

目次			
1	環境	1	
1.1	.Xmodmap	1	
1.2	.vimrc	1	
1.3	Makefile	1	
2	テンプレート	1	
2.1	C++	1	
2.2	Java	1	
3	グラフ	2	
3.1	定義	2	
3.2	定理	2	
3.3	基本要素	2	
3.4	単一始点最短路 (Dijkstra)	2	
3.5	単一始点最短路 (Bellman-Ford)	2	
3.6	全点間最短路 (Warshall-Floyd)	3	
3.7	最小全域木 (Prim)	3	
3.8	最小全域森 (Kruskal)	3	
3.9	最小シュタイナー木 (Dreyfus-Wagner)	3	
3.10	最大独立集合	4	
3.11	トポロジカルソート	4	
3.12	強連結成分分解	4	
3.13	橋	4	
3.14	関節点	5	
3.15	2-SAT	5	
3.16	最小共通先祖	5	
3.17	二部性判定	5	
4	フロー	6	
4.1	最大流 (Dinic)	6	
4.2	最小費用流	6	
4.3	2部グラフの最大マッチング (naïve)	7	
4.4	2部グラフの最大マッチング (Hopcroft-Karp)	7	
4.5	DAG の最小独立パス被覆	8	
4.6	無向グラフの全域最小カット (Nagamochi-Ibaraki)	8	
5	文字列	8	
5.1	Knuth-Morris-Pratt	8	
5.2	最長回文 (Manacher)	8	
5.3	Suffix Array (Kärkkäinen-Sanders)	9	
5.4	Suffix Array (SA-IS)	10	
6	動的計画法	11	
6.1	最長共通部分列	11	
6.2	最長増加部分列	11	
7	データ構造	11	
7.1	Union-Find Tree	11	
7.2	Fenwick Tree	11	
7.3	Segment Tree	12	
7.4	Range Tree	13	
8	数学	14	
8.1	数学定数	14	
8.2	ユークリッドの互除法	14	
8.3	逆元	14	
8.4	エラトステネスの篩	14	
8.5	素数判定 (Miller-Rabin)	14	
8.6	Gauss-Jordan の消去法	15	
9	幾何	15	
9.1	基本要素	15	
9.2	回転方向	16	
9.3	面積	16	
9.4	交差判定	16	
9.5	距離	16	
9.6	円の接線	17	
9.7	凸包	17	
9.8	凸多角形の包含判定	17	
9.9	凸多角形の切断	17	
9.10	線分アレンジメント	17	
10	その他	18	
10.1	ビット演算	18	
10.2	ハッシュ関数	19	
10.3	Fairfield の公式	19	
10.4	さいころ	19	

1 環境

1.1 .Xmodmap

[変換] と [半角/全角], [無変換] と [Escape] を入れ替え , [CapsLock] を [Ctrl] に割り当てる .

```
1 keysym Muhenkan = Escape
2 keysym Henkan_Mode = Zenkaku_Hankaku
3 keysym Escape = Muhenkan
4 keysym Zenkaku_Hankaku = Henkan_Mode
5
6 remove Lock = Caps_Lock
7 keysym Caps_Lock = Control_L
8 add Control = Control_L
```

1.2 .vimrc

不要そうなものは適当に削る .

```
1 set number
2 set tabstop=4 shiftwidth=4
3 set textwidth=0
4 set nobackup noswapfile
5
6 set hlsearch noincsearch ignorecase
7
8 set cindent
9 set cinoptions=L0,:0,l1,g0,cs,C1,(s,m1
10 set cinkeys-=0#
11 set noshowmatch
12
13 set laststatus=2
14 set tabpagemax=32
15
16 set list listchars=tab:^\ ,trail:_
17
18 noremap <Up> <Nop>
19 noremap <Down> <Nop>
20 noremap <Left> :<C-u>tabprev<Return>
21 noremap <Right> :<C-u>tabnext<Return>
22 inoremap <Return> <Return>x<BS>
23 nnoremap o ox<BS>
24 nnoremap O Ox<BS>
25
26 filetype plugin indent on
27 syntax enable
28
29 augroup misc
30   autocmd filetype * setlocal comments=
31 augroup END
```

1.3 Makefile

```
1 MAIN=
2
3 CXXFLAGS=-Wall -Wextra -Wno-sign-compare -Wno-unused-result -std=c++11 -g -D_GLIBCXX_DEBUG
4 #CXXFLAGS=-Wall -Wextra -Wno-sign-compare -Wno-unused-result -std=c++11 -O2
5
6 all: $(MAIN)
7 clean:
8     find . -maxdepth 1 -type f -perm -111 -delete
9 run: $(MAIN)
10     ./< <$(MAIN).in
```

2 テンプレート

2.1 C++

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define dump(n) cout<<"# "<<n<<'='<<(n)<<endl
5 #define repi(i,a,b) for(int i=int(a);i<int(b);i++)
6 #define peri(i,a,b) for(int i=int(b);i-->int(a);)
7 #define rep(i,n) repi(i,0,n)
8 #define per(i,n) peri(i,0,n)
9 #define all(c) begin(c),end(c)
10 #define mp make_pair
11 #define mt make_tuple
12
13 typedef unsigned int uint;
14 typedef long long ll;
15 typedef unsigned long long ull;
16 typedef pair<int,int> pii;
17 typedef vector<int> vi;
18 typedef vector<vi> vvi;
19 typedef vector<ll> vl;
20 typedef vector<vl> vvl;
21 typedef vector<double> vd;
22 typedef vector<vd> vvd;
23 typedef vector<string> vs;
24
25 const int INF=1e9;
26 const int MOD=1e9+7;
27 const double EPS=1e-9;
28
29 template<typename T1,typename T2>
30 ostream& operator<<(ostream& os,const pair<T1,T2>& p){
31     return os<<'('<<p.first<<','<<p.second<<')';
32 }
33 template<typename T>
34 ostream& operator<<(ostream& os,const vector<T>& a){
35     os<<'[';
36     rep(i,a.size()) os<<(i?" ":"")<<a[i];
37     return os<<']';
38 }
39
40 int main()
41 {
42 }
```

2.2 Java

Solver#main に書く .

```
1 import java.io.*;
2 import java.lang.*;
3 import java.math.*;
4 import java.util.*;
5
6 class Main{public static void main(String[] args){new Solver();}}
7
8 class Solver{
9     Scanner reader=new Scanner(System.in);
10    PrintWriter writer=new PrintWriter(System.out);
11    Solver(){main(); writer.flush();}
12
13    void main(){
```

```

14     }
15 }
16
17 class ScanReader{
18     BufferedReader br;
19     StringTokenizer st;
20     public ScanReader(InputStream in){
21         br=new BufferedReader(new InputStreamReader(in));
22         st=null;
23     }
24     public boolean hasNext(){
25         while(st==null || !st.hasMoreTokens()){
26             try{
27                 if(!br.ready())
28                     return false;
29                 st=new StringTokenizer(br.readLine());
30             }
31             catch(IOException e){
32                 throw new RuntimeException(e);
33             }
34         }
35         return true;
36     }
37     public String nextLine(){
38         hasNext();
39         return st.nextToken("");
40     }
41     public String next(){
42         hasNext();
43         return st.nextToken();
44     }
45 }

```

3 グラフ

3.1 定義

マッチング	端点を共有しない辺集合
独立集合	どの2点も隣接しない頂点集合
クリーク	どの2点も隣接している頂点集合
支配集合	自身とその隣接頂点のみで全頂点を被覆する頂点集合
辺被覆	どの頂点も少なくとも1つの辺に接続している辺集合
点被覆	どの辺も少なくとも1つの頂点に接続している頂点集合

3.2 定理

グラフ $G = (V, E)$ について,

$$| \text{最大マッチング} | + | \text{最小辺被覆} | = |V|$$

$$| \text{最大独立集合} | + | \text{最小点被覆} | = |V|$$

二部グラフであるとき,

$$| \text{最大マッチング} | = | \text{最小点被覆} |$$

$$| \text{最大独立集合} | = | \text{最小辺被覆} |$$

Dilworth の定理・Mirsky の定理

推移性をみたす DAG であるとき,

Dilworth の定理: 濃度最大の反鎖の濃度 = 個数最小の鎖被覆の個数

Mirsky の定理: 濃度最大の鎖の濃度 = 個数最小の反鎖被覆の個数

DAG の最小独立パス被覆

(最小独立パス被覆) = $|V| - (V + V')$ を頂点とする二部グラフの最大マッチング)

DAG が推移性をもつとき,

(最小独立パス被覆) = (最小パス被覆)

3.3 基本要素

```

1 struct Edge{
2     int src,dst,weight;
3     Edge(){}
4     Edge(int s,int d,int w):src(s),dst(d),weight(w){}
5 };
6 typedef vector<vector<Edge>> > Graph;
7 bool operator< (const Edge& a,const Edge& b){return a.weight<b.weight;}
8 bool operator> (const Edge& a,const Edge& b){return a.weight>b.weight;}

```

3.4 単一始点最短路 (Dijkstra)

計算量: $O(E \log V)$

```

1 // Verify: UVA 341, AOJ 0155
2
3 void Dijkstra(const Graph& g,int v,vi& dist,vi& prev)
4 {
5     priority_queue<Edge,vector<Edge>,greater<Edge>> pq;
6     pq.emplace(-1,v,0);
7     while(pq.size()){
8         Edge cur=pq.top(); pq.pop();
9         if(dist[cur.dst]!=INF) continue;
10        dist[cur.dst]=cur.weight;
11        prev[cur.dst]=cur.src;
12        for(Edge e:g[cur.dst])
13            pq.emplace(e.src,e.dst,cur.weight+e.weight);
14    }
15 }
16
17 void BuildPath(const vi& prev,int v,vi& path)
18 {
19     for(int u=v;u!=-1;u=prev[u])
20         path.push_back(u);
21     reverse(all(path));
22 }

```

3.5 単一始点最短路 (Bellman-Ford)

計算量: $O(VE)$

```

1 // Verify: UVA 341
2
3 bool BellmanFord(const Graph& g,int begin,vi& cost,vi& prev)
4 {
5     int size=g.size();
6     cost.assign(size,INFTY);
7     prev.assign(size,-1);
8

```

