Contents

1 Basic 1
- 1.1 Default code . . . 1
- 1.2 Misc 1
- 1.3 Fast read & write 1
- 1.4 Sort cmp 2
- 1.5 Discretization . . 2
- 1.6 Custom unordered_map 2
- 1.7 __int128 read . . 2
- 1.8 字典序 a 嚴格小於 b 2
- 1.9 生成 n 位數的二進制組合 2
- 1.10 Radom 2

2 對拍 2
- 2.1 run.bat 2
- 2.2 run.sh 2

3 Flow & Matching 2
- 3.1 Dicnic 2
- 3.2 最大流最小花費 . . 3
- 3.3 匈牙利匹配 3
- 3.4 KM 4

4 Graph 4
- 4.1 Dijkstra 4
- 4.2 Bellman-Ford . . . 4
- 4.3 SPFA 4
- 4.4 Floyd-Warshall . . 4
- 4.5 歐拉路徑 5
- 4.6 BCC 5
- 4.7 SCC 5
- 4.8 2SAT 6
- 4.9 MaximalClique . . 6
- 4.10 MaximumClique . . 6
- 4.11 Minimum Mean Cycle 6
- 4.12 Dominator Tree . . 7
- 4.13 ManhattanMST . . . 7

5 DP 8
- 5.1 數位 DP 8
- 5.2 SOS DP 8

6 Math 8
- 6.1 Formulas 8
- 6.2 llladdmul 8
- 6.3 Primes 8
- 6.4 Coprime (互質 Pair) 8
- 6.5 Quick Pow 8
- 6.6 Mat quick Pow . . 8
- 6.7 Primes Table . . . 9
- 6.8 Phi 函數 9
- 6.9 Factor Table . . . 9
- 6.10 卡塔蘭數 9
- 6.11 Miller Rabin . . . 9
- 6.12 PollarRho 9
- 6.13 PrimeFactorO(logn) 9
- 6.14 O(1)mul 10
- 6.15 Josephus Problem . 10
- 6.16 Harmonic Sum . . . 10
- 6.17 FFT 10

7 Data Structure 10
- 7.1 BIT 10
- 7.2 BIT 二維 11
- 7.3 並查集 11
- 7.4 稀疏表 O(1) 區間最大最小值 11
- 7.5 Segment Tree . . . 11
- 7.6 動態開點線段數 . . 11
- 7.7 動態開點線段數 2D . 12
- 7.8 持久化線段樹 . . . 13
- 7.9 Time Segment Tree 13
- 7.10 Treap 13
- 7.11 PBDS 14

8 String 14
- 8.1 SA 14
- 8.2 KMP 15
- 8.3 Single Hash . . . 15
- 8.4 Double Hash . . . 15
- 8.5 Trie 16
- 8.6 Z value 16
- 8.7 MinRotation . . . 16
- 8.8 Manacher 馬拉車回文 16
- 8.9 PalTree 回文樹 . . 16
- 8.10 DistinctSubsequence 17

9 Tree 17
- 9.1 LCA 17
- 9.2 TreeHash 17
- 9.3 輕重鏈剖分 17

10 Geometry 18
- 10.1 2D Definition . . 18
- 10.2 Line Definition . 18
- 10.3 Basic 18
- 10.4 PolygonArea . . . 19
- 10.5 IsPointInPolygon . 19
- 10.6 ConvexHull 19
- 10.7 MinkowskiSum . . . 19
- 10.8 Polygon Shortest Distance 19
- 10.9 ConvexHullTrick . 19
- 10.10 Polar Sort 20
- 10.11 PickTheorm 20
- 10.12 ShortestPair . . . 20
- 10.13 FarthestPair . . . 20
- 10.14 幾何中位數 20
- 10.15 矩陣掃描線 21
- 10.16 Polygon Circle intersection area . 21
- 10.17 Tangent line of two circles 21
- 10.18 Circle intersection Point 21
- 10.19 CircleCover . . . 22
- 10.20 半平面交 22
- 10.21 PolygonUnion . . . 23
- 10.22 PolygonCover . . . 23

11 特殊題目 23
- 11.1 包含子字串計數 . . 23
- 11.2 三維偏序 24

12 Python 24
- 12.1 時間日期 Datetime . 24
- 12.2 Decimal 24
- 12.3 Fraction 24
- 12.4 正則表達式 re . . . 24
- 12.5 Misc 25

1 Basic

1.1 Default code

```cpp
#include<bits/stdc++.h>
#include<chrono> // for timing
#pragma GCC optimize("O3,unroll-loops")
#pragma target optimize("avx2,bmi,bmi2,lzcnt,popcnt")
#define IO ios_base::sync_with_stdio(0);cin.tie(0);cout
    .tie(0);
#define pii pair<int,int>
#define ft first
#define sd second
#define int long long
#define ld long double
#define PI acos(-1)
#define SZ(x) (int)x.size()
#define all(v) (v).begin(), (v).end()
#define  _for(i,a,b) for(int i=(a);i<(b);++i)
using namespace std;
template<typename T>
ostream& operator<<(ostream& os,const vector<T>& vn){
  for(int i=0;i<vn.size();++i)os<<vn[i]<<" ";
  return os;
}
template<typename T>
ostream& operator<<(ostream& os,const set<T>& vn){
  for(typename set<T>::iterator it=vn.begin();it!=vn.
      end();++it)os<<*it<<" ";
  return os;
}
mt19937 mt(hash<string>()("Mashu_AC_Please")); //mt();
// mt19937 mt(chrono::steady_clock::now().
    time_since_epoch().count());
// g++ a.cpp -Wall -Wshadow -fsanitize=undefined -o a.
    exe
// ./a.exe
const int MXN=2e5+5;
const int INF=INT_MAX;
void sol() {}
signed main() {
    // auto start=chrono::high_resolution_clock::now();
    // #ifdef LOCAL
    // freopen("input.txt","r",stdin);
    // freopen("output.txt","w",stdout);
    // #endif
    IO
    int t=1;
    // cin>>t;
    while(t--) {sol();}
    // auto stop = chrono::high_resolution_clock::now()
        ;
    // auto duration = chrono::duration_cast<chrono::
        milliseconds>(stop - start);
    // cerr<<"Time:"<<duration.count()<<" ms\n";
}
```

1.2 Misc

```cpp
iota(vec.begin(),vec.end(),1);// 產生1~size的整數列
stoi(s.begin(),s.end(),k);// 法1,字串轉成k進位int
string s;cin>>s;
int x=stoi(s,0,2); // 法2,2可以改其他進位
int bbb = bitset<10>(bb).to_ulong();//二進位轉十進位
__builtin_popcountll // 二進位有幾個1
__builtin_clzll // 左起第一個1前0的個數
__builtin_parityll // 1的個數的奇偶性
__builtin_mul_overflow(a,b,&res) // a*b是否溢位

// double 轉整數 請加 int b=round(a)
// 或是 int b =floor(a+0.5) (floor向下取整)
```

1.3 Fast read & write

```cpp
inline int read() {
    char c = getchar(); int x = 0, f = 1;
```

```
    while(c < '0' || c > '9') {if(c == '-') f = -1; c =
        getchar();}
    while(c >= '0' && c <= '9') x = x * 10 + c - '0', c
        = getchar();
    return x * f;
}
inline void write(int x){
    if(x<0) putchar('-'),x=-x;
    if(x>9) write(x/10);
    putchar(x%10+'0');
}
```

1.4 Sort cmp

```
struct cmp{inline bool operator()(const int a,const int
    b){return a<b;}};//common use
auto cmp=[](vector<int> a, vector<int> b) {return a[1]<
    b[1];};//for set use
set<vector<int>, decltype(cmp)> prepare, done;
```

1.5 Discretization

```
vector<int> vec;
sort(vec.begin(),vec.end());
vec.resize(unique(vec.begin(),vec.end())-vec.begin());
for(int i=0;i<n;++i){//+1是讓 index是1到N 可以不要
    arr[i]=lower_bound(vec.begin(),vec.end(),ll[i])-vec
        .begin()+1;
}
```

1.6 Custom unordered_map

```
struct Type{
    int x;
    string y;
    bool operator==(const Type &other) const {
        return (x == other.x && y == other.y);
    }
};
struct hashs{
    size_t operator()(const Type &o) const {
        return ((hash<int>()(o.x)^(hash<string>()(o.y)
            <<1))>>1);
    }
};
//unordered_map<Type,int,hashs> map;
```

1.7 __int128 read

```
//  __int128_t p;
//  lll n=qr(p);
#define lll __int128
template<class type_name> inline type_name qr(type_name
    sample)
{
    type_name ret=0,sgn=1;
    char cur=getchar();
    while(!isdigit(cur))
        sgn=(cur=='-'?-1:1),cur=getchar();
    while(isdigit(cur))
        ret=(ret<<1)+(ret<<3)+cur-'0',cur=getchar();
    return sgn==-1?-ret:ret;
}
inline void print(__int128 x){
    if(x < 0){
        putchar('-');
        x = -x;
    }
    if(x > 9)
        print(x / 10);
    putchar(x % 10 + '0');
}
```

1.8 字典序 a 嚴格小於 b

```
template<class T> //字典序a嚴格小於b
bool lexicographicallySmaller(const vector<T> &a,const
    vector<T> &b){
    int n=a.size();
    int m=b.size();
    int i;
    for(int i=0;i<n && i<m;++i){
        if(a[i]<b[i])return true;
        else if(b[i]<a[i])return false;
    }
    return (i==n && i<m);
}
```

1.9 生成 n 位數的二進制組合

```
//產生1~n位數的二進位組合
vector<int> comb;
void genBinComb(int n){
    for(int i=1;i<(1<<n);++i){
        string s = bitset<5>(i).to_string();//bitset大
            小要記得設
        comb.push_back(stoi(s));
    }
}
```

1.10 Radom

```
mt19937 gen(0x5EED);
int randint(int lb, int ub)
{ return uniform_int_distribution<int>(lb, ub)(gen); }
```

2 對拍

2.1 run.bat

```
@echo off
g++ ac.cpp -o ac.exe
g++ wa.cpp -o wa.exe
g++ gen1.cpp -o gen.exe

:loop
   echo %%x
   gen.exe > input
   ac.exe < input > ac
   wa.exe < input > wa
   fc ac wa
if not errorlevel 1 goto loop
```

2.2 run.sh

```
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

3 Flow & Matching

3.1 Dicnic

```
// flow.init(n,s,t):有n個點(0~n-1)，起點s終點t
// flow.add_edge(u,v,f):建一條邊，從u點到v點流量為f
// flow.solve():回傳網路最大流答案
//時間複雜度: O(V^2*E)
struct Dinic{
    struct Edge{ int v,f,re; };
    int n,s,t,level[MXN];
    vector<Edge> E[MXN];
```

```
    void init(int _n, int _s, int _t){
        n = _n; s = _s; t = _t;
        for (int i=0; i<n; i++) E[i].clear();
    }
    void add_edge(int u, int v, int f){
        E[u].push_back({v,f,(int)(E[v]).size()});
        E[v].push_back({u,0,(int)(E[u]).size()-1});
    }
    bool BFS(){
        for (int i=0; i<n; i++) level[i] = -1;
        queue<int> que;
        que.push(s);
        level[s] = 0;
        while (!que.empty()){
            int u = que.front(); que.pop();
            for (auto it : E[u]){
            if (it.f > 0 && level[it.v] == -1){
                level[it.v] = level[u]+1;
                que.push(it.v);
        } } }
        return level[t] != -1;
    }
    int DFS(int u, int nf){
        if (u == t) return nf;
        int res = 0;
        for (auto &it : E[u]){
            if (it.f > 0 && level[it.v] == level[u]+1){
            int tf = DFS(it.v, min(nf,it.f));
            res += tf; nf -= tf; it.f -= tf;
            E[it.v][it.re].f += tf;
            if (nf == 0) return res;
        } }
        if (!res) level[u] = -1;
        return res;
    }
    int solve(int res=0){
    while ( BFS() )
        res += DFS(s,2147483647);
    return res;
} }flow;
```

3.2 最大流最小花費

```
//最大流量上的最小花費
//最大流量優先，相同才是找最小花費，複雜度O(V^2*E^2)
// flow.init(n,s,t):有n個點(0~n-1)，起點s終點t
// flow.add_edge(u,v,f,c):建一條邊，從u點到v點流量為f，
    每一單位流量的花費為c
// flow.solve():回傳一個pair(maxFlow,minCost)
// 限制: 圖不能有負環
// 網路最大流的add_edge(u,v,f)可以無痛轉成最大流量上的
    最小花費add_edge(u,v,1,f)即建立一條從u到v的邊流量為
    1，單位流量花費為f
//O(V^2 E^2)
#define ll long long
struct zkwflow{
    static const int maxN=20000;
    struct Edge{ int v,f,re; ll w;};
    int n,s,t,ptr[maxN]; bool vis[maxN]; ll dis[maxN];
    vector<Edge> E[maxN];
    void init(int _n,int _s,int _t){
        n=_n,s=_s,t=_t;
        for(int i=0;i<n;i++) E[i].clear();
    }
    void add_edge(int u,int v,int f,ll w){
        E[u].push_back({v,f,(int)E[v].size(),w});
        E[v].push_back({u,0,(int)E[u].size()-1,-w});
    }
    bool SPFA() {
        fill_n(dis, n, LLONG_MAX);
        fill_n(vis, n, false);
        queue<int> q;
        q.push(s); dis[s]=0;
        while(!q.empty()) {
            int u = q.front(); q.pop();
            vis[u] = false;
            for(auto &it: E[u]){
                if(it.f>0 && dis[it.v]>dis[u]+it.w){
                    dis[it.v] = dis[u]+it.w;
                    if(!vis[it.v]) {vis[it.v] = true; q
                        .push(it.v);}
                }
            }
        }
        if(dis[t]==LLONG_MAX) return false;
        // 不管流量是多少，花費不能是正數時加上這行 (最
            小花費可行流)
        // if(dis[t] >= 0) return false;
        return true;
    }
    int DFS(int u, int nf) {
        if(u==t) return nf;
        int res = 0; vis[u] = true;
        for(int &i=ptr[u] ; i<(int)E[u].size() ; i++) {
            auto &it = E[u][i];
            if(it.f>0 && dis[it.v]==dis[u]+it.w && !vis
                [it.v]) {
                int tf = DFS(it.v, min(nf, it.f));
                res += tf;
                nf-=tf;
                it.f-=tf;
                E[it.v][it.re].f += tf;
                if(nf==0) { vis[u]=false; break; }
            }
        }
        return res;
    }
    pair<int,ll> solve(){
        int flow = 0; ll cost = 0;
        while (SPFA()){
            fill_n(ptr, n, 0);
            int f = DFS(s, INT_MAX);
            flow += f;
            cost += dis[t]*f;
        }
        return {flow, cost};
    } // reset: do nothing
} flow;
```

3.3 匈牙利匹配

