ID, reach[edge]=v, flow[edge]=c1, next[edge]=head[u],head[u]=edge++;  id[edge]=0, reach[edge]=u, flow[edge]=c2, next[edge]=head[v],head[v]=edge++;  }  bool bfs()  {  int u, v, l=0, h=0;  for(int i=0; i<node; i++) dis[i]=-1;  que[l++]=st;  dis[st]=0;  while(l!=h)  {  u=que[h++];  if(h==mn) h=0;  for(int i=head[u]; i>=0; i=next[i])  {  v=reach[i];  if(flow[i]&&dis[v]<0)  {  dis[v]=dis[u]+1;  que[l++]=v;  if(l==mn) l=0;  if(v==sd) return true;  }  }  }  return false;  }  int dfs(int u, int exp)  {  if(u==sd) return exp;  for(int &i=work[u]; i>=0; i=next[i])  {  int v=reach[i], tp;  if(flow[i]&&dis[v]==dis[u]+1&&(tp=dfs(v,min(flow[i],exp)))>0)  {  flow[i]-=tp;  flow[i^1]+=tp;  return tp;  }  }  return 0;  }  void Dinic()  {  int max\_flow=0, flow;  while(bfs())  {  for(int i=0; i<node; i++) work[i]=head[i];  while(flow=dfs(st,oo)) max\_flow+=flow;  }  }  int main()  {  int n, m;  while(~scanf("%d%d",&n,&m))  {  init(n+1,1,n);  for(int i=1; i<=m; i++)  {  int u, v, c, k;  scanf("%d%d%d%d",&u,&v,&c,&k);  if(k) du[u]-=c, du[v]+=c, ans[i]=c;  else addedge(u,v,c,0,i);  }  St=st, Sd=sd, Edge=edge;  st=node, sd=node+1, node+=2; ///增设超级源点和超级汇点，因为网络中规定不能有弧指向st，也不能有流量流出sd  head[st]=head[sd]=-1;  for(int i=1; i<=n; i++)  {  if(du[i]>0) addedge(st,i,du[i],0,0);  if(du[i]<0) addedge(i,sd,-du[i],0,0);  }  Dinic();  addedge(Sd,St,oo,0,0);  Dinic();  bool flag=true;  for(int i=head[st]; i>=0; i=next[i])  if(flow[i]>0) ///当且仅当附加弧达到满负载有可行流  {  flag=false;  break;  }  if(!flag)  puts("Impossible");  else  {  int res=0, i;  for(i=head[Sd]; i>=0; i=next[i])  if(reach[i]==St) break;  res=flow[i^1];  printf("%d\n",res);  for(i=0; i<Edge; i++) ans[id[i]]=flow[i^1];  for(i=1; i<=m; i++)  {  if(i!=m) printf("%d ",ans[i]);  else printf("%d\n",ans[i]);  }  }  }  return 0;  } |

# 其他

## D莫队

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  #define N 51000  #define K 1000100  int n, m, k;  int a[N], b[N], pos[N], c[2][K], ans[N], cnt;  struct node {  int l, r, id;  void sc(int i){  scanf("%d%d", &l, &r);  id = i;  }  }p[N];  bool cmp(node a, node b) {  if(pos[a.l] == pos[b.l]) return a.r < b.r;  return pos[a.l] < pos[b.l];  }  void update(int x, int y) {  if(a[x] != b[x]) cnt -= min(c[0][a[x]], c[1][a[x]]) + min(c[0][b[x]], c[1][b[x]]);  else cnt -= min(c[0][a[x]], c[1][a[x]]);  c[0][a[x]] += y;  c[1][b[x]] += y;  if(a[x] != b[x]) cnt += min(c[0][a[x]], c[1][a[x]]) + min(c[0][b[x]], c[1][b[x]]);  else cnt += min(c[0][a[x]], c[1][a[x]]);  }  void solve() {  memset(c, 0, sizeof(c));  int pl = 0, pr = 0;  cnt = a[0] == b[0] ? 1 : 0;  c[0][a[0]]++; c[1][b[0]]++;  for(int i = 0;i < m;i++) {  int id = p[i].id, l = p[i].l, r = p[i].r;  for(int j = pr + 1;j <= r;j++)  update(j, 1);  for(int j = pr;j > r;j--)  update(j, -1);  for(int j = pl;j < l;j++)  update(j, -1);  for(int j = pl-1; j >= l;j--)  update(j, 1);  pr = r; pl = l;  ans[id] = cnt;  }  for(int i = 0;i < m;i++)  printf("%d\n", ans[i]);  }  int main() {  while(~scanf("%d%d%d", &n, &m, &k)) {  for(int i = 0;i < n;i++)  scanf("%d", &a[i]);  for(int i = 0;i < n;i++)  scanf("%d", &b[i]);  int bk = sqrt(n + 1.0);  for(int i = 0;i < n;i++)  pos[i] = i / bk;  for(int i = 0;i < m;i++)  p[i].sc(i);  sort(p, p+m, cmp);  solve();  }  return 0;  } |