```

9     cost[begin]=0;
10     rep(k,size-1){
11         rep(i,size){
12             rep(j,g[i].size()){
13                 Edge cand=g[i][j]; // candidate
14                 if(cost[cand.dst]>cost[i]+cand.weight){
15                     cost[cand.dst]=cost[i]+cand.weight;
16                     prev[cand.dst]=cand.src;
17                 }
18             }
19         }
20     }
21     rep(i,size){
22         rep(j,g[i].size()){
23             Edge cand=g[i][j]; // candidate
24             if(cost[cand.dst]>cost[j]+cand.weight)
25                 return false;
26         }
27     }
28     return true;
29 }
30
31 void BuildPath(const vi& prev,int begin,int end,vi& path)
32 {
33     path.clear();
34     for(int i=end;i=prev[i]){
35         path.push_back(i);
36         if(i==begin)
37             break;
38     }
39     reverse(all(path));
40 }

```

3.6 全点間最短路 (Warshall-Floyd)

計算量: $O(V^3)$

```

1 // Verify: PKU 0155
2
3 void WarshallFloyd(vvd& dist,vvi& next)
4 {
5     int n=dist.size();
6
7     rep(i,n) rep(j,n)
8         next[i][j]=j;
9
10    rep(k,n) rep(i,n) rep(j,n){
11        if(dist[i][j]>dist[i][k]+dist[k][j]){
12            dist[i][j]=dist[i][k]+dist[k][j];
13            next[i][j]=next[i][k];
14        }
15    }
16 }

```

3.7 最小全域木 (Prim)

root を含む最小全域木を計算する. tree に構成辺を保存し, 重みの総和を返す.

計算量: $O(E \log V)$

```

1 // Verify: A0J 0180, POJ 1287, POJ 1861
2
3 int Prim(const Graph& g,vector<Edge>& tree,int root=0)
4 {
5     tree.clear();

```

```

6
7     vi vis(g.size());
8     priority_queue<Edge,vector<Edge>,greater<Edge> > pq;
9     pq.push(Edge(-1,root,0));
10    int res=0;
11    while(pq.size()){
12        Edge cur=pq.top(); pq.pop();
13        if(vis[cur.dst])
14            continue;
15        vis[cur.dst]=true;
16        res+=cur.weight;
17        if(cur.src!=-1)
18            tree.push_back(cur);
19        foreach(e,g[cur.dst])
20            pq.push(*e);
21    }
22    return res;
23 }

```

3.8 最小全域森 (Kruskal)

forest に構成辺を保存し, 重みの総和を返す.

計算量: $O(E \log V)$

```

1 // Verify: A0J 0180, POJ 1287, POJ 1861
2
3 int Kruskal(const Graph& g,vector<Edge>& forest)
4 {
5     forest.clear();
6
7     vector<Edge> es;
8     rep(i,g.size()) rep(j,g[i].size())
9         es.push_back(g[i][j]);
10    sort(all(es));
11
12    UnionFind uf(g.size());
13    int res=0;
14    rep(i,es.size())
15        if(uf.Find(es[i].src)!=uf.Find(es[i].dst)){
16            uf.Unite(es[i].src,es[i].dst);
17            res+=es[i].weight;
18            forest.push_back(es[i]);
19        }
20    return res;
21 }

```

3.9 最小シュタイナー木 (Dreyfus-Wagner)

与えられた隣接行列とターミナルの集合に対し, 最小シュタイナー木の重みの総和を返す.

計算量: $O(3^T V + 2^T V^2 + V^3)$

```

1 // Verify: A0J 1040
2
3 int DreyfusWagner(const vvi& _dist,const vi& ts)
4 {
5     int n=_dist.size();
6     vvi dist=_dist;
7     rep(k,n) rep(i,n) rep(j,n)
8         dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);
9
10    int tsize=ts.size();
11    vvi dp(1<<tsize,vi(n,INFTY));
12    rep(i,tsize) rep(j,n)

```

```

13     dp[1<<i][j]=dist[ts[i]][j];
14     rep(i,1<<tsize){
15         for(int j=i;j;j=i&(j-1)) rep(k,n)
16             dp[i][k]=min(dp[i][k],dp[j][k]+dp[i^j][k]);
17         rep(j,n) rep(k,n)
18             dp[i][j]=min(dp[i][j],dp[i][k]+dist[k][j]);
19     }
20     return *min_element(all(dp.back()));
21 }

```

3.10 最大独立集合

計算量: $O(1.466^n n)$

```

1 // Verify: POJ 1419, SRM 589 Medium
2
3 void DFS(const Graph& g,int u,vi& vis,int cur,int rem,int& res)
4 {
5     int n=g.size();
6     if(cur+rem<=res) return;
7     res=max(res,cur);
8     if(u==n) return;
9
10    if(vis[u]){
11        DFS(g,u+1,vis,cur,rem,res);
12        return;
13    }
14
15    // use u
16    vi tmp=vis;
17    vis[u]=1;
18    rep(i,g[u].size()){
19        int v=g[u][i].dst;
20        if(!vis[v]) vis[v]=1,rem--;
21    }
22    DFS(g,u+1,vis,cur+1,rem-1,res);
23    swap(vis,tmp);
24    rep(i,g[u].size()){
25        int v=g[u][i].dst;
26        if(!vis[v]) rem++;
27    }
28
29    // don't use u
30    if(g[u].size()>1){
31        vis[u]=1;
32        DFS(g,u+1,vis,cur,rem-1,res);
33        vis[u]=0;
34    }
35 }
36
37 int MaxIndependentSet(const Graph& g)
38 {
39     int n=g.size();
40     vi vis(n);
41     int res=0;
42     DFS(g,0,vis,0,n,res);
43     return res;
44 }

```

3.11 トポロジカルソート

計算量: $O(V + E)$

```

1 // Verify: UVA 10305
2

```

```

3 bool Visit(const Graph& g,int v,vi& color,vi& order)
4 {
5     if(color[v]==1)
6         return false;
7     if(color[v]==2)
8         return true;
9     color[v]=1;
10    rep(i,g[v].size())
11        if(!Visit(g,g[v][i].dst,color,order))
12            return false;
13    color[v]=2;
14    order.push_back(v);
15    return true;
16 }
17
18 bool TopologicalSort(const Graph& g,vi& order)
19 {
20     int size=g.size();
21     order.clear();
22     vi colors(size);
23     rep(i,size){
24         if(!Visit(g,i,colors,order)){
25             order.clear();
26             return false;
27         }
28     }
29     reverse(all(order));
30     return true;
31 }

```

3.12 強連結成分分解

計算量: $O(V + E)$

```

1 // Verify: POJ 2186, AOJ 2505
2
3 void DFS(const Graph& g,int v,vi& vis,vi& order)
4 {
5     if(vis[v]) return;
6     vis[v]=1;
7     foreach(e,g[v])
8         DFS(g,e->dst,vis,order);
9     order.push_back(v);
10 }
11
12 vvi SCC(const Graph& g)
13 {
14     int n=g.size();
15     vi vis(n),order;
16     rep(i,n) DFS(g,i,vis,order);
17     Graph t(n);
18     rep(i,n) foreach(e,g[i])
19         t[e->dst].push_back(Edge(e->dst,e->src));
20     fill(all(vis),0);
21     vvi res;
22     per(i,n) if(!vis[order[i]]){
23         res.resize(res.size()+1);
24         DFS(t,order[i],vis,res.back());
25     }
26     return res;
27 }

```

3.13 橋

計算量: $O(V + E)$

```

1 // Verify: LOJ 1026
2
3 int DFS(const Graph& g, int u, int prev, int x, vi& ord, vi& low)
4 {
5     ord[u]=low[u]=x++;
6     rep(i, g[u].size()){
7         int v=g[u][i].dst;
8         if(ord[v]==-1){
9             x=DFS(g, v, u, x, ord, low);
10            low[u]=min(low[u], low[v]);
11        }
12        else if(v!=prev)
13            low[u]=min(low[u], ord[v]);
14    }
15    return x;
16 }
17
18 void Bridge(const Graph& g, vector<Edge>& bs)
19 {
20     bs.clear();
21     int n=g.size();
22     vi ord(n, -1), low(n);
23     rep(i, n) if(ord[i]==-1)
24         DFS(g, i, -1, 0, ord, low);
25     rep(i, n) foreach(e, g[i])
26         if(ord[e->src]<low[e->dst])
27             bs.push_back(*e);
28 }

```

3.14 関節点

計算量: $O(V + E)$

```

1 // Verify: POJ 1144
2
3 int DFS(const Graph& g, int u, int prev, int x, vi& ord, vi& low, vi& as)
4 {
5     ord[u]=low[u]=x++;
6     int deg=0, isart=0;
7     rep(i, g[u].size()){
8         int v=g[u][i].dst;
9         if(ord[v]==-1){
10            x=DFS(g, v, u, x, ord, low, as);
11            low[u]=min(low[u], low[v]);
12            deg++;
13            isart|=ord[u]<=low[v];
14        }
15        else if(v!=prev)
16            low[u]=min(low[u], ord[v]);
17    }
18    if(prev==-1 && deg>=2 || prev!=-1 && isart)
19        as.push_back(u);
20    return x;
21 }
22
23 void Articulation(const Graph& g, vi& as)
24 {
25     as.clear();
26     int n=g.size();
27     vi ord(n, -1), low(n);
28     rep(i, n) if(ord[i]==-1)
29         DFS(g, i, -1, i, ord, low, as);
30 }

```

3.15 2-SAT

$u \vee v \Leftrightarrow (\bar{u} \rightarrow \bar{v}) \wedge (\bar{v} \rightarrow \bar{u})$ と式変形し, 二項関係 \rightarrow で辺を張り強連結成分分解する. 強連結成分の真偽は一致するため, x と \bar{x} が同じ強連結成分に属するならば充足不能. そうでないならば充足可能.

3.16 最小共通先祖

計算量: 構築 $O(V \log V)$, クエリ $O(\log V)$