```
//匈牙利演算法-二分圖最大匹配
//記得每次使用需清空vis數組
//O(nm)
//其中Map為鄰接表(Map[u][v]為u和v是否有連接) S為紀錄這
    個點與誰匹配(S[i]為答案i和誰匹配)
const int M=505, N=505;
bool Map[M][N] = {0};
int S[N];
bool vis[N];
bool dfs(int u){
    for(int i=0;i<N;i++){
        if(Map[u][i]&&!vis[i]){ //有連通且未拜訪
            vis[i]=1; //紀錄是否走過
            if(S[i]==-1||dfs(S[i])){  //紀錄匹配
                S[i]=u;
                return true;  //反轉匹配邊以及未匹配邊
                    的狀態
            }
        }
    }
    return false;
}
//此二分圖為左邊M個點右邊N個點, 跑匈牙利只要跑1~M就可以
    了, (S[右邊的點] -> 左邊的點)
memset(S,-1,sizeof(S));
int ans = 0;
for(int i=0;i<M;i++){
    memset(vis,0,sizeof(vis));
    if(dfs(i)) ans++;
    //跑匈牙利
}
cout<<ans<<"\n";
for(int i=0 ; i<N ;i++) {
    if(S[i]!=-1) cout<<"pair: "<<S[i]<<" "<<i<<"\n";
}
```

3.4 KM

```
//二分圖最大權完美匹配
//二分圖左邊的點都要匹配到右邊的點，且每條邊都有權重，
    求權重最大值，複雜度O(V^3)
// graph.init(n):二分圖左右各n個點
// graph.add_edge(u,v,w):建一條邊，從u點到v點權重為w
// graph.solve():回傳最大權重
struct KM{ // max weight, for min negate the weights
    int n, mx[MXN], my[MXN], pa[MXN];
    ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
    bool vx[MXN], vy[MXN];
    void init(int _n) { // 1-based, N個節點
        n = _n;
        for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0)
            ;
    }
    void add_edge(int x, int y, ll w) {g[x][y] = w;} //
        左邊的集合節點x連邊右邊集合節點y權重為w
    void augment(int y) {
        for(int x, z; y; y = z)
          x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
    }
    void bfs(int st) {
        for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i
            ]=0;
        queue<int> q; q.push(st);
        for(;;) {
            while(q.size()) {
                int x=q.front(); q.pop(); vx[x]=1;
                for(int y=1; y<=n; ++y) if(!vy[y]){
                    ll t = lx[x]+ly[y]-g[x][y];
                    if(t==0){
                        pa[y]=x;
                        if(!my[y]){augment(y);return;}
                        vy[y]=1, q.push(my[y]);
                    }else if(sy[y]>t) pa[y]=x,sy[y]=t;
                }
            }
            ll cut = INF;
            for(int y=1; y<=n; ++y)
                if(!vy[y]&&cut>sy[y]) cut=sy[y];
            for(int j=1; j<=n; ++j){
                if(vx[j]) lx[j] -= cut;
                if(vy[j]) ly[j] += cut;
                else sy[j] -= cut;
            }
            for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y
                ]==0){
                if(!my[y]){augment(y);return;}
                vy[y]=1, q.push(my[y]);
            }
        }
    }
    ll solve(){ // 回傳值為完美匹配下的最大總權重
        fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
        fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
        for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
             // 1-base
          lx[x] = max(lx[x], g[x][y]);
        for(int x=1; x<=n; ++x) bfs(x);
        ll ans = 0;
        for(int y=1; y<=n; ++y) ans += g[my[y]][y];
        return ans;
    }
} graph;
```

4 Graph

4.1 Dijkstra

```
int dis[N],vis[N];
void dijkstra(int s){//O(V^2+E)
    memset(dis,0x3f,sizeof(dis));
    memset(vis,0,sizeof(vis));
    dis[s] = 0;
    priority_queue<pii,vector<pii>,greater<pii>> pq;
    pq.push({0,s});
    while(!pq.empty()){
        auto [nowd,now] = pq.top();
        pq.pop();
        if(vis[now])
            continue;
        vis[now] = 1;
        for(auto [nxt,w]: graph[now]){
            if(dis[nxt] > dis[now] + w){
                dis[nxt] = dis[now] + w;
                pq.push({dis[nxt],nxt});
            }
        }
    }
}
```

4.2 Bellman-Ford

```
//總共m條邊，鬆弛n-1次->O(nm)
//在第n次做一次鬆弛，如果有點被鬆弛到，代表這張圖存在負
    環
for(int j = 0; j < n-1; j++){
    for(int i = 0; i < m; i++){ // 對於所有邊都嘗試鬆弛
        if(dis[ edge[i].to ] > dis[ edge[i].from ] +
            edge[i].weight){
            dis[ edge[i].to ] = dis[ edge[i].from ] +
                edge[i].weight;
        }
    }
}
```

4.3 SPFA

```
#define MXN 200005
struct SPFA{//O(kE) k:小常數
  int n;
  ll inq[MXN], len[MXN];
  vector<ll> dis;
  vector<pair<int, ll>> edge[MXN];
  void init(int _n){
    n = _n;
    dis.clear(); dis.resize(n, 1e18);
    for(int i = 0; i < n; i++){
      edge[i].clear();
      inq[i] = len[i] = 0;
  } }
  void addEdge(int u, int v, ll w){
    edge[u].push_back({v, w});
  }
  vector<ll> solve(int st = 0){
    deque<int> dq; //return {-1} if has negative cycle
    dq.push_back(st); //otherwise return dis from st
    inq[st] = 1; dis[st] = 0;
    while(!dq.empty()){
      int u = dq.front();  dq.pop_front();
      inq[u] = 0;
      for(auto [to, d] : edge[u]){
        if(dis[to] > d+dis[u]){
          dis[to] = d+dis[u];
          len[to] = len[u]+1;
          if(len[to] > n) return {-1};
          if(inq[to]) continue;
          (!dq.empty()&&dis[dq.front()] > dis[to]?
              dq.push_front(to) : dq.push_back(to));
          inq[to] = 1;
    } } }
    return dis;
} }spfa;
```

4.4 Floyd-Warshall

```
for(int k=0;k<n;++k){//O(N^3)
    for(int i=0;i<n;++i){
        for(int j=0;j<n;++j){
            if(graph[i][k]!=MAX&&graph[k][j]!=MAX)//避
                免不連通圖
                graph[i][j]=min(graph[i][j],graph[i][k
                    ]+graph[k][j]);
        }
```

```
    }
}
for(int k=0;k<n;++k){//判斷負環
    for(int i=0;i<n;++i){
        for(int j=0;j<n;++j){
            if(graph[i][k]!=MAX&&graph[k][j]!=MAX&&
                graph[k][k]<0){//避免不連通圖&&負環
                graph[i][j]=-MAX;
            }
        }
    }
}
// if(graph[a][b]==-MAX)
//     cout<<"-Infinity\n";
// else if(graph[a][b]==MAX)
//     cout<<"Impossible\n";
// else
//     cout<<graph[a][b]<<"\n";
```

4.5 歐拉路徑

```
const int MXN = 2e5+7;
struct ola{
    vector<pair<int, int>> edge[MXN];
    int ind[MXN], ru[MXN], use[MXN*3], es=0, n, go;
    void init(int _n) {
        n = _n;
        for(int i=0 ; i<n ; i++) edge[i].clear(), ind[i
            ]=0;
    }
    void add_edge(int a, int b) {
        use[es] = 0;
        edge[a].push_back({b, es});
        edge[b].push_back({a, es++});
        ru[a]++; ru[b]++;
    }
    bool check() {
        int cnt = 0;
        for(int i=0 ; i<n ; i++) {
            if(ru[i]&1) go = i,cnt++;
        }
        if(cnt==0 || cnt==2) return true;
        return false;
    }
    vector<int> ans;
    void dfs(int x) {
        for(int i=ind[x] ; i<edge[x].size() ; i=ind[x])
            {
            int u = edge[x][i].first, pos = edge[x][i].
                second;
            ind[x]++;
            if(!use[pos]) {
                use[pos]=1;
                dfs(u);
            }
        }
        ans.push_back(x);
    }
    vector<int> solve() {
        if(!check()) return {-1};
        ans.clear(); dfs(go);
        if(ans.size()!=es+1) return {-1};
        return ans;
    }
} euler;
```

4.6 BCC

```
//無向圖上，不會產生割點的連通分量稱為點雙連通分量，
    0base
#define PB push_back
#define REP(i, n) for(int i = 0; i < n; i++)
struct BccVertex {
    int n, nScc, step, dfn[MXN], low[MXN];
    vector<int> E[MXN], sccv[MXN];
    int top, stk[MXN];
    void init(int _n) {
        n = _n;
        nScc = step = 0;
        for (int i = 0; i < n; i++)
            E[i].clear();
    }
    void addEdge(int u, int v) {
        E[u].PB(v); E[v].PB(u);
    }
    void DFS(int u, int f) {
        dfn[u] = low[u] = step++;
        stk[top++] = u;
        for (auto v : E[u]) {
            if (v == f) continue;
            if (dfn[v] == -1) {
                DFS(v, u);
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u]) {
                    int z;
                    sccv[nScc].clear();
                    do {
                        z = stk[--top];
                        sccv[nScc].PB(z);
                    } while (z != v);
                    sccv[nScc++].PB(u);
                }
            }
            else low[u] = min(low[u], dfn[v]);
        }
    }
    vector<vector<int>> solve() {//回傳每個點雙聯通分量
        vector<vector<int>> res;
        for (int i = 0; i < n; i++)
            dfn[i] = low[i] = -1;
        for (int i = 0; i < n; i++)
            if (dfn[i] == -1) {
                top = 0;
                DFS(i, i);
            }
        REP(i, nScc) res.PB(sccv[i]);
        return res;
    }
} graph;
```

4.7 SCC

```
//在有向圖裡的任兩點u、v，皆存在至少一條 u 到 v 的路徑
    以及 v 到 u 的路徑
//fill zero 注意多筆測資要改fill
//注意要0base
#define PB push_back
#define FZ(x) memset(x, 0, sizeof(x))
const int MXN = 1e5;
struct Scc {
    int n, nScc, vst[MXN], bln[MXN];//nScc 有幾個強連通
        分量
    vector<int> E[MXN], rE[MXN], vec;
    void init(int _n) {
        n = _n;
        for (int i = 0; i < MXN; i++)
            E[i].clear(), rE[i].clear();
    }
    void addEdge(int u, int v) {
        E[u].PB(v); rE[v].PB(u);
    }
    void DFS(int u) {
        vst[u] = 1;
        for (auto v : E[u])
            if (!vst[v]) DFS(v);
        vec.PB(u);
    }
    void rDFS(int u) {
        vst[u] = 1;
        bln[u] = nScc;
        for (auto v : rE[u])
            if (!vst[v]) rDFS(v);
    }
    void solve() {
        nScc = 0;
        vec.clear();
        FZ(vst);
        for (int i = 0; i < n; i++)
```

```
            if (!vst[i]) DFS(i);
        reverse(vec.begin(), vec.end());
        FZ(vst);
        for (auto v : vec)
            if (!vst[v]) {rDFS(v); nScc++;}
    }
} scc;
```

4.8 2SAT

```
有N個 boolean 變數$a_1 ⍰ a_N$
ex: 滿足 (¬a1 or a2)and(a2 or a3)and(¬a3 or ¬a4) 的解
* **想法(把2-SAT 轉 SCC)**
把n個boolean值分成true和false兩種節點(共$2n$個節點)
如果有一個條件 (p and q)，則建兩條邊
not p -> q (if p為false 則 q必為true)
not q -> p (if q為false 則 p必為true)
然後跑一次SCC
我們可以知道對於當前變數$a_i$有true和false兩種
* 如果($a_i$和$¬a_i$)在同一個強連通分量裡表示
    (if $a_i$為true 則 $a_i$必為false，因為有一條路徑從
        $a_i$到$¬a_i$)
    (if $a_i$為false 則 $a_i$必為true，因為有一條路徑從
        $¬a_i$到$a_i$)
    很明顯矛盾了...(無解)
* 如果($a_i$和$¬a_i$)**不**在同一個強連通分量裡表示
    如果把SCC縮點成DAG
    則會有$a_i$的強連通分量流到$¬a_i$的強連通分量 or
        $¬a_i$的強連通分量流到$a_i$的強連通分量(其一)
    if (有$a_i$的強連通分量流到$¬a_i$的強連通分量) 則表
        示
       如果 $a_i$為true 則 $a_i$必為false，但
        沒有表示
       ~~如果 $a_i$為false 則 $a_i$必為true~~
       此時把 $a_i$的值設false即可
    ps: 在模板中如果有$a_i$的強連通分量流到$¬a_i$的強連
        通分量則$bln[¬a_i]>bln[a_i]$
```

4.9 MaximalClique

```
//極大團
//對於一張圖選任意的點子集，如果不能在多選一個點使得選
    的點子集為更大的團
#define N 80
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int lnk[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      lnk[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int ans , stk[N], id[N] , di[N] , deg[N];
  Int cans;
  void dfs(int elem_num, Int candi, Int ex){
    if(candi.none()&&ex.none()){
      cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
      ans = elem_num; //cans=1 is in maximal clique
      return;
    }
    int pivot = (candi|ex)._Find_first();
    Int smaller_candi = candi & (~lnk[pivot]);
    while(smaller_candi.count()){
      int nxt = smaller_candi._Find_first();
      candi[nxt] = smaller_candi[nxt] = 0;
      ex[nxt] = 1;
      stk[elem_num] = nxt;
      dfs(elem_num+1,candi&lnk[nxt],ex&lnk[nxt]);
  } }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) lnk[di[i]][di[j]] = 1;
    ans = 1; cans.reset(); cans[0] = 1;
    dfs(0, Int(string(n,'1')), 0);
    return ans;
} }solver;
```

4.10 MaximumClique

```
//最大團:圖上最多可以選幾個點，使選的彼此之間都有連邊
//最大獨立集:圖上最多可以選幾個點，使選的彼此之間都沒有
    連邊
//最大獨立集通常會轉換為用補圖做最大團
//O(1.1888^n)
#define N 111
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int linkto[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      linkto[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int popcount(const Int& val)
  { return val.count(); }
  int lowbit(const Int& val)
  { return val._Find_first(); }
  int ans , stk[N];
  int id[N] , di[N] , deg[N];
  Int cans;
  void maxclique(int elem_num, Int candi){
    if(elem_num > ans){
      ans = elem_num; cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
    }
    int potential = elem_num + popcount(candi);
    if(potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while(smaller_candi.count() && potential > ans){
      int next = lowbit(smaller_candi);
      candi[next] = !candi[next];
      smaller_candi[next] = !smaller_candi[next];
      potential --;
      if(next == pivot || (smaller_candi & linkto[next
          ]).count()){
        stk[elem_num] = next;
        maxclique(elem_num + 1, candi & linkto[next]);
  } } }
  int solve(){//回傳值為最大團的點數量
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) linkto[di[i]][di[j]] = 1;
    Int cand; cand.reset();
    for(int i = 0 ; i < n ; i ++) cand[i] = 1;
    ans = 1;
    cans.reset(); cans[0] = 1;
    maxclique(0, cand);
    return ans;
} }solver;
```

4.11 Minimum Mean Cycle

