

AI模板

Baileys Thea Starry



2016-10-16

NJUST

目录

[(一) 图论 2](#_Toc464397884)

[1. R最短路Dijkastra 2](#_Toc464397885)

[2. R最短路Floyd 2](#_Toc464397886)

[3. R最短路SPFA 2](#_Toc464397887)

[4. R最小生成树Kruskal 3](#_Toc464397888)

[5. R最小生成树Prim 4](#_Toc464397889)

[6. R强连通分量Tarjan 4](#_Toc464397890)

[7. D连通分量Tarjan 6](#_Toc464397891)

[8. D连通分量korasaju 7](#_Toc464397892)

[9. R曼哈顿生成树 7](#_Toc464397893)

[10. R欧拉路径输出 10](#_Toc464397894)

[(二) 数据结构 13](#_Toc464397895)

[1. R堆 13](#_Toc464397896)

[2. R平衡树avl 13](#_Toc464397897)

[3. R平衡树sbt 15](#_Toc464397898)

[4. R树状数组二维 17](#_Toc464397899)

[5. R主席树 17](#_Toc464397900)

[6. R平衡树 18](#_Toc464397901)

[7. R树链剖分 19](#_Toc464397902)

[(三) DP 23](#_Toc464397903)

[1. 背包多重 23](#_Toc464397904)

[2. 背包完全 24](#_Toc464397905)

[3. 背包混合 24](#_Toc464397906)

[4. 轮廓线DP 25](#_Toc464397907)

[(四) 字符串 29](#_Toc464397908)

[(五) 计算几何 29](#_Toc464397909)

[(六) 数学 29](#_Toc464397910)

[(七) 博弈 29](#_Toc464397911)

[(八) 其他 29](#_Toc464397912)

[1. D莫队 29](#_Toc464397913)

[2. R2-SAT 31](#_Toc464397914)

[3. R高精度 33](#_Toc464397915)

[4. R莫队分块 36](#_Toc464397916)

[5. R莫队曼哈顿 38](#_Toc464397917)

# 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

## 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;  } |

## 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

# 计算几何

# 数学

# 博弈

# 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 ;  } |