```

1 // Verify: POJ 1330
2
3 struct LCA{
4     vi ds;
5     vvi ps;
6     LCA(vvi& g, int root){
7         ds.resize(g.size());
8         ps.resize(33-__builtin_clz(g.size()-1), vi(g.size(), -1));
9         DFS(g, root, -1, 0);
10        rep(i, ps.size()-1)
11            rep(j, ps[i].size())
12                ps[i+1][j]=ps[i][j]==-1?-1:ps[i][ps[i][j]];
13    }
14    void DFS(vvi& g, int cur, int prev, int depth){
15        ds[cur]=depth;
16        ps[0][cur]=prev;
17        rep(i, g[cur].size())
18            if(g[cur][i]!=prev)
19                DFS(g, g[cur][i], cur, depth+1);
20    }
21    int Find(int a, int b){
22        if(ds[a]>ds[b])
23            swap(a, b);
24        rep(i, ps.size())
25            if(ds[b]-ds[a]&1<i)
26                b=ps[i][b];
27        if(a==b)
28            return a;
29        per(i, ps.size())
30            if(ps[i][a]!=ps[i][b]){
31                a=ps[i][a];
32                b=ps[i][b];
33            }
34        return ps[0][a];
35    }
36 };

```

3.17 二部性判定

計算量: $O(V + E)$

```

1 // Verify: SRM618 Family
2
3 bool DFS(const Graph& g, int u, int c, vi& color){
4     if(color[u]!=-1)
5         return color[u]==c;
6     color[u]=c;
7     rep(i, g[u].size())
8         if(!DFS(g, g[u][i].dst, !c, color))
9             return false;
10    return true;
11 }
12
13 bool IsBipartite(const Graph& g){
14     int n=g.size();
15     vi color(n, -1);

```

```

16     rep(i,n) if(color[i]==-1)
17         if(!DFS(g,i,0,color))
18             return false;
19     return true;
20 }

```

4 フロー

4.1 最大流 (Dinic)

計算量: $O(V^2E)$

```

1 // headはvector<int>(頂点数,-1)で初期化されていること.
2 // 典型的な問題では, グラフの各辺について
3 //   AddEdge(es,head,src,dst,0,cap);
4 //   AddEdge(es,head,src,dst,0,0);
5 //   と書く.
6 // Verify: POJ 1459(141ms), POJ 3498(2625ms), SPOJ 4110(int->ll,2.81s), AOJ 2076(0.025s)
7 //   UVA 10249(0.588s), COJ 1644(421ms)
8
9 struct Edge{
10     int src,dst,flow,capacity,next;
11     Edge(){}
12     Edge(int s,int d,int f,int c,int n):src(s),dst(d),flow(f),capacity(c),next(n){}
13 };
14
15 void AddEdge(vector<Edge>& es,vi& head,int src,int dst,int flow,int capacity)
16 {
17     es.push_back(Edge(src,dst,flow,capacity,head[src]));
18     head[src]=es.size()-1;
19 }
20
21 vi BFS(const vector<Edge>& es,const vi& head,int begin)
22 {
23     int size=head.size();
24     vi label(size,size);
25     queue<pii> q;
26     q.push(mp(begin,0));
27     while(q.size()){
28         pii cur=q.front(); q.pop();
29         if(label[cur.first]<=cur.second)
30             continue;
31         label[cur.first]=cur.second;
32         for(int i=head[cur.first];i!=-1;i=es[i].next)
33             if(es[i].capacity-es[i].flow>0)
34                 q.push(mp(es[i].dst,cur.second+1));
35     }
36     return label;
37 }
38
39 int DFS(vector<Edge>& es,vi& head,int v,int end,const vi& label,int f)
40 {
41     if(v==end)
42         return f;
43     for(int& i=head[v];i!=-1;i=es[i].next){
44         Edge& e=es[i];
45         if(label[e.src]>=label[e.dst])
46             continue;
47         int residue=e.capacity-e.flow;
48         if(residue<=0)
49             continue;
50         int augment=DFS(es,head,e.dst,end,label,min(residue,f));
51         if(augment>0){
52             e.flow+=augment;

```

```

53         es[i].flow-=augment;
54         return augment;
55     }
56 }
57 return 0;
58 }
59
60 int Dinic(vector<Edge>& es,const vi& head,int begin,int end)
61 {
62     int size=head.size();
63     for(;;){
64         vi label=BFS(es,head,begin);
65         if(label[end]==size)
66             break;
67         vi temp=head;
68         while(DFS(es,temp,begin,end,label,INFTY))
69             ;
70     }
71     int res=0;
72     for(int i=head[begin];i!=-1;i=es[i].next)
73         res+=es[i].flow;
74     return res;
75 }

```

4.2 最小費用流

計算量: $O(FE \log V)$

```

1 // Verify: PKU 2135(297ms), PKU 2195(157ms), UVA 10594(int ll,344ms),
2 //   SRM506Div1 Medium, SRM611 GoodCompanyDiv0ne
3
4 struct Edge{
5     int src,dst,ca,co;
6     Edge(){}
7     Edge(int s,int d,int ca,int co):src(s),dst(d),ca(ca),co(co){}
8 };
9 typedef vector<vector<Edge>> > Graph;
10
11 int MinCostFlow(const Graph& _g,int begin,int end,int flow)
12 {
13     int size=_g.size();
14     Graph g(size);
15     vvi rev(size);
16     rep(i,size) foreach(e,_g[i]){
17         g[i].push_back(*e);
18         g[e->dst].push_back(Edge(e->src,0,-e->co));
19         rev[i].push_back(g[e->dst].size()-1);
20         rev[e->dst].push_back(g[i].size()-1);
21     }
22
23     int res=0;
24     vi pots(size);
25     while(flow){
26         vi dist(size,INFTY);
27         vector<pii> prev(size,mp(-1,-1));
28         priority_queue<pii,vector<pii>,greater<pii> > pq;
29         pq.push(mp(0,begin));
30         dist[begin]=0;
31         while(!pq.empty()){
32             int curd=pq.top().first,curv=pq.top().second; pq.pop();
33             if(dist[curv]<curd)
34                 continue;
35             rep(i,g[curv].size()){
36                 Edge e=g[curv][i];
37                 if(e.cap==0)

```

```

38         continue;
39         int newdist=dist[e.src]+e.cost+pots[e.src]-pots[e.dst];
40         if(dist[e.dst]>newdist){
41             dist[e.dst]=newdist;
42             prev[e.dst]=mp(e.src,i);
43             pq.push(mp(newdist,e.dst));
44         }
45     }
46 }
47
48 if(prev[end].first==-1)
49     return -1;
50 rep(i,size)
51     pots[i]+=dist[i];
52
53 int f=flow;
54 for(int v=end;v!=begin;v=prev[v].first){
55     Edge e=g[prev[v].first][prev[v].second];
56     f=min(f,e.cap);
57 }
58 for(int v=end;v!=begin;v=prev[v].first){
59     int i=prev[v].first,j=prev[v].second;
60     g[i][j].cap-=f;
61     g[g[i][j].dst][rev[i][j]].cap+=f;
62 }
63 flow-=f;
64 res+=f*pots[end];
65 }
66
67 return res;
68 }

```

4.3 2部グラフの最大マッチング (naïve)

計算量: $O(VE)$

```

1 // Verify: A0J 1163, LOJ 1149
2
3 struct Edge{
4     int src,dst;
5     Edge(){}
6     Edge(int s,int d):src(s),dst(d){}
7 };
8 typedef vector<vector<Edge> > Graph;
9
10 bool Augment(const Graph& g,int u,vi& vis,vi& match)
11 {
12     rep(i,g[u].size()){
13         int v=g[u][i].dst;
14         if(vis[v]) continue;
15         vis[v]=1;
16         if(match[v]==-1 || Augment(g,match[v],vis,match)){
17             match[u]=v;
18             match[v]=u;
19             return true;
20         }
21     }
22     return false;
23 }
24
25 int BipartiteMatching(const Graph& g,vi& match)
26 {
27     int n=g.size();
28     match.assign(n,-1);
29     int res=0;

```

```

30     rep(i,n) if(match[i]==-1){
31         vi vis(n);
32         res+=Augment(g,i,vis,match);
33     }
34     return res;
35 }

```

4.4 2部グラフの最大マッチング (Hopcroft-Karp)

計算量: $O(\sqrt{VE})$

```

1 // Verify: A0J 1163, LOJ 1171, LOJ 1356, POJ 1466, POJ 2446, SPOJ 4206 (2.31s)
2
3 struct Edge{
4     int src,dst;
5     Edge(){}
6     Edge(int s,int d):src(s),dst(d){}
7 };
8 typedef vector<vector<Edge> > Graph;
9
10 bool BFS(const Graph& g,const vi& side,const vi& match,vi& level)
11 {
12     int n=g.size();
13     level.assign(n,n);
14     queue<pii> q;
15     rep(i,n) if(side[i]==0 && match[i]==-1)
16         q.push(mp(i,0));
17     bool res=false;
18     while(!q.empty()){
19         pii cur=q.front(); q.pop();
20         int u=cur.first,l=cur.second;
21         if(level[u]<=l) continue;
22         level[u]=l;
23         rep(i,g[u].size()){
24             int v=g[u][i].dst;
25             if(match[v]==-1)
26                 res=true;
27             else
28                 q.push(mp(match[v],l+2));
29         }
30     }
31     return res;
32 }
33
34 bool DFS(const Graph& g,const vi& side,int u,vi& match,vi& level)
35 {
36     rep(i,g[u].size()){
37         int v=g[u][i].dst;
38         if(match[v]==-1 || level[match[v]]>level[u] && DFS(g,side,match[v],match,level)){
39             match[u]=v;
40             match[v]=u;
41             return true;
42         }
43     }
44     level[u]=-1;
45     return false;
46 }
47
48 int HopcroftKarp(const Graph& g,const vi& side,vi& match)
49 {
50     int n=g.size();
51     match.assign(n,-1);
52     int res=0;
53     for(vi level;BFS(g,side,match,level);)
54         rep(i,n) if(side[i]==0 && match[i]==-1)

```



```

55         res+=DFS(g,side,i,match,level);
56     return res;
57 }

```

4.5 DAG の最小独立パス被覆

```

1 // Verify: AOJ 2251
2
3 int MinPathCover(const Graph& g)
4 {
5     int n=g.size();
6     Graph bg(2*n);
7     rep(i,n) foreach(e,g[i])
8         bg[e->src].push_back(Edge(e->src,n+e->dst));
9     vi side(2*n),match(2*n);
10    fill(n+all(side),1);
11    return n-HopcroftKarp(bg,side,match);
12 }

```

4.6 無向グラフの全域最小カット (Nagamochi-Ibaraki)

グラフが連結でないとき 0 を返す . 計算量 : $O(V^3)$