```
//給定一張有向圖，邊上有權重，要找到一個環其平均權重最小
/* minimum mean cycle O(VE) */
struct MMC{
#define E 101010
#define V 1021
#define inf 1e9
#define eps 1e-6
  struct Edge { int v,u; double c; };
  int n, m, prv[V][V], prve[V][V], vst[V];
  Edge e[E];
  vector<int> edgeID, cycle, rho;
  double d[V][V];
  void init( int _n )
  { n = _n; m = 0; }
  // WARNING: TYPE matters
  //建一條單向邊 (u, v) 權重為 w
  void addEdge( int vi , int ui , double ci )
  { e[ m ++ ] = { vi , ui , ci }; }
  void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
      fill(d[i+1], d[i+1]+n, inf);
      for(int j=0; j<m; j++) {
        int v = e[j].v, u = e[j].u;
        if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
          d[i+1][u] = d[i][v]+e[j].c;
          prv[i+1][u] = v;
          prve[i+1][u] = j;
  } } } }
  double solve(){//回傳值為最小平均權重 (小數)
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
      double avg=-inf;
      for(int k=0; k<n; k++) {
        if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i
            ])/(n-k));
        else avg=max(avg,inf);
      }
      if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    fill(vst,0); edgeID.clear(); cycle.clear(); rho.
        clear();
    for (int i=n; !vst[st]; st=prv[i--][st]) {
      vst[st]++;
      edgeID.PB(prve[i][st]);
      rho.PB(st);
    }
    while (vst[st] != 2) {
      if(rho.empty()) return inf;
      int v = rho.back(); rho.pop_back();
      cycle.PB(v);
      vst[v]++;
    }
    reverse(ALL(edgeID));
    edgeID.resize(SZ(cycle));
    return mmc;
} }mmc;
```

4.12 Dominator Tree

```
// 給一張有向圖，圖上有一個起點 S 可以走到所有點。
// 定義 "支配" 為從起點 S 出發，所有能走到節點 x 的路徑的最後一個必經點
// 最後 idom[i] 為點 i 的支配點
struct DominatorTree{ // O(n+m)
#define REP(i,s,e) for(int i=(s);i<=(e);i++)
#define REPD(i,s,e) for(int i=(s);i>=(e);i--)
  int n , s;
  vector< int > g[ MAXN ] , pred[ MAXN ];
  vector< int > cov[ MAXN ];
  int dfn[ MAXN ] , nfd[ MAXN ] , ts;
  int par[ MAXN ]; //idom[u] s到u的最後一個必經點
  int sdom[ MAXN ] , idom[ MAXN ];
  int mom[ MAXN ] , mn[ MAXN ];
  inline bool cmp( int u , int v )
  { return dfn[ u ] < dfn[ v ]; }
  int eval( int u ){
    if( mom[ u ] == u ) return u;
    int res = eval( mom[ u ] );
    if(cmp( sdom[ mn[ mom[ u ] ] ] , sdom[ mn[ u ] ] ))
      mn[ u ] = mn[ mom[ u ] ];
    return mom[ u ] = res;
  }
  //節點數量，起點編號 1-base
  void init( int _n , int _s ){
    ts = 0; n = _n; s = _s;
    REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
  }
  void addEdge( int u , int v ){
    g[ u ].push_back( v );
    pred[ v ].push_back( u );
  }
  void dfs( int u ){
    ts++;
    dfn[ u ] = ts;
    nfd[ ts ] = u;
    for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
      par[ v ] = u;
      dfs( v );
  } }
  void build(){// 建立支配樹
    REP( i , 1 , n ){
      dfn[ i ] = nfd[ i ] = 0;
      cov[ i ].clear();
      mom[ i ] = mn[ i ] = sdom[ i ] = i;
    }
    dfs( s );
    REPD( i , n , 2 ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      for( int v : pred[ u ] ) if( dfn[ v ] ){
        eval( v );
        if( cmp( sdom[ mn[ v ] ] , sdom[ u ] ) )
          sdom[ u ] = sdom[ mn[ v ] ];
      }
      cov[ sdom[ u ] ].push_back( u );
      mom[ u ] = par[ u ];
      for( int w : cov[ par[ u ] ] ){
        eval( w );
        if( cmp( sdom[ mn[ w ] ] , par[ u ] ) )
          idom[ w ] = mn[ w ];
        else idom[ w ] = par[ u ];
      }
      cov[ par[ u ] ].clear();
    }
    REP( i , 2 , n ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      if( idom[ u ] != sdom[ u ] )
        idom[ u ] = idom[ idom[ u ] ];
} } }domT;
```

4.13 ManhattanMST

```
// 出來的是保證能做出MST的邊 邊是亂的 ~~魔術師~~
// 需要swap(u.first.first,u.first.second) + sort unique
#define ld long long
struct Pt{
    ld x,y;
    Pt(ld x=0,ld y=0):x(x),y(y){}
    Pt operator+(const Pt &a) const {return Pt(x+a.x, y
        +a.y);  }
    Pt operator-(const Pt &a) const {return Pt(x-a.x, y
        -a.y);  }
};
vector<pair<pair<int,int>,int>> ManhattanMST(vector<Pt>
     P) {
    vector<int> id(P.size());
    iota(id.begin(),id.end(), 0);
    vector<pair<pair<int,int>, int>> edg;
    for (int k = 0; k < 4; k++) {
        sort(id.begin(),id.end(), [&](int i, int j) {
                return (P[i] - P[j]).x < (P[j] - P[i]).
                    y;
            });
        map<int, int> sweep;
        for (int i : id) {
```

```
            auto it = sweep.lower_bound(-P[i].y);
            while (it != sweep.end()) {
                int j = it->second;
                Pt d = P[i] - P[j];
                if (d.y > d.x) {
                    break;
                }
                edg.push_back({{i, j},d.x + d.y});
                it = sweep.erase(it);
            }
            sweep[-P[i].y] = i;
        }
        for (Pt &p : P) {
            if (k % 2) {
                p.x = -p.x;
            } else {
                swap(p.x, p.y);
            }
        }
    }
    return edg;
}
```

5 DP

5.1 數位 DP

```
// dp[位數][狀態]
// dp[pos][state]: 定義為目前位數在前導狀態為state的時候的計數
// ex: 求數字沒有出現66的數量 l~r
// -> dp[pos][1] 可表示計算pos個位數在前導出現一個6的計數 -> dp[3][1] 則計算 6XXX
// 模板的pos是反過來的，但不影響(只是用來dp記憶用)

// pos: 目前位數
// state: 前導狀態
// lead: 是否有前導0 (大部分題目不用但有些數字EX:00146如果有影響時要考慮)
// limit: 使否窮舉有被num限制
vector<int> num;
int dp[20][state];
int dfs(int pos, int state, bool lead, bool limit) {
    if(pos==num.size()) {
        //有時要根據不同state回傳情況
        return 1;
    }
    if(limit==false && lead==false && dp[pos][state
        ]!=-1) return dp[pos][state];
    int up = limit?num[pos]:9;
    int ans = 0;
    for(int i=0 ; i<=up ; i++) {
        //有時要考慮那些狀況要continue
        ans += dfs(pos+1, state||(check[i]==2), lead&&i
            ==0, limit&&i==num[pos]);
    }
    if(limit==false && lead==false) dp[pos][state] =
        ans;
    return ans;
}
```

5.2 SOS DP

```
for (int mask = 0; mask < (1 << N); mask++)
    F[mask] = A[mask];

for (int i = 0; i < N; i++)
    for (int mask = 0; mask < (1 << N); mask++)
        if (mask & (1 << i))
            F[mask] += F[mask ^ (1 << i)];
```

6 Math

6.1 Formulas

```
//五次方冪次和 1, 33, 276, 1300, 4425, 12201, 29008, 61776
a(n) = n^2*(n+1)^2*(2*n^2+2*n-1)/12
//四次方冪次和 1, 17, 98, 354, 979, 2275, 4676, 8772, 15333
a(n) = n*(1+n)*(1+2*n)*(-1+3*n+3*n^2)/30
//錯位排列 0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961
dp[1]=0;dp[2]=1;
for(int i=3;i<=n;++i){dp[i]=(i-1)*(dp[i-2]+dp[i-1])%MOD
    ;}
```

6.2 llladdmul

```
#define ll long long
#define lll __int128
ll mypow(lll n,lll k,ll p){
    ll res=1;
    for(;k;k>>=1,n=n*n%p)if(k&1)res=res*n%p;
    return res;
}
ll mul(ll a,ll b,ll mod){
    lll c=a%mod;
    return c*b%mod;
}
ll add(ll x,ll y,ll mod){
    lll c=x;c+=y;
    return c%mod;
}
```

6.3 Primes

```
mashu lucky prime : 91145149
1097774749, 1076767633, 100102021, 999997771
1001010013, 1000512343, 987654361, 999991231
999888733, 98789101, 987777733, 999991921, 1010101333
```

6.4 Coprime (互質 Pair)

```
// dp[i] 為認挑2個數字，最大公因數為i的組合數
const int mxn =  1e6+7;
int cnt[mxn], dp[mxn];
int sol(vector<int> vec) {
    for(int u: vec) cnt[u]++;
    for(int i=mxn-1 ; i>=1 ; i--) {
        int a=0,b=0;
        for(int j=i ; j<mxn ; j+=i) {
            a+=cnt[j]; b+=dp[j];
        }
        dp[i] = (a*(a-1)/2) - b;
    }
    return dp[1];
}
```

6.5 Quick Pow

```
// a^b
const int MOD = 1e9+7;
int qpow(int n, int k,int p) {
    int ret = 1;
    for(;k; k >>= 1, n = n * n % p) if(k & 1) ret = ret
        * n % p;
    return ret;
}
// a^(b^c) = a^(q*(p-1)+r) = a^r so let b^c mod p-1
bc =qpow(b,c,p-1);
ans=qpow(a,bc,p);
```

6.6 Mat quick Pow

```
struct mat{
    long long a[200][200],r,c; // resize
    mat(int _r,int _c){r=_r;c=_c;memset(a,0,sizeof(a))
        ;}
    void build(){for(int i=0;i<r;++i)a[i][i]=1;}
};
mat operator * (mat &x,mat &y){
    mat z(x.r,y.c);
    for(int i=0;i<x.r;++i)for(int j=0;j<x.c;++j)for(int
         k=0;k<y.c;++k)
        z.a[i][j]=(z.a[i][j]+x.a[i][k]*y.a[k][j]%MOD)%
            MOD;
    return z;
}
mat qpow(mat a,int k){
    mat r(a.r,a.r);r.build();while(k){if(k&1)r=r*a;a=a*
        a;k>>=1;}return r;
}
```

6.7 Primes Table

```
int np[MXN];
vector<int> vec;
void sol(){
    np[0]=np[1]=1;
    for(int i=2;i<MXN;++i){
        if(!np[i]){
            for(int j=i;j<MXN;j+=i){
                np[j]=1;
            }
            vec.push_back(i);
        }
    }
}
```

6.8 Phi 函數

```
// 計算小於n的數中與n互質的有幾個
// O(sqrtN)

// a^b mod c = a^(b%phi(c) + phi(c)) mod c    : if b>=
    phi(c)
//           = a^b mod c                      : if b<
    phi(c)
int phi(int n){
    int res = n, a=n;
    for(int i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
    }   }
    if(a>1) res = res/a*(a-1);
    return res;
}
```

6.9 Factor Table

```
const int MXN = 1e7+7; //if>1e7 TLE
int np[MXN],fac[MXN],num[MXN];
// isprime , 最大質因數 , 質因數數量
void table(){
    np[1]=1;
    for(int i=2;i<MXN;++i){
        if(np[i])continue;
        for(int j=i;j<MXN;j+=i){
            if(i!=j)np[j]=1;
            fac[j]=i;
            num[j]++;
        }
    }
}
//質因數分解
vector<int> res;
void div(int x){
    for(;x>1;x/=fac[x])res.push_back(fac[x]);
}
```

6.10 卡塔蘭數

```
//1 1 2 5 14 42 132 429 1430 4862 16796
// O(N)，要記得開long long 跟設定 MOD
cat[0]=1; cat[1]=1;
for(ll i=1 ; i<N ; i++)
    cat[i] = cat[i-1]*(i*4-2)%MOD*qpow(i+1, MOD-2,MOD)%
        MOD;
```

6.11 Miller Rabin

```
// n < 4,759,123,141          3 :  2, 7, 61
// n < 1,122,004,669,633      4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

6.12 PollarRho

```
// does not work when n is prime  O(n^(1/4))
ll f(ll x, ll mod){ return add(mul(x,x,mod),1,mod); }
ll pollard_rho(ll n) {
    if(!(n&1)) return 2;
    while(true){
    ll y=2, x=rand()%(n-1)+1, res=1;
    for(int sz=2; res==1; sz*=2) {
        for(int i=0; i<sz && res<=1; i++) {
            x = f(x, n);
            res = __gcd(abs(x-y), n);
        }
        y = x;
    }
    if (res!=0 && res!=n) return res;
} }
// 如果被卡隨機 用下面的
ll f(ll x,ll c,ll mod){return add(mul(x,x,mod),c,mod);}
ll pollard_rho(ll n){
    ll c=1,x=0,y=0,p=2,q,t=0;
    while(t++%128 or __gcd(p,n)==1){
        if(x==y)c++,y=f(x=2,c,n);
        if(q=mul(p,abs(x-y),n))p=q;
        x=f(x,c,n);y=f(f(y,c,n),c,n);
    }
    return __gcd(p,n);
}
```

6.13 PrimeFactorO(logn)

```
vector<ll> ret;
void fac(ll x){
    if(x<2)return;
    if (miller_rabin(x)) {
      ret.push_back(x);
      return;
    }
    ll f = pollard_rho(x);
    fac(f); fac(x/f);
}
```

6.14 O(1)mul

```
// < long long
ll mul(ll x,ll y,ll mod){
  ll ret=x*y-(ll)((long double)x/mod*y)*mod;
  // ll ret=x*y-(ll)((long double)x*y/mod+0.5)*mod;
  return ret<0?ret+mod:ret;
}
```

6.15 Josephus Problem

```
//base1 n people count k find lastone O(n)
int jo(int n, int k){return n>1?(jo(n-1,k)+k-1)%n+1:1;}
//base0 when k<n O(klogn)
int jo(int n, int k) {
    if (n == 1) return 0;
    if (k == 1) return n - 1;
    if (k > n) return (jo(n - 1, k) + k) % n;
    int f = jo(n - n / k, k) - n % k;
    return f + (f < 0 ? n : (f / (k - 1)));
}
//base1 when k=2 fast find mth
int jo2(int n, int m, int f=0){
    if(n == 1) return 1;
    int kill = (n + f) / 2;
    if(m <= kill) return 2 * m - f;
    return 2 * jo2(n - kill, m - kill, (n ^ f) & 1) -
        (1 ^ f);
}
```

6.16 Harmonic Sum

```
struct Harmonic{
    const double gamma = 0.5772156649;
    //求第N個調和級數
    double nthHarmonic(int n){
        double result = log(n)+gamma;
        return result;
    }
    //求項數n的Sn>k
    int findNearstN(int k){
        int n = exp(k-gamma)+0.5;
        return n;
    }
    // 16n
    // n/1 + n/2 + n/3 + ... + n/n
    //就是這東西
        [20,10,6,5,4,3,2,2,2,2,1,1,1,1,1,1,1,1,1]
    //這是N以下的全因數和
    int nthHarmonicSum9(int n){
        int inv2=qpow(2,MOD-2,MOD),ans=0;
        for(int i=1;i<=n;){
            int v = n/i; int j = n/v;
            int area=(((j-i+1)%MOD)*((j+i)%MOD))%MOD*
                inv2%MOD; //梯形
            ans=(ans+v*area%MOD)%MOD;
            i=j+1;
        }
        return ans;
    }
};
```