## R2-SAT

|  |
| --- |
| void Init()  {  cur=idx=tot=0;  memset(w, 0, sizeof w);  memset(fir, 0, sizeof fir);  memset(dfn, 0, sizeof dfn);  memset(low, 0, sizeof low);  memset(ins, false, sizeof ins);  while(!stk.empty()) stk.pop();  return ;  }  void Add(int u, int v)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur;  return ;  }  void Read()  {  n<<=1;  char s[10];  for(int i=1, u, v, t; i<=m; i++)  {  scanf("%d%d%d%s", &u, &v, &t, s);  u<<=1; v<<=1;  switch(s[0])  {  case 'A': if(t) Add(u^1, u), Add(v^1, v);  else Add(u, v^1), Add(v, u^1);  break;  case 'O': if(t) Add(u^1, v), Add(v^1, u);  else Add(u, u^1), Add(v, v^1);  break;  case 'X': if(t) Add(u, v^1), Add(v, u^1), Add(u^1, v), Add(v^1, u);  else Add(u, v), Add(v, u), Add(u^1, v^1), Add(v^1, u^1);  break;  }  }  return ;  }  void Tarjan(int u)  {  dfn[u]=low[u]=++idx;  ins[u]=true;  stk.push(u);  for(int i=fir[u], v; i; i=nxt[i])  {  if(!dfn[v=ver[i]])  {  Tarjan(v);  low[u]=min(low[u], low[v]);  }  else if(ins[v]) low[u]=min(low[u], low[v]);  }  if(dfn[u]==low[u])  {  int v=stk.top();  stk.pop();  w[v]=++tot, ins[v]=false;  while(v!=u)  {  v=stk.top();  stk.pop();  w[v]=tot, ins[v]=false;  }  }  return ;  }  bool Ok()  {  for(int i=0; i<n; i+=2) if(w[i]==w[i^1]) return false;  return true;  }  void Work()  {  for(int i=0; i<n; i++) if(!dfn[i]) Tarjan(i);  if(Ok()) printf("YES\n");  else printf("NO\n");  return ;  } |