```

1 // Verify: POJ 2914(5657ms)
2
3 int NagamochiIbaraki(const Graph& g)
4 {
5     int size=g.size();
6     vvi a(size,vi(size));
7     rep(i,size) foreach(e,g[i])
8         a[e->src][e->dst]+=e->weight;
9     vi toi(size);
10    rep(i,size) toi[i]=i;
11
12    int res=INFTY;
13    for(int n=size;n>=2;n--){
14        vi ws(n);
15        int u,v,w;
16        rep(i,n){
17            u=v; v=max_element(all(ws))-ws.begin();
18            w=ws[v]; ws[v]=-1;
19            rep(j,n) if(ws[j]>=0)
20                ws[j]+=a[toi[v]][toi[j]];
21        }
22        res=min(res,w);
23        rep(i,n){
24            a[toi[u]][toi[i]]+=a[toi[v]][toi[i]];
25            a[toi[i]][toi[u]]+=a[toi[i]][toi[v]];
26        }
27        toi.erase(toi.begin()+v);
28    }
29    return res;
30 }

```

5 文字列

5.1 Knuth-Morris-Pratt

最短周期は $n - \text{fail}[n]$ で求まる .

計算量 : テキスト長 N , パターン長 M として , 初期化 $O(M)$, 検索 $O(M + N)$

```
1 // Verify: [ctor] POJ 1961, POJ 2406, [MatchAll] POJ 3461, LOJ 1255, SPOJ 32
```

```

2
3 struct KMP{
4     string pat;
5     vi fail;
6     KMP(const string& p):pat(p),fail(p.size()+1){
7         fail[0]=-1;
8         for(int i=0,j=-1;i<pat.size();){
9             while(j>=0 && pat[i]!=pat[j])
10                 j=fail[j];
11             i++,j++;
12             fail[i]=j;
13         }
14     }
15     int Match(const string& text){
16         for(int i=0,j=0;i<text.size();){
17             while(j>=0 && text[i]!=pat[j])
18                 j=fail[j];
19             i++,j++;
20             if(j==pat.size())
21                 return i-j;
22         }
23         return -1;
24     }
25     vi MatchAll(const string& text){
26         vi res;
27         for(int i=0,j=0;i<text.size();){
28             while(j>=0 && text[i]!=pat[j])
29                 j=fail[j];
30             i++,j++;
31             if(j==pat.size()){
32                 res.push_back(i-j);
33                 j=fail[j];
34             }
35         }
36         return res;
37     }
38 };

```

5.2 最長回文 (Manacher)

計算量 : $O(N)$

```

1 // Verify: PKU 3974
2
3 int LongestPalindrome(const string& _s)
4 {
5     int n=_s.size();
6     string s(2*n+1, '.');
7     rep(i,n)
8         s[i*2+1]=_s[i];
9     n=s.size();
10
11    vi rad(n);
12    for(int i=0,j=0;i<n;){
13        for(;0<=i-j && i+j<n && s[i-j]==s[i+j];j++)
14            ;
15        rad[i]=j;
16
17        int k=1;
18        for(;0<=i-k && i+k<n && rad[i-k]<rad[i]-k;k++)
19            rad[i+k]=rad[i-k];
20
21        j=max(rad[i]-k,0);
22        i+=k;
23    }

```

```

24
25     return *max_element(all(rad))-1;
26 }

```

5.3 Suffix Array (Kärkkäinen-Sanders)

計算量：構築 $O(N)$, LCP の計算 $O(N)$.

```

1  // Verify: [構築] LiveArchive 3451, POJ 1509, POJ 2774, POJ 3080, POJ 3261, POJ 3450,
2  //              SPOJ 6409 (0.76s), UVA 1223, UVA 1227, UVA 11512
3  //              [LCP] LiveArchive 3451, POJ 2774, POJ 3080, POJ 3261, POJ 3450,
4  //              UVA 1223, UVA 1227, UVA 11512
5  //              [LCS] POJ 3080, UVA 1227
6
7  template<typename C>
8  void RadixSort(const vi& src,vi& dst,const C& s,int ofs,int n,int asize)
9  {
10     vi hist(asize+1);
11     rep(i,n) hist[s[ofs+src[i]]]++;
12     rep(i,asize) hist[i+1]+=hist[i];
13     per(i,n) dst[--hist[s[ofs+src[i]]]]=src[i];
14 }
15
16 bool Less(int a1,int a2,int b1,int b2)
17 {
18     return a1!=b1?a1<b1:a2<b2;
19 }
20 bool Less(int a1,int a2,int a3,int b1,int b2,int b3)
21 {
22     return a1!=b1?a1<b1:Less(a2,a3,b2,b3);
23 }
24
25 // s[0..n-1]: 入力文字列 . 末尾に'\0'が3つ必要(s[n]=s[n+1]=s[n+2]=0) .
26 // sa[0..n-1]: 接尾辞配列 .
27 // asize: アルファベットサイズ, s[i] \in [1,asize]
28 template<typename C>
29 void KaerkkainenSanders(const C& s,vi& sa,int asize)
30 {
31     int n=s.size();
32     int n0=(n+2)/3,n1=(n+1)/3,n2=n/3,n02=n0+n2;
33     vi s12(n02+3),sa12(n02);
34
35     for(int i=0,j=0;i<n+(n0-n1);i++)
36         if(i%3) s12[j++]=i;
37     RadixSort(s12,sa12,s,2,n02,asize);
38     RadixSort(sa12,s12,s,1,n02,asize);
39     RadixSort(s12,sa12,s,0,n02,asize);
40
41     int name=0,x=-1,y=-1,z=-1;
42     rep(i,n02){
43         int j=sa12[i];
44         if(s[j]!=x || s[j+1]!=y || s[j+2]!=z)
45             x=s[j],y=s[j+1],z=s[j+2],name++;
46         if(j%3==1) s12[j/3]=name;
47         else
48             s12[n0+j/3]=name;
49     }
50     if(name==n02) // unique
51         rep(i,n02) sa12[s12[i]-1]=i;
52     else{
53         KaerkkainenSanders(s12,sa12,name);
54         rep(i,n02) s12[sa12[i]]=i+1;
55     }
56
57     vi s0(n0),sa0(n0);

```

```

58     for(int i=0,j=0;i<n02;i++)
59         if(sa12[i]<n0) s0[j++]=3*sa12[i];
60     RadixSort(s0,sa0,s,0,n0,asize);
61
62     int i=0,j=n0-n1,k=0;
63     while(i<n0 && j<n02){
64         int p=sa0[i],q=sa12[j]<n0?sa12[j]*3+1:(sa12[j]-n0)*3+2;
65         if(sa12[j]<n0?
66             Less(s[p],s12[p/3],s[q],s12[n0+sa12[j]]):
67             Less(s[p],s[p+1],s12[n0+p/3],s[q],s[q+1],s12[sa12[j]+1-n0]))
68             sa[k++]=p,i++;
69         else
70             sa[k++]=q,j++;
71     }
72     for(;i<n0;k++,i++)
73         sa[k]=sa0[i];
74     for(;j<n02;k++,j++)
75         sa[k]=sa12[j]<n0?sa12[j]*3+1:(sa12[j]-n0)*3+2;
76 }
77 void KaerkkainenSanders(const string& s,vi& sa)
78 {
79     KaerkkainenSanders<string>(s+string(3,0),sa,127);
80 }
81 void KaerkkainenSanders(const char* s,vi& sa)
82 {
83     KaerkkainenSanders<string>(s+string(3,0),sa,127);
84 }
85
86 // s[0..n-1] (s[n]=0), sa[0..n-1]
87 // lcp[0..n-1] (lcp[i]:s[sa[i-1]..]とs[sa[i]..]のLCP.lcp[0]=0)
88 template<typename C>
89 void LCP(const C& s,const vi& sa,vi& lcp)
90 {
91     int n=s.size();
92     vi rank(n);
93     rep(i,n) rank[sa[i]]=i;
94     for(int i=0,h=0;i<n;i++){
95         if(h>0) h--;
96         if(rank[i]>0)
97             for(int j=sa[rank[i]-1];;h++){
98                 if(s[j+h]!=s[i+h]) break;
99                 lcp[rank[i]]=h;
100             }
101     }
102
103     // n: 文字列の数
104     // ls[0..n-1]: 各文字列の長さ
105     // LCSはs.substr(sa[res],lcp[res])で得られる .
106     // LCSが空の時res=0 . lcp[res]=0でもある .
107     int LCS(int n,const vi& ls,const vi& sa,const vi& lcp)
108     {
109         int m=s.size();
110         vi is(m);
111         for(int i=0,j=0;i<n;i++){
112             rep(k,ls[i]) is[j+k]=i;
113             j+=ls[i];
114             if(i<n-1) is[j++]=i;
115         }
116         int cnt=0,res=0;
117         vi occ(n);
118         deque<int> q;
119         for(int i=n-1,j=n-1;i<m;i++){
120             for(;j<m && cnt<n;j++){
121                 if(++occ[is[sa[j]]]==1) cnt++;
122                 while(!q.empty() && lcp[q.back()]>lcp[j]) q.pop_back();

```

```

123         q.push_back(j);
124     }
125     if(cnt<n) break;
126     if(q.front()==i) q.pop_front();
127     assert(q.size());
128     if(lcp[res]<lcp[q.front()])
129         res=q.front();
130     if(--occ[is[sa[i]]]==0) cnt--;
131 }
132 return res;
133 }

```

5.4 Suffix Array (SA-IS)

計算量：構築 $O(N)$, LCP の計算 $O(N)$. ただし $|\Sigma| = O(1)$ とする .