6.17 FFT

```
// MAXN一定要2的冪次  先跑pre_fft()
// (must be 2^k)
// before any usage, run pre_fft() first
const int MAXN = 1024*1024;
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
  for(int i=0; i<=MAXN; i++)
    omega[i] = exp(i * 2 * PI / MAXN * I);
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
  int basic = MAXN / n;
  int theta = basic;
  for (int m = n; m >= 2; m >>= 1) {
    int mh = m >> 1;
    for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MAXN-(i*theta%MAXN)
      : i*theta%MAXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
    } }
    theta = (theta * 2) % MAXN;
  }
  int i = 0;
  for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^= k); k >>= 1);
    if (j < i) swap(a[i], a[j]);
  }
  if(inv) for (i = 0; i < n; i++) a[i] /= n;
}
cplx arr[MAXN+1];
inline void mul(int _n,ll a[],int _m,ll b[],ll ans[]){
  int n=1,sum=_n+_m-1;
  while(n<sum)
    n<<=1;
  for(int i=0;i<n;i++) {
    double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
    arr[i]=complex<double>(x+y,x-y);
  }
  fft(n,arr);
  for(int i=0;i<n;i++)
    arr[i]=arr[i]*arr[i];
  fft(n,arr,true);
  for(int i=0;i<sum;i++)
    ans[i]=(long long)(arr[i].real()/4+0.5);
}
```

7 Data Structure

7.1 BIT

```
//注意值域
const int N = 1e5+5;
int bit[N];
struct BIT {
    int n;
    void init(int _n){ n = _n;}
    int low(int x) {return x&-x;}
    void update(int x, int val) {
        while(x<n) bit[x]+=val, x+=low(x);
    }
    int query(int x) {
        int res = 0;
        while(x) res += bit[x], x-=low(x);
        return res;
    }
    int query(int l, int r) {return query(r) - query(l
        - 1); }
};
```

7.2 BIT 二維

```
struct BIT {
    static const int mxn = 2005;
    int bit[mxn][mxn] = {0};
    int low(int x) {return x&-x;}
    void add(int x, int y, int val) {
        for(int i=x ; i<mxn ; i+=low(i)) for(int j=y ;
            j<mxn ; j+=low(j)) bit[i][j]+=val;
    }
    int query(int x, int y) {
        int ans = 0;
        for(int i=x ; i ; i-=low(i)) for(int j=y ; j ;
            j-=low(j)) ans+=bit[i][j];
        return ans;
    }
    int range_query(int a, int b, int x, int y) {
        return query(x, y) - query(x, b-1) - query(a-1,
             y) + query(a-1, b-1);
    }
} bit;
```

7.3 並查集

```
int bin[mxn];
vector<tuple<int, int, int, int>> timing;
// int ans = n; 連通快數量
int fa(int x) {return bin[x]<0?x:fa(bin[x]);}
void uion(int x, int y) {
    x = fa(x); y = fa(y);
    timing.push_back({x, bin[x], y, bin[y]});
    if(x==y) return;
    //ans--;
    if(-bin[x] > -bin[y]) {bin[x]+=bin[y]; bin[y]=x;}
    else {bin[y]+=bin[x]; bin[x]=y;}
}
void undo() {
    auto [a,b,c,d] = timing.back();
    timing.pop_back();
    //if(a!=c) ans++;
    bin[a] = b; bin[c] = d;
}
```

7.4 稀疏表 O(1) 區間最大最小值

```
//st[i][j]表示[i,i+2^j-1]的最值,區間最大長度為log2(n)
//i為1base
const int N = 5e4+5;
int stMax[N][20],stMin[N][20],a[N];
struct ST{
    int k;
    void build(int n,int a[]){
        k=log2(n);
        for(int i = 1; i <= n; i++)  stMin[i][0] =
            stMax[i][0] = a[i];
        for(int j = 1; j <= k; j++){
            for(int i = 1; i + (1 << j) - 1 <= n; i++){
                stMax[i][j] = max(stMax[i][j - 1],
                    stMax[i + (1 << (j - 1))][j - 1]);
                stMin[i][j] = min(stMin[i][j - 1],
                    stMin[i + (1 << (j - 1))][j - 1]);
            }
        }
    }
    int queryMax(int l,int r){
        int j = log2(r-l+1);
        return max(stMax[l][j],stMax[r-(1<<j)+1][j]);
    }
    int queryMin(int l,int r){
        int j = log2(r-l+1);
        return min(stMin[l][j],stMin[r-(1<<j)+1][j]);
    }
}st;
```

7.5 Segment Tree

```
struct seg {
    #define left (index<<1)
    #define right (index<<1|1)
    static const int MXN = 200005;
    int val[MXN*4], tag[MXN*4];
    int a[MXN];
    void push(int index, int l, int r) {
        if(tag[index]!=0) {
            val[index]+=tag[index]*(r-l+1);
            if(l!=r) {
                tag[left] += tag[index];
                tag[right] += tag[index];
            }
            tag[index]=0;
        }
    }
    void pull(int index, int l, int r) {
        int mid = l+r>>1;
        push(left, l, mid);
        push(right, mid+1, r);
        val[index] = val[left]+val[right];
    }
    void build(int index, int l, int r) {
        if(l==r) {
            val[index] = a[l];
            return;
        }
        int mid = (l+r)>>1;
        build(left, l, mid);
        build(right, mid+1, r);
        pull(index, l, r);
    }
    void add(int index, int s, int e, int l, int r, int
         v) {
        if(e<l || r<s) return;
        if(l<=s && e<=r) {
            tag[index] += v;
            push(index, s, e);
            return;
        }
        int mid = (s+e)>>1;
        push(index, s, e);
        add(left, s, mid, l, r, v);
        add(right, mid+1, e, l, r, v);
        pull(index, s, e);
    }
    int query(int index, int s, int e, int l, int r) {
        if(e<l || r<s) return 0;
        if(l<=s && e<=r) {
            push(index, s, e);
            return val[index];
        }
        push(index, s, e);
        int mid = (s+e)>>1;
        return query(right, mid+1, e, l, r)
            +query(left, s, mid, l, r);
    }
} tree;
```

7.6 動態開點線段數

```
// tree.init(區間大小 0~n)
// tree.add(ql, qr, val) 區間加值
// tree.query(ql, qr) 區間總和查詢
struct seg {
    struct Node {
        int val, tag;
        Node *l, *r;
        Node(int v=0) : val(v), tag(0), l(nullptr), r(
            nullptr) {}
    };
    Node* root;
    int n;
    void init(int _n) {
        n = _n;
        root = new Node();
    }
    void push(Node* cur, int l, int r) {
```

```
        if(cur->tag) {
            cur->val += (r-l+1)*cur->tag;
            if(l!=r) {
                if (!cur->l) cur->l = new Node();
                if (!cur->r) cur->r = new Node();
                cur->l->tag += cur->tag;
                cur->r->tag += cur->tag;
            }
        }
        cur->tag = 0;
    }

    void pull(Node* node, int l, int r) {
        int mid = l+r>>1;
        push(node->l, l, mid);
        push(node->r, mid+1, r);
        node->val = node->l->val + node->r->val;
    }

    void add(Node* cur, int l, int r, int ql, int qr,
        int val) {
        if (ql <= l && r <= qr) {
            cur->tag += val;
            push(cur, l, r);
            return;
        }
        if (!cur->l) cur->l = new Node();
        if (!cur->r) cur->r = new Node();

        int mid = (l + r) / 2;
        push(cur, l, r);
        if(ql<=mid) add(cur->l, l, mid, ql, qr, val);
        if(mid+1<=qr) add(cur->r, mid + 1, r, ql, qr,
            val);
        pull(cur, l, r);
    }

    int query(Node* cur, int l, int r, int ql, int qr)
        {
        if(ql<=l && r<=qr) {
            push(cur, l, r);
            return cur->val;
        }
        if (!cur->l) cur->l = new Node();
        if (!cur->r) cur->r = new Node();
        int mid = l+r>>1;
        int ans = 0;
        push(cur, l, r);
        if(ql<=mid) ans+=query(cur->l, l, mid, ql, qr);
        if(mid+1<=qr) ans+=query(cur->r, mid+1, r, ql,
            qr);
        pull(cur, l, r);
        return ans;
    }
    int query(int ql, int qr) {
        return query(root, 0, n, ql, qr);
    }
    void add(int ql, int qr, int val) {
        add(root, 0, n, ql, qr, val);
    }
} tree;
```

7.7 動態開點線段數 2D

```
// tree.init(n,m) 二維大小
// tree.add(qx, qy, val) 座標 (qx, qy) 加值 val
// tree.query(qlx, qly, qrx, qry) 座標 左下角(qlx, qly)
    到 座標 右上角(qrx, qry) 的矩陣總和
// 單點加值
// 區間查詢
// O(lg(n)lg(m))
struct segy {
    int n;
    struct Node {
        int val;
        Node *l, *r;
        Node(int v=0) : val(v), l(nullptr), r(nullptr)
            {}
    };
    segy(int _n=1e9) {n=_n; root=new Node();}
    Node* root;
    void init(int _n=1e9) {
        n = _n;
        root = new Node();
    }
    void pull(Node* node) {
        node->val = (node->l?node->l->val:0) + (node->r
            ?node->r->val:0);
    }
    void add(Node* cur, int l, int r, int pos, int val)
        {
        if (l==r) {
            cur->val += val;
            return;
        }
        int mid = l+r>>1;
        if(pos<=mid) {if(!cur->l) cur->l = new Node();
            add(cur->l, l, mid, pos, val);}
        else {if(!cur->r) cur->r = new Node();add(cur->
            r, mid + 1, r, pos, val);}
        pull(cur);
    }

    int query(Node* cur, int l, int r, int ql, int qr)
        {
        if(ql<=l && r<=qr) {
            return cur->val;
        }
        int mid = l+r>>1;
        int ans = 0;
        if(ql<=mid) {if(!cur->l) cur->l = new Node();
            ans+=query(cur->l, l, mid, ql, qr);}
        if(mid+1<=qr) {if(!cur->r) cur->r = new Node();
            ans+=query(cur->r, mid+1, r, ql, qr);}
        pull(cur);
        return ans;
    }
    int query(int ql, int qr) {
        return query(root, 0, n, ql, qr);
    }
    void add(int pos, int val) {
        add(root, 0, n, pos, val);
    }
};
struct segx {
    struct Node {
        segy * tree_y;
        Node *l, *r;
        Node(int m) : tree_y(new segy(m)), l(nullptr),
            r(nullptr) {}
    };
    Node* root;
    int n,m;
    segx(int _n=1e9, int _m=1e9) {n=_n; m=_m; root=new
        Node(m);}
    void init(int _n=1e9, int _m=1e9) {n=_n; m=_m; root
        =new Node(m);}
    void add(Node* cur, int l, int r, int qx, int qy,
        int val) {
        if(l!=r) {
            int mid = (l + r) / 2;
            if(qx<=mid) {if(!cur->l) cur->l = new Node(
                m); add(cur->l, l, mid, qx, qy, val);}
            else {if(!cur->r) cur->r = new Node(m); add
                (cur->r, mid + 1, r, qx, qy, val);}
        }
        cur->tree_y->add(qy, val);
    }
    int query(Node* cur, int lx, int rx, int qlx, int
        qrx, int qly, int qry) {
        if(qlx<=lx && rx<=qrx) {
            return cur->tree_y->query(qly, qry);
        }
        int mid = lx+rx>>1;
        int ans = 0;
        if(qlx<=mid) {if(!cur->l) cur->l = new Node(m);
            ans+=query(cur->l, lx, mid, qlx, qrx, qly,
            qry);}
        if(mid+1<=qrx) {if(!cur->r) cur->r = new Node(m
            ); ans+=query(cur->r, mid+1, rx, qlx, qrx,
            qly, qry);}
        return ans;
    }
```

```
    int query(int qlx, int qly, int qrx, int qry) {
        return query(root, 0, n, qlx, qrx, qly, qry);
    }
    void add(int qx, int qy, int val) {
        add(root, 0, n, qx, qy, val);
    }
} tree;
```

7.8 持久化線段樹

```
struct seg {
    // 加值持久化線段樹
    struct Node {
        int val;
        Node *l, *r;
    };
    vector<Node*> version;
    void pull(Node* node) {
        node->val = node->l->val+node->r->val;
    }
    Node* build(int l,int r) {
        Node* node=new Node;
        if(l==r) {
            node->val = 0;  //初始值
            return node;
        }
        int mid = (l+r)/2;
        node->l = build(l,mid);
        node->r = build(mid+1,r);
        pull(node);
        return node;
    }
    Node* update(Node* cur,int l,int r,int pos,int v) {
        Node* node=new Node;
        if(l==r){
            //改成加值換這行
            //node->val=cur->val + v;
            node->val=v;
            return node;
        }
        int mid=(l+r)/2;
        if(pos<=mid) {
            node->l=update(cur->l,l,mid,pos,v);
            node->r=cur->r;
        } else {
            node->l=cur->l;
            node->r=update(cur->r,mid+1,r,pos,v);
        }
        pull(node);
        return node;
    }
    int query(Node* cur,int s, int e, int ql, int qr){
        if(ql<=s && e<=qr) return cur->val;
        int ans = 0;
        int mid = (s+e)/2;
        if(ql<=mid) ans += query(cur->l, s, mid, ql, qr
            );
        if(mid+1<=qr) ans += query(cur->r, mid+1, e, ql
            , qr);
        return ans;
    }
} tree;
// push 初始的樹
// tree.version.push_back(tree.build(1, n));

// update(舊版, 1, n, pos, v) return 新版
// 把pos值改成v
```

7.9 Time Segment Tree

```
#include <bits/stdc++.h>
#define int long long int
using namespace std;
int n, q;
struct node{
    int val;
    node *l, *r;
    node(int v) {val=v; l=r=nullptr;}
    node() {val=0; l=r=nullptr;}
};
vector<node*> timing;
node* build(int s, int e) {
    node *ret = new node();
    if(s==e) return ret;
    int mid = (s+e)>>1;
    ret->l = build(s, mid);
    ret->r = build(mid+1, e);
    ret->val = ret->l->val + ret->r->val;
    return ret;
}
node* update(node* pre, int s, int e, int pos, int v) {
    node *ret = new node();
    if(s==e) {ret->val=pre->val+v; return ret;}
    int mid = (s+e)>>1;
    if(pos<=mid) {
        ret->l = update(pre->l, s, mid, pos, v);
        ret->r = pre->r;
    } else {
        ret->r = update(pre->r, mid+1, e, pos, v);
        ret->l = pre->l;
    }
    ret->val = ret->l->val + ret->r->val;
    return ret;
}
void add(int pos, int v) {
    timing.push_back(update(timing.back(), 1, n, pos, v
        ));
}
int que(node* pre, node* now, int l, int r, int k) {
    if(l==r) return r;
    int mid = (l+r)>>1;
    int diff = now->l->val - pre->l->val;
    //printf("now %d~%d  diff %d\n", l, r, diff);
    if(diff>=k) return que(pre->l, now->l, l, mid, k);
    else return que(pre->r, now->r, mid+1, r, k-diff);
    return -1;
}
int query(int l, int r, int k) {
    l--;
    return que(timing[l], timing[r], 1, n, k);
}
int num[100005];
vector<int> sor;
map<int, int> mp;
signed main() {
    cin>>n>>q;
    timing.push_back(build(1, n));
    for(int i=0,a ; i<n ; i++) {
        cin>>a; num[i] = a; sor.push_back(a);
    }
    // add: 1 1 1 2 1
    // num: 3 3 3 4 3
    // sor: 3 4
    sort(sor.begin(), sor.end());
    sor.erase(unique(sor.begin(), sor.end()), sor.end()
        );
    for(int i=0 ; i<n ;i++) {
        int pos = lower_bound(sor.begin(), sor.end(),
            num[i]) - sor.begin() + 1;
        //printf("mp[%d] = %d\n", pos, num[i]);
        mp[pos] = num[i];
        num[i] = pos;
        add(num[i], 1);
    }
    while(q--) {
        int a, b, c; cin>>a>>b>>c;
        cout<<mp[query(a, b, c)]<<endl;
    }
}
```