## R高精度

|  |
| --- |
| struct node  {  lint s[MAXN], lim;  node(){ memset(s, 0, sizeof s); s[0]=1LL; lim=10000LL; } //lim压4位 1-s[0]低到高位  bool operator <(const node&t) const  {  if(s[0]!=t.s[0]) return s[0]<t.s[0];  for(int i=s[0]; i>=1; i--) if(s[i]!=t.s[i]) return s[i]<t.s[i];  return false;  }  bool operator >(const node&t) const  {  if(s[0]!=t.s[0]) return s[0]>t.s[0];  for(int i=s[0]; i>=1; i--) if(s[i]!=t.s[i]) return s[i]>t.s[i];  return false;  }  bool operator ==(const node&t) const  {  if(s[0]!=t.s[0]) return false;  for(int i=1; i<=s[0]; i++) if(s[i]!=t.s[i]) return false;  return true;  }  void operator +=(lint t) // 高精度数加单数 末位相加再进位  {  s[1]+=t;  for(int i=1; i<=s[0]; i++)  {  s[i+1]+=(s[i]/lim), s[i]%=lim;  if(i==s[0]&&s[i+1]) ++s[0];  }  return ;  }  node operator +(lint t)  {  node tp=\*this; tp+=t;  return tp;  }  void operator +=(node t) // 高精度数加高精度数 逐位相加再进位  {  s[0]=max(s[0], t.s[0]);  for(int i=1; i<=s[0]; i++) s[i]+=t.s[i];  for(int i=1; i<=s[0]; i++)  {  s[i+1]+=(s[i]/lim), s[i]%=lim;  if(i==s[0]&&s[i+1]) ++s[0];  }  return ;  }  node operator +(node t)  {  node tp=\*this; tp+=t;  return tp;  }  void operator -=(lint t) // 高精度数减单数 末位相减再借位 判断是否降位  {  s[1]-=t;  for(int i=1; s[i]<0&&i<s[0]; i++) if(s[i]<0) s[i]+=lim, s[i+1]-=1;  for(; !s[s[0]]&&s[0]>1; --s[0]); // 高位为0 降位  return ;  }  node operator -(lint t)  {  node tp=\*this; tp-=t;  return tp;  }  void operator -=(node t) // 高精度数减高精度数 逐位相减 同时借位  {  for(int i=1; i<=s[0]; i++)  {  if(s[i]<t.s[i]) s[i]+=lim, s[i+1]-=1;  s[i]-=t.s[i];  }  for(; !s[s[0]]&&s[0]>1; --s[0]);  return ;  }  node operator -(node t)  {  node tp=\*this; tp-=t;  return tp;  }  void operator \*=(lint t) // 高精度数乘单数 逐位相乘再进位  {  for(int i=1; i<=s[0]; i++) s[i]\*=t;  for(int i=1; i<=s[0]; i++)  {  s[i+1]+=(s[i]/lim), s[i]%=lim;  if(i==s[0]&&s[i+1]) ++s[0];  }  return ;  }  node operator \*(lint t)  {  node tp=\*this; tp\*=t;  return tp;  }  void operator \*=(node t) // 高精度数乘高精度数 逐位相乘再进位  {  node tp;  tp.s[0]=s[0]+t.s[0]-1;  for(int i=1; i<=s[0]; i++)  for(int j=1; j<=t.s[0]; j++)  tp.s[i+j-1]+=s[i]\*t.s[j]; // a[i]与b[j]相乘得到c[i+j-1]  for(int i=1; i<=tp.s[0]; i++)  {  tp.s[i+1]+=(tp.s[i]/lim), tp.s[i]%=lim;  if(i==tp.s[0]&&tp.s[i+1]) ++tp.s[0];  }  \*this=tp;  return ;  }  node operator \*(node t)  {  node tp=\*this; tp\*=t;  return tp;  }  void operator /=(lint t) // 高精度数除以单数 逐位相除 同时借位 判断是否降位  {  for(int i=s[0]; i>1; i--) s[i-1]+=(s[i]%t\*lim), s[i]/=t;  for(s[1]/=t; s[0]>1&&!s[s[0]]; --s[0]);  return ;  }  node operator /(lint t)  {  node tp=\*this; tp/=t;  return tp;  }  void operator /=(node t) // 高精度数除以高精度数 二分答案  {  node L, M, R=\*this;  while(L<R)  {  M=(L+R+1)/2;  if((t\*M)>\*this) R=M-1;  else L=M;  }  \*this=L;  return ;  }  node operator /(node t)  {  node tp=\*this; tp/=t;  return tp;  }  void read(string st="")  {  cin>>st; s[0]=st.size();  for(int i=0, j; j=(s[0]-i-1)/4+1, i<s[0]; i++) s[j]=s[j]\*10+st[i]-'0';  (s[0]%4)?(s[0]=s[0]/4+1):(s[0]/=4);  return ;  }  void put()  {  printf("%lld", s[s[0]]);  for(int i=s[0]-1; i; i--) printf("%04lld", s[i]);  return ;  }  }A, B, C, D;  node Gcd(node a,node b)  {  if(a.s[0]==1&&a.s[1]==0) return b;  if(b.s[0]==1&&b.s[1]==0) return a;  if((a.s[1]&1)&&(b.s[1]&1)) return (a>b) ? Gcd(a-b,b) : Gcd(b-a,a);  if(a.s[1]&1) return Gcd(a,b/2);  if(b.s[1]&1) return Gcd(a/2,b);  return Gcd(a/2,b/2)\*2;  } |

## R莫队分块

|  |
| --- |
| int n, m, unit, col[maxn], c[maxn];  struct node  {  int l, r, id;  lint ans;  bool operator <(const node&t)const  {  int al=l/unit, bl=t.l/unit;  return al==bl ? r<t.r : al<bl;  }  }p[maxn];  bool Cmp(node a, node b)  {  return a.id<b.id;  }  lint Gcd(lint a, lint b)  {  return b==0ll ? a : Gcd(b, a%b);  }  void Read()  {  for(int i=1; i<=n; i++) scanf("%d", &col[i]);  for(int i=1; i<=m; i++)  {  scanf("%d%d", &p[i].l, &p[i].r);  p[i].id=i;  }  unit=(int)sqrt(n);  sort(p+1, p+1+m);  return ;  }  void Work()  {  lint tmp=0;  memset(c, 0, sizeof c);  int l=1, r=0;  for(int i=1; i<=m; i++)  {  while(r<p[i].r)  {  r++;  tmp+=2ll\*(c[col[r]]++);  }  while(r>p[i].r)  {  tmp-=2ll\*(--c[col[r]]);  r--;  }  while(l<p[i].l)  {  tmp-=2ll\*(--c[col[l]]);  l++;  }  while(l>p[i].l)  {  l--;  tmp+=2ll\*(c[col[l]]++);  }  p[i].ans=tmp;  }  sort(p+1, p+1+m, Cmp);  for(int i=1; i<=m; i++)  {  if(p[i].ans==0ll || p[i].l==p[i].r) printf("0/1\n");  else  {  lint len=(p[i].r-p[i].l+1ll)\*(p[i].r-p[i].l);  lint tmp=Gcd(p[i].ans, len);  printf("%lld/%lld\n", p[i].ans/tmp, len/tmp);  }  }  return ;  } |