```

1 // Verify: UVA 1227(5.436s), SPOJ 6409(0.51s), POJ 1743, POJ 3261
2
3 enum{LTYPE,STYPE};
4
5 inline bool IsLMS(const vector<bool>& ts,int i)
6 {
7     return i>0 && ts[i]==STYPE && ts[i-1]==LTYPE;
8 }
9
10 template<typename T>
11 void InitBucket(const T* s,int n,int* bucket,int bsize,bool inclusive)
12 {
13     fill(bucket,bucket+bsize,0);
14     rep(i,n)
15         bucket[s[i]]++;
16     for(int i=0,sum=0;i<bsize;i++){
17         sum+=bucket[i];
18         bucket[i]=inclusive?sum:sum-bucket[i];
19     }
20 }
21
22 template<typename T>
23 void InduceSAL(const T* s,int* sa,int n,int* bucket,int bsize)
24 {
25     InitBucket(s,n,bucket,bsize,false);
26     rep(i,n){
27         int j=sa[i]-1;
28         if(j>=0 && s[j]>=s[j+1])
29             sa[bucket[s[j]]++]=j;
30     }
31 }
32
33 template<typename T>
34 void InduceSAS(const T* s,int* sa,int n,int* bucket,int bsize)
35 {
36     InitBucket(s,n,bucket,bsize,true);
37     per(i,n){
38         int j=sa[i]-1;
39         if(j>=0 && s[j]<=s[j+1])
40             sa[--bucket[s[j]]]=j;
41     }
42 }
43
44 // s[0..n-1]: 入力文字列, n>=2かつs[n-1]=0であること
45 // sa[0..n-1]: 接尾辞配列
46 // asize: アルファベットサイズ, s[n-1]を除き1以上asize以下
47 template<typename T>
48 void SAIS(const T* s,int* sa,int n,int asize)
49 {

```

```

50     vector<bool> ts(n,STYPE);
51     per(i,n-1)
52         if(s[i]>s[i+1] || (s[i]==s[i+1] && ts[i+1]==LTYPE))
53             ts[i]=LTYPE;
54
55     fill(sa,sa+n,-1);
56
57     int* bucket=new int[asize+1];
58     InitBucket(s,n,bucket,asize+1,true);
59     per(i,n) if(IsLMS(ts,i))
60         sa[--bucket[s[i]]]=i;
61     InduceSAL(s,sa,n,bucket,asize+1);
62     InduceSAS(s,sa,n,bucket,asize+1);
63     delete[] bucket;
64
65     int n1=0;
66     rep(i,n) if(IsLMS(ts,sa[i]))
67         sa[n1++]=sa[i];
68     fill(sa+n1,sa+n,-1);
69
70     int name=0,prev=-1;
71     rep(i,n1){
72         int cur=sa[i];
73         bool diff=prev!=-1;
74         for(int j=0;!diff;j++){
75             if(j>0 && IsLMS(ts,cur+j))
76                 break;
77             if(s[cur+j]!=s[prev+j])
78                 diff=true;
79         }
80         if(diff)
81             name++;
82         sa[n1+cur/2]=name-1;
83         prev=cur;
84     }
85     for(int i=n1,j=n1;i<n;i++)
86         if(sa[i]>=0)
87             sa[j++]=sa[i];
88     fill(sa+n1+n1,sa+n,-1);
89
90     int *sa1=sa,*s1=sa+n1;
91     if(name==n1)
92         rep(i,n1) sa1[s1[i]]=i;
93     else
94         SAIS(s1,sa1,n1,name);
95
96     n1=0;
97     rep(i,n) if(IsLMS(ts,i))
98         s1[n1++]=i;
99     rep(i,n1)
100         sa1[i]=s1[sa1[i]];
101     fill(sa+n1,sa+n,-1);
102
103     bucket=new int[asize+1];
104     InitBucket(s,n,bucket,asize+1,true);
105     per(i,n1){
106         int j=sa1[i]; sa1[i]=-1;
107         sa[--bucket[s[j]]]=j;
108     }
109     InduceSAL(s,sa,n,bucket,asize+1);
110     InduceSAS(s,sa,n,bucket,asize+1);
111     delete[] bucket;
112 }
113
114 // nはsとsaのサイズ, lcpのサイズはn-1

```

```

115 template<typename T>
116 void LCP(const T* s, const int* sa, int n, int* lcp)
117 {
118     vi rank(n);
119     rep(i, n) rank[sa[i]] = i;
120     for(int i=0, h=0; i<n-1; i++){
121         if(h>0) h--;
122         for(int j=sa[rank[i]-1];; h++){
123             if(s[j+h] != s[i+h])
124                 break;
125             lcp[rank[i]-1] = h;
126         }
127     }

```

6 動的計画法

6.1 最長共通部分列

計算量: $O(MN)$

```

1 // Verify: PKU 1458
2
3 int LCS(const string& s1, const string& s2)
4 {
5     vvi dp(2, vi(s2.size()+1));
6     rep(i, s1.size()){
7         rep(j, s2.size()){
8             int ii=i+1, jj=j+1;
9             if(s1[i]==s2[j])
10                 dp[ii&1][jj]=dp[(ii-1)&1][jj-1]+1;
11             else
12                 dp[ii&1][jj]=max(dp[(ii-1)&1][jj-1], max(dp[(ii-1)&1][jj], dp[ii&1][jj-1]));
13         }
14     }
15     return dp[s1.size()&1][s2.size()];
16 }

```

6.2 最長増加部分列

計算量: $O(N \log N)$.

```

1 // Verify: PKU 3903
2
3 int LIS(const vi& a)
4 {
5     vi dp;
6     rep(i, a.size()){
7         int j=lower_bound(all(dp), a[i]) - dp.begin();
8         if(j==dp.size())
9             dp.push_back(a[i]);
10        else
11            dp[j]=a[i];
12    }
13    return dp.size();
14 }

```

7 データ構造

7.1 Union-Find Tree

計算量: 各操作に関して $O(\alpha(N))$. ただし $\alpha(N)$ はアッカーマン関数の逆関数.

```

1 // Verify: PKU 2524, PKU 1182
2
3 struct UnionFind{
4     vi data;
5     UnionFind(int size):data(size, -1){}
6     int Find(int n){
7         return data[n]<0?n:(data[n]=Find(data[n]));
8     }
9     void Unite(int a, int b){
10         int ra=Find(a), rb=Find(b);
11         if(ra!=rb){
12             if(-data[ra]<-data[rb])
13                 swap(ra, rb);
14             data[ra] += data[rb];
15             data[rb] = ra;
16         }
17     }
18     int Size(int n){
19         return -data[Find(n)];
20     }
21 };

```

7.2 Fenwick Tree

計算量: 各操作に関して $O(\log N)$

```

1 // 点更新/区間質問
2 // Verify: POJ 3067, POJ 3928
3
4 struct FenwickTree{
5     vector<ll> data;
6     FenwickTree(int n):data(n+1){}
7     void Add(int i, int x){
8         for(i++; i<data.size(); i+=i&-i)
9             data[i] += x;
10    }
11    ll Sum(int i){
12        ll res=0;
13        for(; i>0; i-=i&-i)
14            res += data[i];
15        return res;
16    }
17    ll Sum(int i, int j){
18        return Sum(j) - Sum(i);
19    }
20 };
21
22 // 区間更新/点質問
23 // Verify: AOJ 2412, LOJ 1080, POJ 2182
24
25 struct FenwickTree{
26     vector<ll> data;
27     FenwickTree(int n):data(n+1){}
28     void AddRange(int i, int x){
29         for(; i>0; i-=i&-i)
30             data[i] += x;
31    }
32     void AddRange(int i, int j, int x){
33         AddRange(i, -x);
34         AddRange(j, x);
35    }
36     int Get(int i){
37         ll res=0;
38         for(i++; i<data.size(); i+=i&-i)

```

```

39         res+=data[i];
40         return res;
41     }
42 };
43
44 // 2次元版, 点更新/区間質問
45 // Verify: POJ 1195
46
47 struct FenwickTree2D{
48     vector<FenwickTree> data;
49     FenwickTree2D(int h,int w):data(h+1,FenwickTree(w)){}
50     void Add(int i,int j,int x){
51         for(i++;i<data.size();i+=i&-i)
52             data[i].Add(j,x);
53     }
54     ll Sum(int i,int j){
55         ll res=0;
56         for(;i; i-=i&-i)
57             res+=data[i].Sum(j);
58         return res;
59     }
60     ll Sum(int i,int j,int k,int l){
61         return Sum(k,l)-Sum(k,j)-Sum(i,l)+Sum(i,j);
62     }
63 };

```

7.3 Segment Tree

計算量: 初期化 $O(N)$, 更新 $O(\log N)$, クエリ $O(\log N)$

```

1 // Verify: UVa 12299, POJ 3264, AOJ 1068
2
3 int Need(int x)
4 {
5     x--;
6     rep(i,5) x|=x>>(1<<i);
7     return x+1;
8 }
9
10 struct SegmentTree{
11     int size;
12     vi data;
13     SegmentTree(int s):size(Need(s)),data(size*2,INFTY){}
14     SegmentTree(const vi& a):size(Need(a.size())),data(size*2,INFTY){
15         copy(all(a),data.begin()+size);
16         for(int i=size;--i;)
17             data[i]=min(data[i*2],data[i*2+1]);
18     }
19     int Get(int i){
20         return data[size+i];
21     }
22     void Update(int i,int x){
23         data[size+i]=x;
24         for(i=(i+size)/2;i;i/=2)
25             data[i]=min(data[i*2],data[i*2+1]);
26     }
27     int Query(int a,int b,int i,int l,int r){
28         if(b<=l || r<=a) return INFTY;
29         if(a<=l && r<=b) return data[i];
30         return min(Query(a,b,i*2,l,(l+r)/2),Query(a,b,i*2+1,(l+r)/2,r));
31     }
32     int Query(int a,int b){
33         return Query(a,b,1,0,size);
34     }
35 };

```