7.10 Treap

```
struct Treap {
  int sz, val, pri, tag;
  Treap *l , *r;
  Treap(int _val){
    val=_val; sz=1;
    pri=rand(); l=r=NULL; tag=0;
  }
```

```
};
int Size(Treap *a) {return a?a->sz:0;}
void pull(Treap *a) {
  a->sz = Size(a->l) + Size(a->r) + 1;
}
//val of a is always bigger than val of b
Treap* merge(Treap *a ,Treap *b) {
  if(!a || !b) return a ? a : b;
  if(a->pri>b->pri) {
    a->r = merge(a->r ,b);
    pull(a);
    return a;
  } else {
    b->l = merge( a , b->l );
    pull(b);
    return b;
  }
}
// a<k, b>=k
void split(Treap *t, int k, Treap*&a, Treap*&b){
  if(!t) {a=b=NULL; return; }
  if(k <= t->val) {
    b = t;
    split(t->l, k, a, b->l);
    pull(b);
  }
  else {
    a = t;
    split(t->r,k,a->r,b);
    pull(a);
    }
}
Treap* add(Treap *t, int v) {
    Treap *val = new Treap(v);
    Treap *l = NULL, *r = NULL;
    split(t, v, l, r);
    return merge(merge(l, val), r);
}
Treap* del(Treap *t, int v) {
    Treap *l, *mid, *r, *temp;
    split(t, v, l, temp);
    split(temp, v+1, mid, r);
    return merge(l, r);
}

// base 1
int position(Treap *t, int p) {
    if(Size(t->l)+1==p) return t->val;
    if(Size(t->l)<p) return position(t->r, p-Size(t->l)
        -1);
    else return position(t->l, p);
}

//num of >= k
int query(Treap *t, int k) {
    if(!t) return 0;
    if(t->val==k) return Size(t->l)+1;
    if(t->val>k) return query(t->l, k);
    return Size(t->l)+1+query(t->r, k);
}
```

7.11 PBDS

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
using namespace __gnu_pbds;
// ordered_set s;
// s.insert(1); s.erase(s.find(1));
// order_of_key (k) : Number of items strictly smaller
    than k .
// find_by_order(k) : K-th element in a set (counting
    from zero). (return iterator)
```

8 String

8.1 SA

```
#pragma GCC optimize("O3,unroll-loops")
#pragma target optimize("avx2,bmi,bmi2,lzcnt,popcnt")
#include<bits/stdc++.h>
#include<chrono>
#define mid (l + r) / 2
using namespace std;
const int N = 100010;
struct SA{
#define REP(i,n) for ( int i=0; i<int(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )
  bool _t[N*2];
  int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
      hei[N], r[N];
  int operator [] (int i){ return _sa[i]; }
  void build(int *s, int n, int m){
    memcpy(_s, s, sizeof(int) * n);
    sais(_s, _sa, _p, _q, _t, _c, n, m);
    mkhei(n);
  }
  void mkhei(int n){
    REP(i,n) r[_sa[i]] = i;
    hei[0] = 0;
    REP(i,n) if(r[i]) {
      int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
      while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
      hei[r[i]] = ans;
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t,
      int *c, int n, int z){
    bool uniq = t[n-1] = true, neq;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
        lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
    memcpy(x, c, sizeof(int) * z); \
    XD; \
    memcpy(x + 1, c, sizeof(int) * (z - 1)); \
    REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[s[sa[i
        ]-1]]++] = sa[i]-1; \
    memcpy(x, c, sizeof(int) * z); \
    for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i
        ]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
    MS0(c, z);
    REP(i,n) uniq &= ++c[s[i]] < 2;
    REP(i,z-1) c[i+1] += c[i];
    if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
    for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
        +1] ? t[i+1] : s[i]<s[i+1]);
    MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[--x[s[i
        ]]]=p[q[i]=nn++]=i);
    REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
      neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
      ns[q[lst=sa[i]]]=nmxz+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz
        + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[s[p[
        nsa[i]]]]] = p[nsa[i]]);
  }
}sa;
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
  // should padding a zero in the back
  // ip is int array, len is array length
  // ip[0..n-1] != 0, and ip[len] = 0
  ip[len++] = 0;
  sa.build(ip, len, 128);
  for (int i=0; i<len; i++) {
    H[i] = sa.hei[i + 1];
    SA[i] = sa._sa[i + 1];
  }
  // resulting height, sa array \in [0,len)
}
bool check(string &s,string &t,int p){
    for(int i=0;i<t.size() && i+p<s.size();++i){
        if(t[i]<s[i+p])return 1;
        else if(t[i]>s[i+p]) return 0;
    }
    if(t.size()>s.size()-p) return 0;
    return 1;
}
```

```
//example for finding patterns in a string
string s,t;
int ip[N],len;
int main(){
    int n;
    cin>>s>>n;
    len = s.length();
    for(int i=0;i<len;++i) ip[i]=(int)s[i];
    ip[len] = 0;
    suffix_array(ip,len);
    int l,r;
    for(int i=0;i<n;++i){
        cin>>t;
        l = 0, r = s.size()-1;
        while(l!=r){
            if(check(s,t,SA[mid])) r=mid;
            else l = mid+1;
        }
        bool f=1;
        if(t.size()>s.size()-SA[l]){
            cout<<"NO\n",f=0;
            continue;
        }
        for(int j=0;j<t.size();++j){
            if(t[j]!=s[j+SA[l]]){
                cout<<"NO\n",f=0;
                break;;
            }
        }
        if(f) cout<<"YES\n";
    }
}
```

8.2 KMP

```
// 回傳所有匹配成功的起始位置，s為文本，t為匹配字串
// nxt表示為匹配失敗時要退回的位置，也是t字串的相等前綴
    後綴的最大長度
// *注意前綴後綴為長度最多為n-1的子字串
// nxt[j] = -1 if j=0
//        0  if 沒有相等的前綴後綴
//        K  k 為相等前綴後綴的最大長度
// 以下為例子
//       j: 0 1 2 3 4 5 6
//       t: a b a a b e
// nxt[j]:-1 0 0 1 1 2 0
// O(n+m)，n為s長，m為t長
const int MXN = 1e6+5;
int nxt[MXN];
vector<int> KMP(string s,string t){
    int slen = s.length(), tlen = t.length(), i=0,j=0,k
        =-1;
    nxt[0]=-1;
    while(j<tlen){//build nxt
        if(k==-1 || t[j]==t[k]) nxt[++j] = ++k;
        else    k=nxt[k];
    }
    i=0,j=0;
    vector<int> ret;
    while(i<slen){// matching
        if(j==-1||s[i]==t[j]) i++,j++;
        else    j=nxt[j];
        if(j==tlen){
            ret.push_back(i-tlen+1);//1-base
            j=nxt[j];
        }
    }
    return ret;
}

//另一版
//if t is the substring of s:
//if t in s:
bool cmp(string s, string t) {
    vector<int> front(t.size(), 0);
    for(int i=1, j=0 ; i<t.size() ; i++) {
        while(j>0 && t[i]!=t[j]) j = front[j-1];
        if(t[i]==t[j]) j++;
        front[i] = j;
    }
    int j=0, i=0;
    while(i<s.size()) {
        if(s[i]==t[j]) j++,i++;
        else {i += (j==0); j = (j<1?0:front[j-1]);}
        if(j>=t.size()) return true;
    }
    return false;
}
```

8.3 Single Hash

```
//字串雜湊前的idx是0-base，雜湊後為1-base
//H[R] - H[L-1] * p^(R-L+1)
//cmp的+modl是為了防止負數
//記得build完之後要buildPow
//小心遇到hash出負數要記得+modl
#define int long long
const int p = 75577, modl = 1e9 + 7,MXN = 1e6+5;
int Hash[MXN],qpow[MXN];
void build(const string& s) {
    Hash[0]=0;
    for(int i=1; i<=s.size(); i++)
        Hash[i] = (Hash[i-1] * p + s[i-1]) % modl;
}
void buildPow(){
    qpow[0]=1;
    for(int i=1;i<MXN;++i) qpow[i]=qpow[i-1]*p%modl;
}
bool cmp(int i, int j, int len) {
    return (Hash[i+len-1] - Hash[i-1] * qpow[len] %
        modl + modl) % modl ==
    (Hash[j+len-1] - Hash[j-1] * qpow[len] % modl +
        modl) % modl;
}
int get(int i, int j) {
    return (Hash[j]-Hash[i-1]*qpow[j-i+1]%modl+modl)%
        modl;
}
```

8.4 Double Hash

```
//字串雜湊前的idx是0-base，雜湊後為1-base
//即區間為 [0,n-1] -> [1,n]
//若要取得區間[L,R]的值則
//H[R] - H[L-1] * p^(R-L+1)
//cmp為比較從i開始長度為len的字串和從j開始長度為len的字
    串是否相同
//(h[i+len-1] - h[i-1] * qpow(p, len) % modl + modl)
#define int long long
#define x first
#define y second
const int P1 = 75577, P2 = 17, MOD = 1e9 + 7,MXN = 1e6
    +5;
pair<int,int> Hash[MXN];
int qpow[2][MXN];
void build(const string& s){
  pair<int,int> val = make_pair(0,0);
  Hash[0]=val;
  for(int i=1; i<=s.size(); i++){
  val.x = (val.x * P1 + s[i-1]) % MOD;
  val.y = (val.y * P2 + s[i-1]) % MOD;
  Hash[i] = val;
  }
}
void buildPow(){
    qpow[0][0]=qpow[1][0]=1;
    for(int i=1;i<MXN;++i){
        qpow[0][i]=qpow[0][i-1]*P1%MOD;
        qpow[1][i]=qpow[1][i-1]*P2%MOD;
    }
}
bool cmp( int i, int j, int len ) {
    return ((Hash[i+len-1].x-Hash[i-1].x*qpow[0][len]%
        MOD+MOD)%MOD == (Hash[j+len-1].x-Hash[j-1].x*
        qpow[0][len]%MOD+MOD)%MOD)
    && ((Hash[i+len-1].y-Hash[i-1].y*qpow[1][len]%MOD+
        MOD)%MOD == (Hash[j+len-1].y-Hash[j-1].y*qpow
        [1][len]%MOD+MOD)%MOD);
```

```
}
pair<int, int> get(int i, int j) {
    return {(Hash[j].x-Hash[i-1].x*qpow[0][j-i+1]%MOD+
        MOD)%MOD, (Hash[j].y-Hash[i-1].y*qpow[1][j-i
        +1]%MOD+MOD)%MOD};
}
```

8.5 Trie

```
//cnt為記錄有多少個一樣的單詞且end的時候才有數字
const int MXN=1e6+5;//MXN取文本長
int trie[MXN][26], cnt[MXN],tot=0;//0 base
void update(string s){
    int p=0;//0 base
    for(int i=0;i<s.size();++i){
        int ch = s[i]-'a';
        if(!trie[p][ch]) trie[p][ch]=++tot;
        p = trie[p][ch];
    }
    cnt[p]++;
}
int query(string s){
    int p=0;
    for(int i=0;i<s.size();++i){
        int ch=s[i]-'a';
        p = trie[p][ch];
        if(!p) return 0;
    }
    return cnt[p];
}
void visualizeTrie(int node = 0, int depth = 0) {//for
    debug
    for (int i = 0; i < 26; ++i) {
        if (trie[node][i]) {
            for (int j = 0; j < depth; ++j) cout << "
                ";
            cout << (char)('a' + i) << " (" << cnt[trie
                [node][i]] << ")\n";
            visualizeTrie(trie[node][i], depth + 1);
        }
    }
}
```

8.6 Z value

```
// O(n)
//z[i] = lcp(s[1...],s[i...])
//1base
int z[MAXN];
void Z_value(const string& s) {
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

8.7 MinRotation

```
//rotate(begin(s),begin(s)+minRotation(s),end(s))
//For example,rotations of acab are acab, caba, abac,
    and baca.
//find lexicographically minimal rotation of a string
int minRotation(string s) {
  int a = 0, N = s.size(); s += s;
  for(int b=0;b<N;b++) for(int k=0;k<N;k++) {
    if(a+k == b || s[a+k] < s[b+k])
      {b += max(0, k-1); break;}
    if(s[a+k] > s[b+k]) {a = b; break;}
  } return a;
}
```

8.8 Manacher 馬拉車回文

```
// O(N)求以每個字元為中心的最長回文半徑
// 頭尾以及每個字元間都加入一個
// 沒出現過的字元，這邊以'@'為例
// s為傳入的字串，len為字串長度
// z為儲存以每個字元為中心的回文半徑+1(有包含'@'要小心)
// ex: s = "abaac" -> "@a@b@a@a@c@"
// z =                [12141232121]
const int MXN = 1e6+5;
int z[2*MXN];
char s[2*MXN];
void z_value_pal(char *s,int len,int *z){
  len=(len<<1)+1;
  for(int i=len-1;i>=0;i--)
    s[i]=i&1?s[i>>1]:'@';
  z[0]=1;
  for(int i=1,l=0,r=0;i<len;i++){
    z[i]=i<r?min(z[l+l-i],r-i):1;
    while(i-z[i]>=0&&i+z[i]<len&&s[i-z[i]]==s[i+z[i]])
        ++z[i];
    if(i+z[i]>r) l=i,r=i+z[i];
} }
// cin>>s;
// z_value_pal(s,strlen(s),z);
// int mx=-1,mxi=0;
// for(int i=0;i<=strlen(s);++i)
//     if(mx<z[i]) mx = z[i], mxi = i;
// mx--;
// for(int i=mxi-mx;i<=mxi+mx;++i)
//     if(s[i]!='@') cout<<s[i];
```

8.9 PalTree 回文樹

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴，aba的fail是a
const int MXN = 1000010;
struct PalT{
    int nxt[MXN][26],fail[MXN],len[MXN];
    int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
    int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
    char s[MXN]={-1};
    int newNode(int l,int f){
        len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
        memset(nxt[tot],0,sizeof(nxt[tot]));
        diff[tot]=(l>0?l-len[f]:0);
        sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f)
            ;
        return tot++;
    }
    int getfail(int x){
        while(s[n-len[x]-1]!=s[n]) x=fail[x];
        return x;
    }
    int getmin(int v){
        dp[v]=fac[n-len[sfail[v]]-diff[v]];
        if(diff[v]==diff[fail[v]])
            dp[v]=min(dp[v],dp[fail[v]]);
        return dp[v]+1;
    }
    int push(){
        int c=s[n]-'a',np=getfail(lst);
        if(!(lst=nxt[np][c])){
            lst=newNode(len[np]+2,nxt[getfail(fail[np])
                ][c]);
            nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
        }
        fac[n]=n;
        for(int v=lst;len[v]>0;v=sfail[v])
            fac[n]=min(fac[n],getmin(v));
        return ++cnt[lst],lst;
    }
    void init(const char *_s){
        tot=lst=n=0;
        newNode(0,1),newNode(-1,1);
        for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push()
            ;
```