## R莫队曼哈顿

|  |
| --- |
| int n, m, col[maxn], tot, fa[maxn], c[maxn];  int val[maxn], pos[maxn];  int cur, fir[maxn], ver[maxn<<1], nxt[maxn<<1];  bool vis[maxn];  struct node  {  int l, r, id;  lint ans;  bool operator <(const node&t)const  {  return l==t.l ? r<t.r : l<t.l;  }  }p[maxn];  struct line  {  int u, v, d;  line(int \_u=0, int \_v=0, int \_d=0)  {  u=\_u, v=\_v, d=\_d;  }  bool operator <(const line&t)const  {  return d<t.d;  }  }s[maxn];  bool Cmp(node a, node b)  {  return a.id<b.id;  }  lint Gcd(lint a, lint b)  {  return b==0ll ? a : Gcd(b, a%b);  }  int Dist(node u, node v)  {  return abs(u.l-v.l)+abs(u.r-v.r);  }  void Line(int u, int v, int d)  {  s[++tot]=line(u, v, d);  return ;  }  int Rt(int x)  {  return x==fa[x] ? x : fa[x]=Rt(fa[x]);  }  void Add(int u, int v)  {  ver[++cur]=v, nxt[cur]=fir[u], fir[u]=cur;  ver[++cur]=u, nxt[cur]=fir[v], fir[v]=cur;  return ;  }  void Update(int x, int \_val, int \_pos)  {  for(int i=x; i; i-=lowbit(i)) if(\_val<val[i])  {  val[i]=\_val;  pos[i]=\_pos;  }  return ;  }  int Query(int x, int M, int \_val=INF, int \_pos=0)  {  for(int i=x; i<=M; i+=lowbit(i)) if(val[i]<\_val)  {  \_val=val[i];  \_pos=pos[i];  }  return \_pos;  }  void Manhattan()  {  tot=0;  int a[maxn], b[maxn];  for(int d=0; d<4; d++)  {  if(d==1 || d==3)  {  for(int i=1; i<=m; i++) swap(p[i].l, p[i].r);  }  else if(d==2)  {  for(int i=1; i<=m; i++) p[i].l=-p[i].l;  }  sort(p+1, p+1+m);  for(int i=1; i<=m; i++) a[i]=b[i]=p[i].r-p[i].l;  sort(b+1, b+1+m);  int M=unique(b+1, b+1+m)-(b+1);  for(int i=1; i<=M; i++) val[i]=INF, pos[i]=0;  for(int i=m; i; i--)  {  a[i]=lower\_bound(b+1, b+1+M, a[i])-b;  int ret=Query(a[i], M);  if(ret) Line(p[i].id, p[ret].id, Dist(p[i], p[ret]));  Update(a[i], p[i].l+p[i].r, i);  }  }  cur=0;  sort(s+1, s+1+tot);  for(int i=1; i<=m; i++) fa[i]=i;  int cnt=1, i=0, u, v, d;  while(cnt++<m)  {  while(++i<=tot)  {  u=Rt(s[i].u), v=Rt(s[i].v), d=s[i].d;  if(u!=v) break;  }  (u<v) ? fa[v]=u : fa[u]=v;  Add(s[i].u, s[i].v);  }  return ;  }  void Del(int l, int r, lint &tmp)  {  for(int i=l; i<=r; i++) tmp-=2ll\*(--c[col[i]]);  return ;  }  void Ins(int l, int r, lint &tmp)  {  for(int i=l; i<=r; i++) tmp+=2ll\*(c[col[i]]++);  return ;  }  void Dfs(int u, lint tmp)  {  vis[u]=true;  for(int i=fir[u], v; i; i=nxt[i])  {  if(vis[v=ver[i]]) continue;  if(p[u].l<p[v].l) Del(p[u].l, p[v].l-1, tmp);  if(p[u].l>p[v].l) Ins(p[v].l, p[u].l-1, tmp);  if(p[u].r>p[v].r) Del(p[v].r+1, p[u].r, tmp);  if(p[u].r<p[v].r) Ins(p[u].r+1, p[v].r, tmp);  p[v].ans=tmp;  Dfs(v, tmp);  if(p[u].l<p[v].l) Ins(p[u].l, p[v].l-1, tmp);  if(p[u].l>p[v].l) Del(p[v].l, p[u].l-1, tmp);  if(p[u].r>p[v].r) Ins(p[v].r+1, p[u].r, tmp);  if(p[u].r<p[v].r) Del(p[u].r+1, p[v].r, tmp);  }  return ;  }  void Work()  {  sort(p+1, p+1+m, Cmp);  memset(vis, false, sizeof vis);  memset(c, 0, sizeof c);  for(int i=1; i<=m; i++) p[i].r=-p[i].r;  p[1].ans=0ll;  Ins(p[1].l, p[1].r, p[1].ans);  Dfs(1, p[1].ans);  for(int i=1; i<=m; i++)  {  if(p[i].ans==0ll || p[i].l==p[i].r) printf("0/1\n");  else  {  lint len=(p[i].r-p[i].l+1ll)\*(p[i].r-p[i].l);  lint tmp=Gcd(p[i].ans, len);  printf("%lld/%lld\n", p[i].ans/tmp, len/tmp);  }  }  return ;  } |