```

36
37 // 2次元版
38 // Verify: AOJ 1068
39
40 struct SegmentTree2D{
41     int size;
42     vector<SegmentTree> data;
43     SegmentTree2D(int h,int w):size(Need(h)),data(size*2,SegmentTree(w)){}
44     SegmentTree2D(const vi& a):size(Need(a.size())),data(size*2,SegmentTree(a[0].size())){
45         copy(all(a),data.begin()+size);
46         for(int i=size;--i;)
47             rep(j,1,data[i].data.size())
48                 data[i].data[j]=min(data[i*2].data[j],data[i*2+1].data[j]);
49     }
50     int Get(int i,int j){
51         return data[size+i].Get(j);
52     }
53     void Update(int i,int j,int x){
54         data[size+i].Update(j,x);
55         for(i=(i+size)/2;i;i/=2)
56             data[i].Update(j,min(data[i*2].Get(j),data[i*2+1].Get(j)));
57     }
58     int Query(int a,int b,int c,int d,int i,int to,int bo){
59         if(c<=to || bo<=a) return INFTY;
60         if(a<=to && bo<=c) return data[i].Query(b,d);
61         return min(Query(a,b,c,d,i*2,to,(to+bo)/2),Query(a,b,c,d,i*2+1,(to+bo)/2,bo));
62     }
63     int Query(int a,int b,int c,int d){
64         return Query(a,b,c,d,1,0,size);
65     }
66 };
67
68 // 区間更新/区間質問 . PropagateとMergeを適切に書き換える
69 // Verify: SPOJ 7259
70
71 struct SegmentTree{
72     int size;
73     vi data,prop;
74     SegmentTree(int s):size(Need(s)),data(size*2),prop(size*2){}
75     SegmentTree(const vi& a):size(Need(a.size())),data(size*2),prop(size*2){
76         copy(all(a),data.begin()+size);
77         for(int i=size;--i;)
78             data[i]=Merge(data[i*2],data[i*2+1]);
79     }
80     void Update(int a,int b,int x,int i,int l,int r){
81         Propagate(i,l,r);
82         if(b<=l || r<=a)
83             return;
84         if(a<=l && r<=b){
85             prop[i]=x;
86             Propagate(i,l,r);
87             return;
88         }
89         Update(a,b,x,i*2+0,l,(l+r)/2);
90         Update(a,b,x,i*2+1,(l+r)/2,r);
91         data[i]=Merge(data[i*2],data[i*2+1]);
92     }
93     int Query(int a,int b,int i,int l,int r){
94         Propagate(i,l,r);
95         if(b<=l || r<=a) return 0;
96         if(a<=l && r<=b) return data[i];
97         ll x=Query(a,b,i*2+0,l,(l+r)/2);
98         ll y=Query(a,b,i*2+1,(l+r)/2,r);
99         return Merge(x,y);
100     }

```

```

101 void Update(int a,int b,int x){
102     Update(a,b,x,1,0,size);
103 }
104 int Query(int a,int b){
105     return Query(a,b,1,0,size);
106 }
107 void Propagate(int i,int l,int r){
108     if(i<size){
109         prop[i*2+0]^=prop[i];
110         prop[i*2+1]^=prop[i];
111     }
112     if(prop[i])
113         data[i]=r-l-data[i];
114     prop[i]=0;
115 }
116 int Merge(ll x,ll y){
117     return x+y;
118 }
119 };
120
121 // 区間Set,Reset,Flip,Count
122 // Verify: UVa 11402, Codeforces 242E
123
124 struct SegmentTree{
125     int size;
126     vi data,prop;
127     SegmentTree(int s):size(Need(s)),data(size*2),prop(size*2){}
128     SegmentTree(const vi& a):size(Need(a.size())),data(size*2),prop(size*2){
129         copy(all(a),data.begin()+size);
130         for(int i=size;--i;){
131             data[i]=Merge(data[i*2],data[i*2+1]);
132         }
133     void Update(int a,int b,int x,int i,int l,int r){
134         Propagate(i,l,r);
135         if(b<=l || r<=a)
136             return;
137         if(a<=l && r<=b){
138             prop[i]=x;
139             Propagate(i,l,r);
140             return;
141         }
142         Update(a,b,x,i*2+0,l,(l+r)/2);
143         Update(a,b,x,i*2+1,(l+r)/2,r);
144         data[i]=Merge(data[i*2],data[i*2+1]);
145     }
146     int Query(int a,int b,int i,int l,int r){
147         Propagate(i,l,r);
148         if(b<=l || r<=a) return 0;
149         if(a<=l && r<=b) return data[i];
150         ll x=Query(a,b,i*2+0,l,(l+r)/2);
151         ll y=Query(a,b,i*2+1,(l+r)/2,r);
152         return Merge(x,y);
153     }
154     void Propagate(int i,int l,int r){
155         if(i<size){
156             if(prop[i]==1) prop[i*2+0]=prop[i*2+1]=1;
157             if(prop[i]==2) prop[i*2+0]=prop[i*2+1]=2;
158             if(prop[i]==3){
159                 prop[i*2+0]=3-prop[i*2+0];
160                 prop[i*2+1]=3-prop[i*2+1];
161             }
162         }
163         if(prop[i]==1) data[i]=r-l;
164         if(prop[i]==2) data[i]=0;
165         if(prop[i]==3) data[i]=r-l-data[i];

```

```

166         prop[i]=0;
167     }
168     int Merge(ll x,ll y){
169         return x+y;
170     }
171     void Set(int a,int b){
172         Update(a,b,1,1,0,size);
173     }
174     void Reset(int a,int b){
175         Update(a,b,2,1,0,size);
176     }
177     void Flip(int a,int b){
178         Update(a,b,3,1,0,size);
179     }
180     int Count(int a,int b){
181         return Query(a,b,1,0,size);
182     }
183 };

```

7.4 Range Tree

計算量：初期化 ($O(N \log^2 N)$) , クエリ $O(N \log^2 N)$

```

1 // Verify: LiveArchive 5821, ACPC2012 F
2
3 struct Point{
4     int x,y;
5     Point(){}
6     Point(int x,int y):x(x),y(y){}
7 };
8 bool LessX(Point a,Point b){
9     return a.x<b.x;
10 }
11 bool LessY(Point a,Point b){
12     return a.y<b.y;
13 }
14
15 struct RangeTree2D{
16     vector<Point> yps;
17     struct RangeNode2D{
18         vector<Point> xps;
19         RangeNode2D *left,*right;
20         RangeNode2D(const vector<Point>& ps):xps(ps),left(0),right(0){
21             sort(all(xps),LessX);
22         }
23         ~RangeNode2D(){
24             delete left;
25             delete right;
26         }
27     }*root;
28     RangeTree2D(const vector<Point>& ps):yps(ps){
29         sort(all(yps),LessY);
30         root=Build(0,yps.size());
31     }
32     ~RangeTree2D(){
33         delete root;
34     }
35     RangeNode2D* Build(int first,int last){
36         RangeNode2D* node=new RangeNode2D(vector<Point>(yps.begin()+first,yps.begin()+last));
37         if(last-first>=2){
38             int middle=(first+last)/2;
39             node->left=Build(first,middle);
40             node->right=Build(middle,last);
41         }
42         return node;

```

```

43     }
44     int Query(int to,int le,int bo,int ri,RangeNode2D* root,int first,int last){
45         if(first==last || yps[last-1].y<to || bo<=yps[first].y)
46             return 0;
47         if(to<=yps[first].y && yps[last-1].y<bo){
48             iter(root->xps) i=lower_bound(all(root->xps),Point(le,0),LessX);
49             iter(root->xps) j=lower_bound(all(root->xps),Point(ri,0),LessX);
50             return j-i;
51         }
52         int middle=(first+last)/2,res=0;
53         res+=Query(to,le,bo,ri,root->left,first,middle);
54         res+=Query(to,le,bo,ri,root->right,middle,last);
55         return res;
56     }
57     int Query(int to,int le,int bo,int ri){
58         return Query(to,le,bo,ri,root,0,yps.size());
59     }
60 };

```

8 数学

8.1 数学定数

円周率 $\pi = 3.1415926535897932384626433832795029$

自然対数の底 $e = 2.7182818284590452353602874713526625$

オイラーの定数 $\gamma = 0.5772156649015328606065120900824024$

8.2 ユークリッドの互除法

ExtendedGCD は $ax + by = \gcd(a, b)$ をみたす x, y を求める . 計算量 : $O(\log \max(a, b))$

```

1 int GCD(int a,int b)
2 {
3     return b==0?a:GCD(b,a%b);
4 }
5
6 int LCM(int a,int b)
7 {
8     return a/GCD(a,b)*b;
9 }
10
11 int ExtendedGCD(int a,int b,int& x,int& y)
12 {
13     if(b==0){
14         x=1; y=0;
15         return a;
16     }
17     else{
18         int g=ExtendedGCD(b,a%b,y,x);
19         y-=a/b*x;
20         return g;
21     }
22 }

```

8.3 逆元

$ax \equiv 1 \pmod{m}$ をみたす x を求める . 逆元が存在しない場合は 0 を返す .

```

1 int ModInverse(int a,int m)
2 {
3     int x,y;
4     int g=ExtendedGCD(a,m,x,y);
5     if(g==1)

```

```

6         return (x+m)%m;
7     else
8         return 0; // invalid
9 }

```

8.4 エラトステネスの篩

計算量 : $O(N \log \log N)$

```

1 // Verify: A0J 1172
2
3 void Eratosthenes(vi& isp)
4 {
5     isp[0]=isp[1]=0;
6     fill(2+all(isp),1);
7     for(int i=2;i<isp.size();i++)
8         if(isp[i])
9             for(int j=i*i;j<isp.size();j+=i)
10                 isp[j]=0;
11 }

```

8.5 素数判定 (Miller-Rabin)

問題によって実装を切り替える .

```

1 // ModMulとwsは問題に応じて使い分けること .
2 // witnessは以下のページを参考にした :
3 // http://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test
4 // http://mathworld.wolfram.com/StrongPseudoprime.html
5 // http://oeis.org/A014233
6 // Verify: SP0J 288
7
8 // a*b mod m
9 ull ModMul(ull a,ull b,ull m)
10 {
11     //return a*b%m;
12     ull c=0;
13     for(;b>=1,(a<=1)%=m)
14         if(b&1)
15             (c+=a)%=m;
16     return c;
17 }
18 // a^r mod m
19 ull ModPow(ull a,ull r,ull m)
20 {
21     if(r==0) return 1;
22     ull b=ModPow(a,r/2,m);
23     b=ModMul(b,b,m);
24     return r&1?ModMul(b,a,m):b;
25 }
26 bool MillerTest(ull n,int a,ull d,int s)
27 {
28     ull x=ModPow(a,d,n);
29     if(x==1) return true;
30     for(int i=0;i<s;i++){
31         if(x==n-1) return true;
32         x=ModMul(x,x,n);
33     }
34     return false;
35 }
36 bool MillerRabin(ull n)
37 {
38     if(n<=1) return false;
39     if(n%2==0) return n==2;
40     ull d=n-1; int s=0;

```



```

41 while(d%2==0)
42     d/=2,s++;
43 //int ws[]={2,7,61,-1}; // n<4759123141
44 int ws[]={2,3,5,7,11,13,17,19,23,-1}; // n<3825123056546413051 (conjectured)
45 for(int i=0;~ws[i] && ws[i]<n;i++)
46     if(!MillerTest(n,ws[i],d,s))
47         return false;
48 return true;
49 }

```

8.6 Gauss-Jordan の消去法

解が一意でないとき false を返す．計算量： $O(N^3)$