```
        for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
    }
} palt;
// state 數組
//    state[i] 代表第 i 個字元為結尾的最長回文編號(編號是甚麼不重要)
//
//    S = “abacaaba”
//
//    以第 2(0-base) 個字元為結尾的最長回文是 aba
//    以第 7(0-base) 個字元為結尾的最長回文是 aba
//    兩個最長回文都相同，因此 state[2] 會等於 state[7]

// len 數組
//    求出某個 state 的長度
//
//    S = “aababa”
//
//    (0-base)
//    len[state[1]] = 2 (“aa”)
//    len[state[3]] = 3 (“aba”)
//    len[state[5]] = 5 (“ababa”)

// num 數組
//    某個state的回文有幾個回文後綴
//
//    假設某個 state 代表的回文為 = “ababa” 為例
//    state 代表的回文的 num = 3
//    -> ababa -> aba -> a

// cnt 數組
//    某個 state 的回文在整個字串中出現次數
//
//    S = “aababaa”
//    state[3] 代表的回文為 “aba” 在整個字串中出現 2 次
//    因此 cnt[state[3]] = 2

// fail數組
//    每個 state 的次長回文後綴的 state 編號
//
//    S = “ababa”
//    len[fail[4]] = 3 (fail[4] = “aba”)
//    len[fail[2]] = 1 (fail[2] = “a”)
//    len[fail[0]] = 0 (fail[0] = “” 空字串)
//    0 所代表的 state 是空字串
```

8.10 DistinctSubsequence

```
//預設為小寫字母
//return the number of distinct non-empty subsequences of sting
#define int long long
int mod = 1e9 + 7;
vector<int> cnt(26);
int distinct_subsequences(string s) {
    for (char c : s)
    cnt[c - 'a'] = accumulate(begin(cnt), end(cnt), 1LL) % mod;
    return accumulate(begin(cnt), end(cnt), 0LL) % mod;
}
```

9 Tree

9.1 LCA

```
//先建edge[MXN]
//跑dfs，再跑makeanc
//之後才可以呼叫lca
// 0-base
const int MXN=1e5;
const int logN=__lg(MXN);
int tin[MXN],tout[MXN],anc[MXN][logN+1];
vector<int> edge[MXN];
int ti=0;
void dfs(int x,int f){
    anc[x][0]=f;
    tin[x]=ti++;
    for(int u:edge[x]){
        if(u==f)continue;
        dfs(u,x);
    }
    tout[x]=ti++;
}
// x is y's anc
inline bool isanc(int x,int y){
    return tin[x]<=tin[y] && tout[x]>=tout[y];
}
int lca(int x,int y){
    if(isanc(x,y))return x;
    if(isanc(y,x))return y;
    for(int i=logN;i>=0;--i){
        if(!isanc(anc[y][i],x)){
            y=anc[y][i];
        }
    }
    return anc[y][0];
}
void makeanc(int n){
    for(int i=1;i<=logN;++i){
        for(int j=0;j<n;++j){
            anc[j][i] = anc[anc[j][i-1]][i-1];
        }
    }
}
```

9.2 TreeHash

```
// 1. dfs 先做子樹
// 2. 葉節點的hash值為1
// 3. 對於節點x，其hash值為紀錄x的所有子樹的hash值(紀錄到temp)，然後由小排到大(排除子樹的隨機問題)
// 4. n表示節點x有幾個子樹，p和MOD通常為一個很大的質數，由此算出x的hash值
// 5. 樹根的hash值即為整顆樹的hash值，若兩顆樹的hash值相同，則兩棵樹就是同構
const int MXN = 200005;
int subtree_sz[MXN];
int hash_[MXN];
int base = 44560482149;
int MOD = 274876858367;
int dfs(int x, int fa, vector<int>* edge){
    vector<int> temp;
    subtree_sz[x] = 1;
    for(int child : edge[x]){
        if(child==fa) continue;
        temp.push_back(dfs(child, x, edge));
        subtree_sz[x] += subtree_sz[child];
    }
    sort(temp.begin(), temp.end());
    int ret = subtree_sz[x];
    for(int v : temp){
        ret = (((ret * base + v + ret) % MOD + ret) % MOD + v) % MOD ;
    }
    hash_[x] = ret;
    return ret;
}
```

9.3 輕重鏈剖分

```
const int MXN = 2e5+7;
int top[MXN], son[MXN], dfn[MXN], rnk[MXN], dep[MXN], father[MXN];
vector<int> edge[MXN];
int dfs1(int v, int fa, int d) {
    int maxsz = -1, maxu, total = 1;
    dep[v] = d;
    father[v] = fa;
    for(int u: edge[v]) {
        if(fa == u) continue;
        int temp = dfs1(u, v, d+1);
        total += temp;
```

```
        if(temp>maxsz) {
            maxsz = temp;
            maxu = u;
        }
    }
    if(maxsz==-1) son[v] = -1;
    else son[v] = maxu;
    return total;
}

int times = 1;
void dfs2(int v, int fa) {
    rnk[times] = v;
    dfn[v] = times++;
    top[v] = (fa==-1 || son[fa] != v ? v : top[fa]);
    if(son[v]!=-1) dfs2(son[v], v);
    for(int u: edge[v]) {
        if(fa == u || u == son[v]) continue;
        dfs2(u, v);
    }
}
//rnk: 剖分後的編號 (rnk[時間] = 原點)
//dfn: 剖分後的編號 (dfn[原點] = 時間)
//top: 剖分的頭頭
//son: 剖分的重兒子
```

10 Geometry

10.1 2D Definition

```
#define ld long double
const ld eps=1e-10;
int dcmp(ld x){if(fabs(x)<eps) return 0;else return x
    <0?-1:1;}
struct Pt{
    ld x,y;
    Pt(ld x=0,ld y=0):x(x),y(y){}
    Pt operator+(const Pt &a) const {
        return Pt(x+a.x, y+a.y);  }
    Pt operator-(const Pt &a) const {
        return Pt(x-a.x, y-a.y);  }
    Pt operator*(const ld &a) const {
        return Pt(x*a, y*a);  }
    Pt operator/(const ld &a) const {
        return Pt(x/a, y/a);  }
    ld operator*(const Pt &a) const {//dot
        return x*a.x + y*a.y;  }
    ld operator^(const Pt &a) const {//cross
        return x*a.y - y*a.x;  }
    bool operator<(const Pt &a) const {
        return x < a.x || (x == a.x && y < a.y); }
        //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0
            && dcmp(y-a.y) < 0); }
    bool operator>(const Pt &a) const {
        return x > a.x || (x == a.x && y > a.y); }
        //return dcmp(x-a.x) > 0 || (dcmp(x-a.x) == 0
            && dcmp(y-a.y) > 0); }
    bool operator==(const Pt &a) const {
        return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;  }
        // return x == other.x && y == other.y;
     bool operator!=(const Pt &a) const {
        return !(*this == a);  }
};
typedef Pt Vec;
ld Dot(Vec a,Vec b){return a.x*b.x+a.y*b.y;}
ld Cross(Vec a,Vec b){return a.x*b.y-a.y*b.x;}
ld Length(Vec a){return sqrt(Dot(a,a));}
int Sgn(double x){ return (x > -eps) - (x < eps); }//
    return 0: x==0, 1: x>0, -1: x<0
ld Angle(Vec a,Vec b){return acos(Dot(a,b)/Length(a)/
    Length(b));}//弧度
ld Degree(Vec a,Vec b){return Angle(a,b)*180/acos(-1);}
    //角度
ld Ori(Pt a,Pt b,Pt c){return Cross(b-a,c-a);}//1.(a,b)
    x(a,c)的面積  2. a在bc左側>0 3. a在bc右側<0 4. a在
    bc上==0
Vec Rotate(Vec a,ld rad){return Vec(a.x*cos(rad)-a.y*
    sin(rad),a.x*sin(rad)+a.y*cos(rad));}//逆時針旋轉,
    rad為弧度
Vec Normal(Vec a){ld L=Length(a);return Vec(-a.y/L,a.x/
    L);}//單位法向量，確保a不是零向量
Vec Unit(Vec x) { return x / Length(x); } //單位向量
Vec Perp( const Vec v ){ return { v.y , -v.x };}//垂直
    向量
bool argcmp(const Pt &a, const Pt &b) { // 極角cmp: arg
    (a) < arg(b)
    int f = (Pt{a.y, -a.x} > Pt{} ? 1 : -1) * (a != Pt
        {});
    int g = (Pt{b.y, -b.x} > Pt{} ? 1 : -1) * (b != Pt
        {});
    return f == g ? (a ^ b) > 0 : f < g;
}
struct Circle {
  Pt o; ld r;
  Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};
```

10.2 Line Definition

```
struct Line {
  Pt a, b, v; // start, end, end-start
  ld ang;
  Line(Pt _a=Pt(0, 0), Pt _b=Pt(0, 0)):a(_a),b(_b) { v
      = b-a; ang = atan2(v.y, v.x); }
  bool operator<(const Line &L) const {
    return ang < L.ang;
} };
int PtSide(Pt p, Line L) {//return 1:左側 0:線上 -1:右
    側
    return Sgn(Ori(L.a, L.b, p));
}
bool PtOnSeg(Pt p, Line L) {//點是否在線段上
    return Sgn(Ori(L.a, L.b, p)) == 0 && Sgn((p - L.a)
        * (p - L.b)) <= 0;
}
Pt Proj(Pt p, Line l) {//點到線段的投影點
    Pt dir = Unit(l.b - l.a);
    return l.a + dir * (dir * (p - l.a));
}
bool isInter(Line l, Line m) {//判斷兩線段是否相交
    if (PtOnSeg(m.a, l) || PtOnSeg(m.b, l) ||
        PtOnSeg(l.a, m) || PtOnSeg(l.b, m))
        return true;
    return PtSide(m.a, l) * PtSide(m.b, l) < 0 &&
           PtSide(l.a, m) * PtSide(l.b, m) < 0;
}
Pt LineInter(Line l, Line m) {//兩線段交點
    double s = Ori(m.a, m.b, l.a), t = Ori(m.a, m.b, l.
        b);
    return (l.b * s - l.a * t) / (s - t);
}
```

10.3 Basic

```
//確保兩直線P+tv和Q+tw有唯一交點且Cross(v,w)非零
Pt getLineIntersect(Line a, Line b) {
  Pt p1 = a.a, p2 = a.b, q1 = b.a, q2 = b.b;
  ld f1 = (p2-p1)^(q1-p1),f2 = (p2-p1)^(p1-q2),f;
  if(dcmp(f=f1+f2) == 0)
    return dcmp(f1)?Pt(NAN,NAN):Pt(INFINITY,INFINITY);
  return q1*(f2/f) + q2*(f1/f);
}

//點到直線距離
double distanceToLine(Pt p,Pt a,Pt b){
    Vec v1=b-a,v2=p-a;
    return fabs(Cross(v1,v2)/Length(v1));
}

//點到線段距離
double distanceToSegment(Pt p,Pt a,Pt b){
    if(a==b) return Length(p-a);
    Vec v1=b-a,v2=p-a,v3=p-b;
    if(dcmp(Dot(v1,v2))<0) return Length(v2);
    else if(dcmp(Dot(v1,v3))>0) return Length(v3);
    else return fabs(Cross(v1,v2)/Length(v1));
}
```

```
//點到直線投影
Pt GetLineProjection(Pt p,Pt a,Pt b){
    Vec v=b-a;
    return a+v*(Dot(v,p-a)/Dot(v,v));
}

//點p於直線ab的對稱點
Pt getSymmetryPoint(Pt p,Pt a,Pt b){
    Pt q=getLineProjection(p,a,b);
    return q*2-p;
}

//判斷線段相交(剛好交一點)，若兩線段共線->c1=c2=0
bool isSegmentProperIntersection(Pt a1,Pt a2,Pt b1,Pt
    b2){
    double c1=Cross(a2-a1,b1-a1),c2=Cross(a2-a1,b2-a1),
        c3=Cross(b2-b1,a1-b1),c4=Cross(b2-b1,a2-b1);
    return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
}

//判斷線段相交(只要有交點即可)
bool isSegmentNotProperIntersection(Pt a1,Pt a2,Pt b1,
    Pt b2){
    return max(a1.x,a2.x)>=min(b1.x,b2.x)&&max(b1.x,b2.
        x)>=min(a1.x,a2.x)&&max(a1.y,a2.y)>=min(b1.y,b2
        .y)&&max(b1.y,b2.y)>=min(a1.y,a2.y)
    &&dcmp(Cross(a1-b1,a2-b1))*dcmp(Cross(a1-b2,a2-b2))
        <=0&&dcmp(Cross(b1-a1,b2-a1))*dcmp(Cross(b1-a2,
        b2-a2))<=0;
}

//點是否在線段上
bool isOnSegment(Pt p,Pt a1,Pt a2){
    return dcmp(Cross(a1-p,a2-p))==0&&dcmp(Dot(a1-p,a2-
        p))<=0;
}
```

10.4 PolygonArea

```
//須注意long long 及 加上絕對值
double polygonArea(Pt* p,int n){
    double area=0;
    for(int i=1;i<n-1;++i){
        area+=Cross(p[i]-p[0],p[i+1]-p[0]);
    }
    return area/2;
}
```

10.5 IsPointInPolygon

```
//判斷點是否在多邊形內部
int isPointInPolygon(Pt p,Pt* poly,int n){
    int wn=0;
    for(int i=0;i<n;++i){
        if(isOnSegment(p,poly[i],poly[(i+1)%n])) return
            -1;//在邊界上
        int k=dcmp(Cross(poly[(i+1)%n]-poly[i],p-poly[i
            ]));
        int d1=dcmp(poly[i].y-p.y);
        int d2=dcmp(poly[(i+1)%n].y-p.y);
        if(k>0&&d1<=0&&d2>0) wn++;
        if(k<0&&d2<=0&&d1>0) wn--;
    }
    if(wn!=0) return 1;//內部
    return 0;//外部
}
```

10.6 ConvexHull

```
//若要求高精度用dcmp比較
//若是搞int點要記得定義改int
//輸入不能有重複點，注意h的點未排序!
//若需保留共線點，把while裡的Ori判斷式改成<=0
void hull(vector<Pt> &dots) { // n=1 => ans = {}
  sort(dots.begin(), dots.end());
  vector<Pt> ans(1, dots[0]);
  for (int ct = 0; ct < 2; ++ct, reverse(all(dots)))
    for (int i = 1, t = SZ(ans); i < SZ(dots); ans.
        push_back(dots[i++]))
      while (SZ(ans) > t && Ori(ans[SZ(ans) - 2], ans.
          back(), dots[i]) <= 0)
        ans.pop_back();
  ans.pop_back(), ans.swap(dots);
}
```

10.7 MinkowskiSum