## Java大数

|  |
| --- |
| import java.util.\*;  import java.io.\*;  import java.math.\*;  public class Main {  public static long long extend\_Euclid(long long a, long long b, long long &x, long long &y)  {  if(b==0)  {  x = 1;  y = 0;  return a;  }  long long r = extend\_Euclid(b, a%b, y, x);  y -= a/b\*x;  return r;  }  public static long long anti(long long b, long long mod, long long x, long long y)  {  long long r = extend\_Euclid(b, mod, x, y);  return (x%mod+mod)%mod;  }  public static BigInteger p = new BigInteger("33232930569601");  public static BigInteger two = new BigInteger("2");  public static void main(String[] args) {  Scanner cin = new Scanner(new BufferedInputStream(System.in));  while(cin.hasNext())  {  BigInteger n = cin.nextBigInteger();  BigInteger m = cin.nextBigInteger();  long long s = m.longValue();  long long q = p.longValue();  BigInteger ans = new BigInteger("1");  if(n.divide(two).compareTo(m) < 0)  m = m.subtract(n.divide(two));  for(int i = 0;i < m;i++)  {  BigInteger t = BigInteger.valueOf(i);  ans = ans.multiply(n.subtract(t)).mod(p);  long long a = anti(s-i, q, x, y);  t = BigInteger.valueOf(a);  ans = ans.multiply(t).mod(p);  }  System.println(ans);  }  }  } |

## STL

|  |
| --- |
| typedef map <int, LL> MAP;  typedef pair<int, int> PR;  typedef vector<PR> DV;  DV dv, tp;  MAP mp;  void makeTable{  dv.clear();  for(int i = 0;i < n;i++){  int g = -1;  tp.clear();  dv.push\_back(PR(a[i], i));  for(DV::iterator it = dv.begin();it != dv.end();it++){  if(\_\_gcd(it->first, a[i]) != g){  g = \_\_gcd(it->first, a[i]);  tp.push\_back(PR(g, it->second));  }  }  tp.push\_back(PR(0, i+1));  for(DV::iterator it = tp.begin();it+1 != tp.end();it++)  mp[it->first] += (it+1)->second-it->second;  swap(dv, tp);  }  }  //sort默认按照first、second为第一、二关键字排序输出。如果加个greater<type>()则逆序输出。  bool cmp(const pair<int, string> &lhs, const pair<int, string> &rhs)  {  if(rhs.first == lhs.first) return lhs.second < rhs.second;  return lhs.first > rhs.first;  }  vector< pair <int, string> >ans;  sort(ans.begin(), ans.end(), cmp);  sort(ans.begin(), ans.end(), greater<pair <int, string> >());  //定义优先队列优先级  priority\_queue <node> q;  bool operator < (node a, node b){  return a.v > b.v;  }  struct cmp  {  bool operator ()(const int&a, const int&b)const  { return dis[a]<dis[b];//小根堆 }  }  priority\_queue<int, vector<int>, cmp>; |