```

1 // Verify: A0J 1328, Z0J 3645
2
3 bool GaussJordan(const vvd& _a,const vd& b,vd& x)
4 {
5     int n=_a.size();
6     vvd a(n,vd(n+1));
7     rep(i,n){
8         copy(all(_a[i]),begin(a[i]));
9         a[i][n]=b[i];
10    }
11
12    rep(i,n){
13        int p=i;
14        repi(j,i+1,n) if(abs(a[p][i])<abs(a[j][i])) p=j;
15        if(abs(a[p][i])<EPS) return false;
16        swap(a[i],a[p]);
17        peri(j,i,n+1) a[i][j]/=a[i][i];
18        rep(j,n) if(j!=i) peri(k,i,n+1) a[j][k]-=a[j][i]*a[i][k];
19    }
20
21    rep(i,n) x[i]=a[i][n];
22    return true;
23 }

```

9 幾何

9.1 基本要素

```

1 const double PI=acos(-1);
2
3 int Signum(double x){
4     return x<-EPS?-1:x>EPS?1:0;
5 }
6
7 struct Point{
8     double x,y;
9     Point(){}
10    Point(double x,double y):x(x),y(y){}
11    Point& operator+=(Point p){
12        x+=p.x,y+=p.y;
13        return *this;
14    }
15    Point& operator-=(Point p){
16        x-=p.x,y-=p.y;
17        return *this;
18    }
19    Point& operator*=(double c){
20        x*=c,y*=c;
21        return *this;

```

```

22    }
23    Point& operator/=(double c){
24        x/=c,y/=c;
25        return *this;
26    }
27 };
28 Point operator+(Point a,Point b){
29     return a+b;
30 }
31 Point operator-(Point a,Point b){
32     return a-b;
33 }
34 Point operator*(Point a,double c){
35     return a*=c;
36 }
37 Point operator*(double c,Point a){
38     return a*=c;
39 }
40 Point operator/(Point a,double c){
41     return a/=c;
42 }
43 bool operator==(Point a,Point b){
44     return abs(a.x-b.x)<EPS && abs(a.y-b.y)<EPS;
45 }
46 bool operator!=(Point a,Point b){
47     return !(a==b);
48 }
49
50 double Abs(Point p){
51     return sqrt(p.x*p.x+p.y*p.y);
52 }
53 double Abs2(Point p){
54     return p.x*p.x+p.y*p.y;
55 }
56 double Arg(Point p){
57     return atan2(p.y,p.x);
58 }
59 double Dot(Point a,Point b){
60     return a.x*b.x+a.y*b.y;
61 }
62 double Cross(Point a,Point b){
63     return a.x*b.y-a.y*b.x;
64 }
65 Point Rot(Point p,double t){
66     return Point(cos(t)*p.x-sin(t)*p.y,sin(t)*p.x+cos(t)*p.y);
67 }
68
69 struct Line{
70     Point pos,dir;
71     Line(){}
72     Line(Point p,Point d):pos(p),dir(d){}
73     Line(double px,double py,double dx,double dy):pos(px,py),dir(dx,dy){}
74 };
75
76 Point Proj(Line l,Point p){
77     Point a=p-l.pos,b=l.dir;
78     return l.pos+Dot(a,b)/Abs2(b)*b;
79 }
80
81 struct Segment{
82     Point pos,dir;
83     Segment(){}
84     Segment(Point p,Point d):pos(p),dir(d){}
85     Segment(double px,double py,double dx,double dy):pos(px,py),dir(dx,dy){}
86     explicit Segment(Line l):pos(l.pos),dir(l.dir){}

```



```

87     explicit operator Line()const{return Line(pos,dir);}
88 };
89
90 struct Circle{
91     Point center;
92     double radius;
93     Circle(){
94         Circle(Point c,double r):center(c),radius(r){}
95         Circle(double x,double y,double r):center(x,y),radius(r){}
96     };
97     bool operator==(Circle a,Circle b){
98         return a.center==b.center && abs(a.radius-b.radius)<EPS;
99     }
100    bool operator!=(Circle a,Circle b){
101        return !(a==b);
102    }
103
104    // stream
105    ostream& operator<<(ostream& os,const Point& p){
106        return os<<'('<<p.x<<','<<p.y<<')';
107    }
108    ostream& operator<<(ostream& os,const Line& l){
109        return os<<'('<<l.pos<<','<<l.dir<<')';
110    }
111    ostream& operator<<(ostream& os,const Circle& c){
112        return os<<'('<<c.center.x<<','<<c.center.y<<','<<c.radius<<')';
113    }
114
115    // comparer
116    struct LessX{
117        bool operator()(Point a,Point b){
118            return abs(a.x-b.x)>EPS?a.x<b.x-EPS:a.y<b.y-EPS;
119        }
120    };

```

9.2 回転方向

```

1  int CCW(Point a,Point b,Point c){
2      b-=a,c-=a;
3      if(int sign=Signum(Cross(b,c)))
4          return sign; // 1:ccw,-1:cw
5      if(Dot(b,c)<-EPS)
6          return -2; // c-a-b
7      if(Abs2(b)<Abs2(c)-EPS)
8          return 2; // a-b-c
9      return 0; // a-c-b (inclusive)
10 }

```

9.3 面積

```

1  double Area(const vector<Point>& ps){
2      double res=0;
3      rep1(i,2,ps.size())
4          res+=Cross(ps[i-1]-ps[0],ps[i]-ps[0])/2;
5      return res;
6  }

```

9.4 交差判定

```

1  bool IntersectLL(Line a,Line b){
2      return abs(Cross(a.dir,b.dir))>EPS || abs(Cross(a.dir,b.pos-a.pos))<EPS;
3  }
4  bool IntersectLS(Line l,Segment s){
5      Point a=s.pos-l.pos,b=s.pos+s.dir-l.pos;

```

```

6      return Signum(Cross(l.dir,a))*Signum(Cross(l.dir,b))<=0;
7  }
8  bool IntersectSS(Segment a,Segment b){
9      int c1=CCW(a.pos,a.pos+a.dir,b.pos),c2=CCW(a.pos,a.pos+a.dir,b.pos+b.dir);
10     int c3=CCW(b.pos,b.pos+b.dir,a.pos),c4=CCW(b.pos,b.pos+b.dir,a.pos+a.dir);
11     return c1*c2<=0 && c3*c4<=0;
12 }
13 bool IntersectSP(Segment s,Point p){
14     return CCW(s.pos,s.pos+s.dir,p)==0;
15 }
16 // 交点の個数を返す
17 int IntersectCC(Circle c1,Circle c2){
18     if(c1==c2)
19         return INFY;
20     double r1=c1.radius,r2=c2.radius,d=Abs(c1.center-c2.center);
21     if(d>r1+r2+EPS || d<abs(r1-r2)-EPS)
22         return 0;
23     if(abs(d-(r1+r2))<EPS || abs(d-abs(r1-r2))<EPS)
24         return 1;
25     return 2;
26 }
27
28 // aとbは必ず交差していること
29 // 同一直線上にある場合, a.posを返す
30 Point InterPointLL(Line a,Line b){
31     if(abs(Cross(a.dir,b.dir))<EPS) return a.pos;
32     return a.pos+Cross(b.pos-a.pos,b.dir)/Cross(a.dir,b.dir)*a.dir;
33 }
34 // 同一直線上にある場合, s.posを返す
35 Point InterPointLS(Line l,Segment s){
36     return InterPointLL(Line(s),l);
37 }
38 // 同一直線上にある場合, 両端点を順に試し最初に交点であると判定されたものを返す
39 Point InterPointSS(Segment a,Segment b){
40     if(abs(Cross(a.dir,b.dir))<EPS){
41         if(IntersectSP(b,a.pos)) return a.pos;
42         if(IntersectSP(b,a.pos+a.dir)) return a.pos+a.dir;
43         if(IntersectSP(a,b.pos)) return b.pos;
44         if(IntersectSP(a,b.pos+b.dir)) return b.pos+b.dir;
45     }
46     return InterPointLL(Line(a),Line(b));
47 }
48
49 // c1とc2は必ず2点で交わること
50 pair<Point,Point> InterPointCC(Circle c1,Circle c2){
51     Point p1=c1.center,p2=c2.center;
52     double r1=c1.radius,r2=c2.radius;
53     double d=Abs(p1-p2);
54     double a=(d*d+r1*r1-r2*r2)/(2*d);
55     double s=sqrt(r1*r1-a*a);
56     return mp(p1+a/d*(p2-p1)+s*Rot((p2-p1)/d,PI/2),
57             p1+a/d*(p2-p1)-s*Rot((p2-p1)/d,PI/2));
58 }

```

9.5 距離

```

1  double DistLP(Line l,Point p){
2      return Abs(Proj(l,p)-p);
3  }
4  double DistSP(Segment s,Point p){
5      int ccw=CCW(s.pos,s.pos+s.dir,Proj(Line(s),p));
6      if(ccw== -2) return Abs(s.pos-p);
7      if(ccw== 2) return Abs(s.pos+s.dir-p);
8      return DistLP(Line(s),p);
9  }

```

```

10 double DistLS(Line l,Segment s){
11     if(IntersectLS(l,s)) return 0;
12     return min(DistLP(l,s.pos),DistLP(l,s.pos+s.dir));
13 }
14 double DistSS(Segment a,Segment b){
15     if(IntersectSS(a,b)) return 0;
16     double d1=min(DistSP(a,b.pos),DistSP(a,b.pos+b.dir));
17     double d2=min(DistSP(b,a.pos),DistSP(b,a.pos+a.dir));
18     return min(d1,d2);
19 }

```

9.6 円の接線