```
//定義：給兩點集合A,B，Minkowski Sum是A+B={a+b|a∈A,b∈B}
    的凸包
//視覺化為一個凸包 A 繞著凸包 B 轉一圈，O(N)
//兩個凸多邊形的 Minkowski sum，也會是凸多邊形
//P 和 Q 組成的 Minkowski sum 最多有 |P|+|Q| 個點
//在凸包 A 和 B 上的邊也會在 Minkowski sum 上出現
//傳入的點集合不用逆時針排序
vector<Pt> Minkowski(vector<Pt> A, vector<Pt> B) { // |
    A|,|B|>=3
  hull(A), hull(B);
  vector<Pt> C(1, A[0] + B[0]), s1, s2;
  for (int i = 0; i < SZ(A); ++i)
    s1.push_back(A[(i + 1) % SZ(A)] - A[i]);
  for (int i = 0; i < SZ(B); i++)
    s2.push_back(B[(i + 1) % SZ(B)] - B[i]);
  for (int i = 0, j = 0; i < SZ(A) || j < SZ(B);)
    if (j >= SZ(B) || (i < SZ(A) && Cross(s1[i], s2[j])
        >= 0))
      C.push_back(B[j % SZ(B)] + A[i++]);
    else
      C.push_back(A[i % SZ(A)] + B[j++]);
  return hull(C), C;
}
```

10.8 Polygon Shortest Distance

```
//給兩多邊形，求最短距離
double PolyDist(vector<Pt> A, vector<Pt> B) {
    for (auto &p : B) p = {-p.x, -p.y};
    auto C = Minkowski(A, B); // assert SZ(C) > 0
    if (isPointInPolygon(Pt{},C.data(),C.size()))
        return 0;
    double ans = distanceToSegment(Pt{},C.back(), C[0])
        ;
    for (int i = 0; i + 1 < SZ(C); ++i) {
        ans = min(ans, distanceToSegment(Pt{},C[i], C[i
            + 1]));
    }
    return ans;
}
```

10.9 ConvexHullTrick

```
struct Convex {
    int n;
    vector<Pt> A, V, L, U;
    //init，pass convex hull points
    Convex(const vector<Pt> &_A) : A(_A), n(_A.size())
        { // n >= 3
        auto it = max_element(all(A));
        L.assign(A.begin(), it + 1);
        U.assign(it, A.end()), U.push_back(A[0]);
        for (int i = 0; i < n; i++) {
            V.push_back(A[(i + 1) % n] - A[i]);
        }
    }
    int PtSide(Pt p, Line L) {
        return dcmp((L.b - L.a)^(p - L.a));
    }
    int inside(Pt p, const vector<Pt> &h, auto f) {
        auto it = lower_bound(all(h), p, f);
        if (it == h.end()) return 0;
        if (it == h.begin()) return p == *it;
```

```
        return 1 - dcmp((p - *prev(it))^(*it  - *prev(
            it)));
    }
    // 1. whether a given point is inside the Convex
        Hull
    // ret 0: out, 1: on, 2: in
    int inside(Pt p) {
        return min(inside(p, L, less{}), inside(p, U,
            greater{}));
    }
    static bool cmp(Pt a, Pt b) { return dcmp(a ^ b) >
        0; }
    // 2. Find tangent points of a given vector
    // ret the idx of far/closer tangent point
    int tangent(Pt v, bool close = true) {
        assert(v != Pt{});
        auto l = V.begin(), r = V.begin() + L.size() -
            1;
        if (v < Pt{}) l = r, r = V.end();
        if (close) return (lower_bound(l, r, v, cmp) -
            V.begin()) % n;
        return (upper_bound(l, r, v, cmp) - V.begin())
            % n;
    }
    // 3. Find 2 tang pts on CH of a given outside
        point
    // return index of tangent points
    // return {-1, -1} if inside CH
    array<int, 2> tangent2(Pt p) {
        array<int, 2> t{-1, -1};
        if (inside(p) == 2) return t;
        if (auto it = lower_bound(all(L), p); it != L.
            end() and p == *it) {
            int s = it - L.begin();
            return {(s + 1) % n, (s - 1 + n) % n};
        }
        if (auto it = lower_bound(all(U), p, greater{})
            ; it != U.end() and p == *it) {
            int s = it - U.begin() + L.size() - 1;
            return {(s + 1) % n, (s - 1 + n) % n};
        }
        for (int i = 0; i != t[0]; i = tangent((A[t[0]
            = i] - p), 0));
        for (int i = 0; i != t[1]; i = tangent((p - A[t
            [1] = i]), 1));
        return t;
    }
    int find(int l, int r, Line L) {
        if (r < l) r += n;
        int s = PtSide(A[l % n], L);
        return *ranges::partition_point(views::iota(l,
            r),
            [&](int m) {
                return PtSide(A[m % n], L) == s;
            }) - 1;
    };
    // 4. Find intersection point of a given line
    // intersection is on edge (i, next(i))
    vector<int> intersect(Line L) {
        int l = tangent(L.a - L.b), r = tangent(L.b - L
            .a);
        if(PtSide(A[l], L) == 0)    return {l};
        if(PtSide(A[r], L) == 0)    return {r};
        if (PtSide(A[l], L) * PtSide(A[r], L) > 0)
            return {};
        return {find(l, r, L) % n, find(r, l, L) % n};
    }
};
```

10.10 Polar Sort

```
//極角排序，從270度開始逆時針排序
bool cmp(const Pt& lhs,const Pt&rhs){
    if(Cross((lhs < Pt()),(rhs < Pt())))
        return (lhs < Pt()) < (rhs < Pt());
    return Cross(lhs,rhs) > 0;
}
/* 若要以p[i]為原點排序->計算v=p[j]-p[i]
for(int j=0;j<n;++j){
    if(i!=j){
        Vector v = p[j]-p[i];
        node[nodeSz++] = {v,j};
    }
}
sort(node,node+nodeSz,cmp);
*/
```

10.11 PickTheorm

```
int area,in,on;//area:多邊形面積 in:內部格點數 on:邊界
    格點數
void PickTheorm(Pt* p,int n){
    area=polygonArea(p,n);
    for(int i=0;i<n;++i){
        on+=__gcd(abs((int)p[i].x-(int)p[(i+1)%n].x),
            abs((int)p[i].y-(int)p[(i+1)%n].y));
    }
    in=abs(area)+1-on/2;
}
```

10.12 ShortestPair

```
//最近點對距離注意若整數要define double long long
double closestEuclideanDistance(Pt* p,int n){
    sort(p,p+n);
    set<Pt> s={{p[0].y,p[0].x}};
    int j = 0;
    Pt t;
    double dd=LLONG_MAX,d;
    for(int i=1;i<n;++i){
        d = sqrt(dd);
        while(j<i && p[j].x < p[i].x-d){
            s.erase({p[j].y,p[j++].x});
        }
        auto l = s.lower_bound({p[i].y-d,p[i].x-d});
        auto u = s.upper_bound({p[i].y+d,p[i].x+d});
        for(auto it=l;it!=u;it++){
            t = {it->y,it->x};
            dd =min(dd, Dot(p[i]-t,p[i]-t));
        }
        s.emplace(p[i].y,p[i].x);
    }
    return dd;
}
```

10.13 FarthestPair

```
ld FarthestPair(vector<Pt>& poly){//旋轉卡尺
    ld ret=0;
    for(int i = 0, j = i+1; i<poly.size()&& j<poly.size
        (); i++){
        while( Length(poly[i]-poly[j]) <= Length(poly[i
            ]-poly[(j+1)%poly.size()]) && j!=i)
            j = (j+1) % poly.size();
        ret = max(ret, Length(poly[i]-poly[j]));
    }
    return ret;
}
```

10.14 幾何中位數

```
//回傳為到每個頂點距離和最小的點
Pt weiszfeld(const Pt *p,int n){
    double nn=n;
    Pt cur = p[0], next;
    for(int i=1;i<n;++i)
        cur.x+=p[i].x, cur.y+=p[i].y;
    cur.x/=nn, cur.y/=nn;
    double w,numerX,numerY,denomin;
    while(1){
        numerX=numerY=denomin=0;
        bool update=0;
        double d;
        for(int i=0;i<n;++i){
            d=Length(cur-p[i]);
```

```
            if(d>eps){
                w = 1.0/d;
                numerX+=w*p[i].x;
                numerY+=w*p[i].y;
                denomin+=w;
                update=1;
            }else{
                next = p[i];
                break;
        }   }
        if(update){
            next.x = numerX/denomin;
            next.y = numerY/denomin;
        }
        if(Length(cur-next)<eps) break;
        cur = next;
    }
    return next;
}
```

10.15 矩陣掃描線

```
#include <bits/stdc++.h>
#define int long long int
using namespace std;
int n, st[1000005<<2], lazy[1000005<<2], old
    [1000005<<2];
vector <tuple<int, int, int, int>> v;
vector<int> sor;
void pull(int index, int l, int r) {
    if(lazy[index]) st[index] = old[index];
    else if(l==r) st[index] = 0;
    else st[index] = st[index<<1|1]+st[index<<1];
    // printf("pull %lld~%lld, %lld\n", l, r, st[index
        ]);
    return;
}
void insert(int index, int s, int e, int l, int r, int
    k) {
    //printf("insert: range %lld~%lld, query %lld~%lld\
        n", s, e, l, r);
    if(l<=s && e<=r) {
        lazy[index] +=k;
        pull(index, s, e);
        return;
    }
    int mid = (s+e)/2;
    if(l<=mid) insert(index<<1, s, mid, l, r, k);
    if(mid<r) insert(index<<1|1, mid+1, e, l, r, k);
    pull(index, s, e);
}
void input(int index, int l, int r) {
    if(l==r) {
        old[index] = sor[l]-sor[l-1];
        return;
    }
    int mid = (l+r)/2;
    input(index<<1, l, mid);
    input(index<<1|1, mid+1, r);
    old[index] = old[index<<1] + old[index<<1|1];
    //cout<<l<<" to "<<r<<" is "<<old[index]<<endl;
    return;
}
// int diff=1000005;
signed main(){
    cin >> n;
    int l, r, d, u;
    for (int i = 0; i < n; i++){
        cin >> l >> d >> r >> u;
        // l+=diff;
        // d+=diff;
        // r+=diff;
        // u+=diff;
        sor.push_back(d);
        sor.push_back(u);
        v.push_back({l, d, u, 1});
        v.push_back({r, d, u, -1});
    }
    set<int> temp(sor.begin(), sor.end());
    sor = vector<int>(temp.begin(), temp.end());
    sort(sor.begin(), sor.end());
    for(int i=0 ; i<v.size() ; i++) {
        auto [a, b, c, k] = v[i];
        v[i] = make_tuple(a, (int)(lower_bound(sor.
            begin(), sor.end(), b)-sor.begin()), (int)(
            lower_bound(sor.begin(), sor.end(), c)-sor.
            begin()), k);
    }
    input(1, 1, sor.size()-1);
    // cout<<"get: ";
    // for(int i: sor) cout<<i<<" "; cout<<endl;
    sort(v.begin(), v.end());
    int pre=0;
    int ans=0;
    for(auto [pos, a, b, k]: v) {
        if(pre!=pos) {
            ans+=(pos-pre)*st[1];
            pre = pos;
        }
        insert(1, 1, sor.size()-1, a+1, b, k);
        // printf("now act: pos %lld,  %lld~%lld, act:
            %lld\n", pos, a+1, b, k);
        // printf("now ans: %lld\n", st[1]);
    }
    cout<<ans<<endl;
}
```

10.16 Polygon Circle intersection area

```
//給兩圓，回傳兩圓的交點，若相交回傳的兩點一樣的話要用
    dcmp判斷
vector<Pt> interCircle(Circle c1, Circle c2) {
  Pt o1 = c1.o, o2 = c2.o;
  ld r1 = c1.r, r2 = c2.r;
  if( Length( o1 - o2 ) > r1 + r2 ) return {};
  if( Length( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
      return {};
  ld d2 = ( o1 - o2 ) * ( o1 - o2 );
  ld d = sqrt(d2);
  if( d > r1 + r2 ) return {};
  Pt u = (o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
  ld A = sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2+d))
      ;
  Pt v = Pt( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
  return {u+v, u-v};
}
```

10.17 Tangent line of two circles

```
//給兩圓，求兩圓的外切線或內切線
vector<Line> go( const Circle& c1 , const Circle& c2 ,
    int sign1 ){
  // sign1 = 1 for outer tang, -1 for inter tang
  vector<Line> ret;
  double d_sq = Dot( c1.o - c2.o ,c1.o - c2.o );
  if( d_sq < eps ) return ret;
  double d = sqrt( d_sq );
  Pt v = ( c2.o - c1.o ) / d;
  double c = ( c1.r - sign1 * c2.r ) / d;
  if( c * c > 1 ) return ret;
  double h = sqrt( max( 0.0 , 1.0 - c * c ) );
  for( int sign2 = 1 ; sign2 >= -1 ; sign2 -= 2 ){
    Pt n = { v.x * c - sign2 * h * v.y ,
             v.y * c + sign2 * h * v.x };
    Pt p1 = c1.o + n * c1.r;
    Pt p2 = c2.o + n * ( c2.r * sign1 );
    if( fabs( p1.x - p2.x ) < eps and
        fabs( p1.y - p2.y ) < eps )
      p2 = p1 + Perp( c2.o - c1.o );
    ret.push_back( { p1 , p2 } );
  }
  return ret;
}
```

10.18 Circle intersection Point

```
//傳入多邊形和圓形，回傳多邊形和圓形的交集面積
ld PCIntersect(vector<Pt> v, Circle cir) {
  for(int i = 0 ; i < (int)v.size() ; ++i) v[i] = v[i]
      - cir.o;
  ld ans = 0, r = cir.r;
  int n = v.size();
  for(int i = 0 ; i < n ; ++i) {
    Pt pa = v[i], pb = v[(i+1)%n];
    if(Length(pa) < Length(pb)) swap(pa, pb);
    if(dcmp(Length(pb)) == 0) continue;
    ld s, h, theta;
    ld a = Length(pb), b = Length(pa), c = Length(pb-pa
        );
    ld cosB = (pb*(pb-pa))/a/c, B = acos(cosB);
    if(cosB > 1) B = 0;
    else if(cosB < -1) B = PI;
    ld cosC = (pa*pb)/a/b, C = acos(cosC);
    if(cosC > 1) C = 0;
    else if(cosC < -1) C = PI;
    if(a > r) {
      s = (C/2)*r*r;
      h = a*b*sin(C)/c;
      if(h < r && B < PI/2) s -= (acos(h/r)*r*r - h*
          sqrt(r*r-h*h));
    }
    else if(b > r) {
      theta = PI - B - asin(sin(B)/r*a);
      s = 0.5*a*r*sin(theta) + (C-theta)/2*r*r;
    }
    else s = 0.5*sin(C)*a*b;
    ans += abs(s)*dcmp(v[i]^v[(i+1)%n]);
  }
  return abs(ans);
}
```

10.19 CircleCover