```

1 vector<Line> Intangent(Circle c1,Circle c2)
2 {
3     double r1=c1.radius,r2=c2.radius;
4     double d=Abs(c1.center-c2.center);
5     vector<Line> res;
6     if(d>r1+r2+EPS){
7         double t=acos((r1+r2)/d);
8         rep(i,2){
9             Point p1=c1.center+Rot(r1/d*(c2.center-c1.center),(i?1:-1)*t);
10            Point p2=c2.center+Rot(r2/d*(c1.center-c2.center),(i?1:-1)*t);
11            res.push_back(Line(p1,p2-p1));
12        }
13    }
14    else if(d>r1+r2-EPS){
15        Point p=c1.center+r1/d*(c2.center-c1.center);
16        res.push_back(Line(p,Rot(p-c1.center,PI/2)));
17    }
18    return res;
19 }
20 vector<Line> Extangent(Circle c1,Circle c2)
21 {
22     double r1=c1.radius,r2=c2.radius;
23     double d=Abs(c1.center-c2.center);
24     vector<Line> res;
25     if(d>abs(r2-r1)+EPS){
26         double t=acos((r1-r2)/d);
27         rep(i,2){
28             Point p1=c1.center+Rot(r1/d*(c2.center-c1.center),(i?1:-1)*t);
29             Point p2=c2.center+Rot(r2/d*(c2.center-c1.center),(i?1:-1)*t);
30             res.push_back(Line(p1,p2-p1));
31         }
32    }
33    else if(d>abs(r2-r1)-EPS){
34        Point p=c1.center+r1/d*(c2.center-c1.center);
35        res.push_back(Line(p,Rot(p-c1.center,PI/2)));
36    }
37    return res;
38 }

```

9.7 凸包

計算量： $O(N \log N)$

```

1 // 連続する3点が同一直線上にある場合はfalseを返す
2 bool IsConvex(const vector<Point>& ps){
3     int n=ps.size(),res=true;
4     rep(i,n) res&=CCW(ps[i],ps[(i+1)%n],ps[(i+2)%n])!=1;
5     return res;
6 }
7
8 // 同一直線上の点はとらない .

```

```

9 // もしとりたければ, while(n>=2 && CCW(cs[n-2],cs[n-1],ps[i])<=0)と書き換える .
10 vector<Point> ConvexHull(vector<Point> ps){
11     if(ps.size()==1) return ps;
12     sort(all(ps),LessX());
13     vector<Point> res;
14     rep(_,2){
15         int n=0;
16         vector<Point> half(ps.size());
17         rep(i,ps.size()){
18             while(n>=2 && CCW(half[n-2],half[n-1],ps[i])!=1)
19                 n--;
20             half[n++]=ps[i];
21         }
22         res.insert(res.end(),half.begin(),half.begin()+n-1);
23         reverse(all(ps));
24     }
25     return res;
26 }

```

9.8 凸多角形の包含判定

```

1 // 凸多角形psの内部に点pがあるときtrue(境界を含む)
2 // ps.size()==0のときtrue
3 // ps.size()==1のとき, ps[0]とpが一致していればtrue
4 // ps.size()==2のとき, ps[0],ps[1]上にpがあればtrue
5 bool CoverGP(const vector<Point>& ps,Point p){
6     rep(i,ps.size()){
7         int ccw=CCW(ps[i],ps[(i+1)%ps.size()],p);
8         if(!(ccw==1 || ccw==0))
9             return false;
10    }
11    return true;
12 }
13 // 凸多角形psが凸多角形qsを内部にもつときtrue(境界を含む)
14 // ps.size()==0またはqs.size()==0のときtrue
15 bool CoverGG(const vector<Point>& ps,vector<Point>& qs){
16     rep(i,qs.size())
17         if(!CoverGP(ps,qs[i]))
18             return false;
19     return true;
20 }

```

9.9 凸多角形の切断

```

1 vector<Point> ConvexCut(const vector<Point>& ps,Line l){
2     int n=ps.size();
3     vector<Point> res;
4     rep(i,n){
5         int c1=CCW(l.pos,l.pos+l.dir,ps[i]);
6         int c2=CCW(l.pos,l.pos+l.dir,ps[(i+1)%n]);
7         if(c1!=1)
8             res.push_back(ps[i]);
9         if(c1*c2==1)
10            res.push_back(InterPointLS(l,Segment(ps[i],ps[(i+1)%n]-ps[i])));
11    }
12    return res;
13 }

```

9.10 線分アレンジメント

同一直線上の推移的な辺を省略している．計算量： $O(N^3)$

```

1 // オーバーラップする線分は取り除いておくこと .
2 struct Edge{

```

```

3   int src,dst;
4   double weight;
5   Edge(){}
6   Edge(int s,int d,double w):src(s),dst(d),weight(w){}
7   bool operator<(const Edge& e) const{return Signum(weight-e.weight)<0;}
8   bool operator>(const Edge& e) const{return Signum(weight-e.weight)>0;}
9 };
10 void SegmentArrangement(const vector<Segment>& ss,Graph& g,vector<Point>& ps){
11     rep(i,ss.size()){
12         ps.push_back(ss[i].pos);
13         ps.push_back(ss[i].pos+ss[i].dir);
14         repi(j,i+1,ss.size()) if(IntersectSS(ss[i],ss[j]))
15             ps.push_back(InterPointSS(ss[i],ss[j]));
16     }
17     sort(all(ps),LessX());
18     ps.erase(unique(all(ps)),ps.end());
19
20     g.resize(ps.size());
21     rep(i,ss.size()){
22         vector<pair<double,int> > ds;
23         rep(j,ps.size()) if(IntersectSP(ss[i],ps[j]))
24             ds.push_back(mp(Abs(ps[j]-ss[i].pos),j));
25         sort(all(ds));
26         rep(j,ds.size()-1){
27             int u=ds[j].second,v=ds[j+1].second;
28             double w=ds[j+1].first-ds[j].first;
29             g[u].push_back(Edge(u,v,w));
30             g[v].push_back(Edge(v,u,w));
31         }
32     }
33 }

```

10 その他

10.1 ビット演算

clz, clzll, ctz, ctzll は, $x = 0$ のときビット長を返す.

```

1 // population count
2 inline int popcount(uint x)
3 {
4     x=(x&0x55555555)+(x>>1&0x55555555);
5     x=(x&0x33333333)+(x>>2&0x33333333);
6     x=(x&0xf0f0f0f0)+(x>>4&0xf0f0f0f0);
7     x=(x&0x00ff00ff)+(x>>8&0x00ff00ff);
8     return (x&0x0000ffff)+(x>>16&0x0000ffff);
9 }
10 inline int popcountll(ull x)
11 {
12     return popcount(x)+popcount(x>>32);
13 }
14
15 // count leading zero
16 inline int clz(uint x)
17 {
18     int i=0;
19     if(!(x&0xffff0000)) i+=16,x<=16;
20     if(!(x&0xff000000)) i+=8,x<=8;
21     if(!(x&0xf0000000)) i+=4,x<=4;
22     if(!(x&0xc0000000)) i+=2,x<=2;
23     if(!(x&0x80000000)) i+=1,x<=1;
24     return i+!x;
25 }
26 inline int clzll(ull x)

```

```

27 {
28     int y=clz(x>>32);
29     return y==32?y+clz(x):y;
30 }
31
32 // count trailing zero
33 inline int ctz(uint x)
34 {
35     int i=0;
36     if(!(x&0x0000ffff)) i+=16,x>=16;
37     if(!(x&0x000000ff)) i+=8,x>=8;
38     if(!(x&0x0000000f)) i+=4,x>=4;
39     if(!(x&0x00000003)) i+=2,x>=2;
40     if(!(x&0x00000001)) i+=1,x>=1;
41     return i+!x;
42 }
43 inline int ctzll(ull x)
44 {
45     int y=ctz(x);
46     return y==32?y+ctz(x>>32):y;
47 }
48
49 // find first set
50 inline int ffs(uint x)
51 {
52     return x?clz(x)+1:0;
53 }
54 inline int ffs(ull x)
55 {
56     return x?clzll(x)+1:0;
57 }
58
59 // bitwise reverse
60 inline uint bitrev(uint x)
61 {
62     x=(x&0xaaaaaaaa)>>1|(x&0x55555555)<<1;
63     x=(x&0xcccccccc)>>2|(x&0x33333333)<<2;
64     x=(x&0xf0f0f0f0)>>4|(x&0x0f0f0f0f)<<4;
65     x=(x&0xff00ff00)>>8|(x&0x00ff00ff)<<8;
66     return (x&0xffff0000)>>16|(x&0x0000ffff)<<16;
67 }
68 inline ull bitrev(ull x)
69 {
70     return bitrev(x>>32)|(ull)bitrev(x)<<32;
71 }
72
73 // 非零部分集合を列挙
74 for(int s=b;s; s=(s-1)&b){
75     ...
76 }
77
78 // {0,1,...,n-1}の, サイズkの部分集合を列挙
79 // 最終行を s|=(1z>>ffs(1o))-1 とすれば除算をなくせる
80 // 注意: 0<k<=n でなければならない
81 for(int s=(1<k)-1;!(s&1<n);){
82     /* ... */
83     int lo=s&-s,lz=(s+lo)&~s;
84     s=(s|lz)&~(lz-1);
85     s|=(1z/lo/2-1);
86 }

```

10.2 ハッシュ関数

ユーザ定義型を `unordered_set` や `unordered_map` のキーとして使うためには、`hash` の特殊化を書く必要がある。以下は FNV-1a による実装例。

```
1 namespace std{
2     template<>
3     struct hash<tuple<int,int>>{
4         size_t operator()(const tuple<int,int>& x)const{
5             const char* p=(const char*)&x;
6             size_t res=2166136261;
7             rep(i,sizeof(x)) (res^=*p++)*=16777619;
8             return res;
9         }
10    };
11 }
```

10.3 Fairfield の公式

西暦 1 年 1 月 0 日からの経過日数を求める。(経過日数) $\equiv 0 \pmod{7}$ のとき日曜日。

```
1 int Fairfield(int y,int m,int d)
2 {
3     if(m<=2) y--,m+=12;
4     return 365*y+y/4-y/100+y/400+153*(m+1)/5+d-428;
5 }
```

10.4 さいころ

```
1 struct Dice{
2     int face[6]; // top,bottom,left,right,front,back
3     void Rotate(int a,int b,int c,int d){
4         int temp=face[a];
5         face[a]=face[b];
6         face[b]=face[c];
7         face[c]=face[d];
8         face[d]=temp;
9     }
10    void TurnF(int n){rep(i,n)Rotate(0,2,1,3);} // x軸を回転軸として右回りに90n度回転
11    void TurnR(int n){rep(i,n)Rotate(0,4,1,5);} // y軸を回転軸として右回りに90n度回転
12    void TurnU(int n){rep(i,n)Rotate(4,3,5,2);} // z軸を回転軸として右回りに90n度回転
13    void TurnF(){TurnF(1);}
14    void TurnR(){TurnR(1);}
15    void TurnU(){TurnU(1);}
16 };
```