```
#define N 100
#define D long double
struct CircleCover{//O(N^2logN)
  int C; Circle c[ N ]; //填入C(圓數量),c(圓陣列,0base)
  bool g[ N ][ N ], overlap[ N ][ N ];
  // Area[i] : area covered by "at least" i circles
  D Area[ N ];
  void init( int _C ){ C = _C; }//總共 _c 個圓
  bool CCinter( Circle& a , Circle& b , Pt& p1 , Pt& p2
      ){
    Pt o1 = a.o , o2 = b.o;
    D r1 = a.r , r2 = b.r;
    if( Length( o1 - o2 ) > r1 + r2 ) return {};
    if( Length( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
        return {};
    D d2 = ( o1 - o2 ) * ( o1 - o2 );
    D d = sqrt(d2);
    if( d > r1 + r2 ) return false;
    Pt u=(o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
    D A=sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2+d));
    Pt v=Pt( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
    p1 = u + v; p2 = u - v;
    return true;
  }
  struct Teve {
    Pt p; D ang; int add;
    Teve() {}
    Teve(Pt _a, D _b, int _c):p(_a), ang(_b), add(_c){}
    bool operator<(const Teve &a)const
    {return ang < a.ang;}
  }eve[ N * 2 ];
  // strict: x = 0, otherwise x = -1
  bool disjuct( Circle& a, Circle &b, int x )
  {return dcmp( Length( a.o - b.o ) - a.r - b.r ) > x;}
  bool contain( Circle& a, Circle &b, int x )
  {return dcmp( a.r - b.r - Length( a.o - b.o ) ) > x;}
  bool contain(int i, int j){
    /* c[j] is non-strictly in c[i]. */
    return (dcmp(c[i].r - c[j].r) > 0 ||
            (dcmp(c[i].r - c[j].r) == 0 && i < j) ) &&
                contain(c[i], c[j], -1);
  }
  void solve(){
    for( int i = 0 ; i <= C + 1 ; i ++ )
      Area[ i ] = 0;
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        overlap[i][j] = contain(i, j);
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        g[i][j] = !(overlap[i][j] || overlap[j][i] ||
                    disjuct(c[i], c[j], -1));
    for( int i = 0 ; i < C ; i ++ ){
      int E = 0, cnt = 1;
      for( int j = 0 ; j < C ; j ++ )
        if( j != i && overlap[j][i] )
          cnt ++;
      for( int j = 0 ; j < C ; j ++ )
        if( i != j && g[i][j] ){
          Pt aa, bb;
          CCinter(c[i], c[j], aa, bb);
          D A=atan2(aa.y - c[i].o.y, aa.x - c[i].o.x);
          D B=atan2(bb.y - c[i].o.y, bb.x - c[i].o.x);
          eve[E ++] = Teve(bb, B, 1);
          eve[E ++] = Teve(aa, A, -1);
          if(B > A) cnt ++;
        }
      if( E == 0 ) Area[ cnt ] += PI * c[i].r * c[i].r;
      else{
        sort( eve , eve + E );
        eve[E] = eve[0];
        for( int j = 0 ; j < E ; j ++ ){
          cnt += eve[j].add;
          Area[cnt] += (eve[j].p ^ eve[j + 1].p) * 0.5;
          D theta = eve[j + 1].ang - eve[j].ang;
          if (theta < 0) theta += 2.0 * PI;
          Area[cnt] +=
            (theta - sin(theta)) * c[i].r*c[i].r * 0.5;
}}}};
```

10.20 半平面交

```
//O(NlgN)
// for point or line solution, change > to >=
bool onleft(Line L, Pt p) {
  return dcmp(L.v^(p-L.a)) > 0;
} // segment should add Counterclockwise
// assume that Lines intersect
// 傳入每條方程式的兩點方程式
// 回傳形成的凸多邊形頂點
// (半平面為像量 ab 的逆時針方向)
//注意題目輸入的點要是逆時針排序
vector<Pt> HPI(vector<Line>& L) {
  sort(L.begin(), L.end()); // sort by angle
  for(auto l:L){
    cerr<<l.a.x<<" "<<l.a.y<<" "<<l.b.x<<" "<<l.b.y<<"
        "<<l.ang<<'\n';
  }
  int n = L.size(), fir, las;
  Pt *p = new Pt[n];
  Line *q = new Line[n];
  q[fir=las=0] = L[0];
  for(int i = 1 ; i < n ; i++) {
    while(fir < las && !onleft(L[i], p[las-1])) las--;
    while(fir < las && !onleft(L[i], p[fir])) fir++;
    q[++las] = L[i];
    if(dcmp(q[las].v^q[las-1].v) == 0) {
      las--;
      if(onleft(q[las], L[i].a)) q[las] = L[i];
    }
    if(fir < las) p[las-1] = getLineIntersect(q[las-1],
        q[las]);
  }
  while(fir < las && !onleft(q[fir], p[las-1])) las--;
  if(las-fir <= 1) return {};
  p[las] = getLineIntersect(q[las], q[fir]);
  int m = 0;
  vector<Pt> ans(las-fir+1);
  for(int i = fir ; i <= las ; i++) ans[m++] = p[i];
  return ans;
}
```

10.21 PolygonUnion

```
//O(N^2lgN)
//傳入二維vector，每個vector代表一個多邊形，每個多邊形的點必須按照順時針或逆時針順序
//回傳聯集多邊形的面積
ld tri(Pt o, Pt a, Pt b){ return (a-o) ^ (b-o);}
double polyUnion(vector<vector<Pt>> py){ //py[0~n-1] must be filled
  int n = py.size();
  int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td, area;
  vector<pair<double,int>> c;
  for(i=0;i<n;i++){
    area=py[i][py[i].size()-1]^py[i][0];
    for(int j=0;j<py[i].size()-1;j++) area+=py[i][j]^py[i][j+1];
    if((area/=2)<0) reverse(py[i].begin(),py[i].end());
    py[i].push_back(py[i][0]);
  }
  for(i=0;i<n;i++){
    for(ii=0;ii+1<py[i].size();ii++){
      c.clear();
      c.emplace_back(0.0,0); c.emplace_back(1.0,0);
      for(j=0;j<n;j++){
        if(i==j) continue;
        for(jj=0;jj+1<py[j].size();jj++){
          ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]));
          tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj+1]));
          if(ta==0 && tb==0){
            if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[i][ii])>0&&j<i){
              c.emplace_back(segP(py[j][jj],py[i][ii],py[i][ii+1]),1);
              c.emplace_back(segP(py[j][jj+1],py[i][ii],py[i][ii+1]),-1);
            }
          }else if(ta>=0 && tb<0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c.emplace_back(tc/(tc-td),1);
          }else if(ta<0 && tb>=0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c.emplace_back(tc/(tc-td),-1);
      } } }
      sort(c.begin(),c.end());
      z=min(max(c[0].first,0.0),1.0); d=c[0].second; s=0;
      for(j=1;j<c.size();j++){
        w=min(max(c[j].first,0.0),1.0);
        if(!d) s+=w-z;
        d+=c[j].second; z=w;
      }
      sum+=(py[i][ii]^py[i][ii+1])*s;
  } }
  return sum/2;
}
```

10.22 PolygonCover

```
//傳入二維vector，每個vector代表一個多邊形，每個多邊形的點必須按照順時針或逆時針順序
// return Area[i] : area covered by "at least" i polygon
//O(N^2lgN)
vector<double> PolyUnion(const vector<vector<Pt>> &P) {
    auto reorder = [&](vector<Pt> &v) { //排序成逆時針且最左下角的點在最前面
        rotate(v.begin(), min_element(all(v)), v.end());
        if (Ori(v[0], v[1], v[2]) < 0) {
            reverse(all(v));
        }
    };
    for(auto &i: py){ reorder(i);}
    const int n = P.size();
    vector<double> Area(n + 1);
    vector<Line> Ls;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < P[i].size(); j++)
            Ls.push_back({P[i][j], P[i][(j + 1) % P[i].size()]});
    auto cmp = [&](Line &l, Line &r) {
        Pt u = l.b - l.a, v = r.b - r.a;
        if (argcmp(u, v)) return true;
        if (argcmp(v, u)) return false;
        return PtSide(l.a, r) < 0;
    };
    sort(all(Ls), cmp);
    for (int l = 0, r = 0; l < Ls.size(); l = r) {
        while (r < Ls.size() and !cmp(Ls[l], Ls[r])) r++;
        Line L = Ls[l];
        vector<pair<Pt, int>> event;
        for (auto &ls : Ls) {
            Pt c = ls.a, d = ls.b;
            if (Sgn((L.a - L.b) ^ (c - d)) != 0) {
                int s1 = PtSide(c, L) == 1;
                int s2 = PtSide(d, L) == 1;
                if (s1 ^ s2) event.emplace_back(LineInter(L, {c, d}), s1 ? 1 : -1);
            } else if (PtSide(c, L) == 0 and Sgn((L.a - L.b) * (c - d)) > 0) {
                event.emplace_back(c, 2);
                event.emplace_back(d, -2);
            }
        }
        sort(all(event), [&](auto i, auto j) {
            return (L.a - i.ft) * (L.a - L.b) < (L.a - j.ft) * (L.a - L.b);
        });
        int cov = 0, tag = 0;
        Pt lst{0, 0};
        for (auto [p, s] : event) {
            if (cov >= tag) {
                Area[cov] += lst ^ p;
                Area[cov - tag] -= lst ^ p;
            }
            if (abs(s) == 1) cov += s;
            else tag += s / 2;
            lst = p;
        }
    }
    for (int i = n - 1; i >= 0; i--) Area[i] += Area[i + 1];
    for (int i = 1; i <= n; i++) Area[i] /= 2;
    return Area;
};
```

11 特殊題目

11.1 包含子字串計數

```
// * 給一個字串s
// * 求長度為len且有包含s的字串有幾種
// * 呼叫solve(s, len)
const int len = 1005;
int aut[len][26];
int dp[len][len];
const int mod = 1e9+7;
void prefix(string &s, vector<int> &pi) {
    for(int i=1, j=0 ; i<s.size() ; i++) {
        while(j>0 && s[i]!=s[j]) j = pi[j-1];
        if(s[i]==s[j]) j++;
        pi[i] = j;
    }
}
void automata(string &s, vector<int> &pi) {
    for(int i=0 ; i<s.size() ; i++) {
        for(int c=0 ; c<26 ; c++) {
            if(i>0 && c+'A' != s[i]) aut[i][c] = aut[pi[i-1]][c];
            else aut[i][c] = i + (c + 'A'==s[i]);
        }
    }
}
int quai(int x, int n) {
    if(n==0) return 1;
```

```
    int mid = quai(x,n/2);
    mid = mid*mid%mod;
    if(n&1) return mid*x%mod;
    return mid;
}
int solve(string s, int len) {
    vector<int> pi(s.size(), 0);
    prefix(s, pi);
    automata(s, pi);
    int n = s.size(), ans = quai(26, len);
    dp[0][0] = 1;
    for(int i=0 ; i<len ; i++) {
        for(int j=0 ; j<n ; j++) {
            for(int c=0 ; c<26 ; c++) {
                dp[i+1][aut[j][c]] += dp[i][j];
                dp[i+1][aut[j][c]] %= mod;
            }
        }
    }
    for(int i=0 ; i<n ; i++) ans = (ans - dp[len][i] +
        mod)%mod;
    return ans;
}
```

11.2 三維偏序

```
// vec
// {{a, b, c},
//  {a, b, c},
//     ...
//  {a, b, c}}
// 貼上 BIT 模板
// 三維偏序
// a <= a, b <= b, c <= c
map<vector<int>, int> cnt;
int cdq(vector<vector<int>> &vec, int l, int r) {
    if(l==r) return 0;
    int mid = l+r>>1;
    int ans = cdq(vec, l, mid)+cdq(vec, mid+1, r);
    vector<vector<int>> temp;
    for(int i=l, j=mid+1 ; i<=mid || j<=r ;) {
        while(i<=mid && (j>r || vec[i][1] <= vec[j][1])
            ) {bit.add(vec[i][2],cnt[vec[i]]); temp.
            push_back(vec[i++]);}
        if(j<=r) {
            temp.push_back(vec[j]);
            ans += bit.query(vec[j][2]);
        }
    }
    for(int i=l ; i<=mid ; i++) bit.add(vec[i][2],-cnt[
        vec[i]]);
    for(int i=l ; i<=r ; i++) vec[i] = temp[i-l];
    return ans;
}
int solve(vector<vector<int>> &vec) {
    bit.init(2e5+5);
    for(vector<int> v: vec) cnt[v]++;
    sort(vec.begin(), vec.end());
    vec.erase(unique(vec.begin(), vec.end()), vec.end()
        );
    return cdq(vec, 0, vec.size()-1);
}
```

12 Python

12.1 時間日期 Datetime

```
from datetime import datetime, date, time, timedelta

# 閏年 2024
# 平年 2023

# 日期相減
start_time = datetime(2024, 8, 31)
end_time = datetime(2024, 9, 1)
delta = end_time - start_time
print(delta.days)
# 1

# 時間相減
start_time = datetime(1000, 1, 1, 14, 30)
end_time = datetime(1000, 1, 1, 16, 50)
delta = end_time - start_time
print(delta) # 2:20:00
print(delta.total_seconds()) # 8400.0


# 時間日期相加
specific_date = datetime(1000, 1, 1)
new_date = specific_date + timedelta(days=3, hours=1,
    minutes=1)
print(new_date)
# 1000-01-04 01:01:00
print(new_date.year, new_date.month, new_date.day,
    new_date.hour, new_date.minute, new_date.second)
# 1000 1 4 1 1 0
```

12.2 Decimal

```
from decimal import Decimal, getcontext, ROUND_FLOOR
getcontext().prec = 250 # set precision (MAX_PREC)
getcontext().Emax = 250 # set exponent limit (MAX_EMAX)
getcontext().rounding = ROUND_FLOOR # set round floor
itwo,two,N = Decimal(0.5),Decimal(2),200
pi = angle(Decimal(-1))
```

12.3 Fraction

```
from fractions import Fraction
import math
"""專門用來表示和操作有理數，可以進行算"""
frac1 = Fraction(1)    # 1/1
frac2 = Fraction(1, 3) # 1/3
frac3 = Fraction(0.5)  # 1/2
frac4 = Fraction('22/7') # 22/7
frac5 = Fraction(8, 16) # 自動約分為 1/2
frac9 = Fraction(22, 7)
frac9.numerator   # 22
frac9.denominator # 7
x = Fraction(math.pi)
y2 = x.limit_denominator(100) # 分母限制為 100
print(y2)  # 311/99
float(x) #轉換為浮點數
```

12.4 正則表達式 re

```
import re

# \d  0~9
# \D  非0~9
# \s  空白字符
# \S  非空白字符
# ^   開頭
# $   結尾
# re*   0個或多個
# re+   1個或多個
# re?   0個或1個
# re{n} 恰好n次

s = 'a1bb2c2'
print(re.search(r'(?<=\d)[a-z](?=\d)', s).group())
# b
# 匹配前面和後面都要是數字的a~z字元

s = 'a1a a2b b3b'
print(re.search(r'(?<!a)\d(?!a)', s).group())
# 3
# 匹配前後都不是a的數字

s = 'aBc'
print(re.search(r'abc', s, re.I).group())
# aBc
# re.I 忽略大小寫
# re.S 任意字符 . 不受換行(\n)和空白限制
```

```
s = 'abbbbb'
print(re.search(r'ab+?', s).group())
# ab
# 非貪婪用法，匹配ab+但長度越短越好

s = 'abcde'
print(re.findall(r'[a-z]', s))
# ['a', 'b', 'c', 'd', 'e']

s = 'abbbba'
print(re.sub('(?<=a).*(?=a)', '', s))
# aa
# 把.*的部分替換成空字串
```

12.5 Misc

```
# 轉為高精度整數比，(分子, 分母)
x=0.2
x.as_integer_ratio()  # (8106479329266893,
    9007199254740992)
x.is_integer()        # 判斷是否為整數
x.__round__()         # 四捨五入
int(eval(num.replace("/","//"))) # parser string num
```