

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	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	ZKW Flow	3
3.3	Hungarian	3
3.4	KM	3
4	Graph	4
4.1	BCC	4
4.2	SCC	4
4.3	2SAT	4
4.4	MaximalClique	5
4.5	MaximumClique	5
4.6	Minimum Mean Cycle	5
4.7	Dominator Tree	6
5	Math	6
5.1	Formulas	6
5.2	Quick Pow	6
5.3	Mat quick Pow	6
5.4	Primes Table	7
5.5	Factor Table	7
5.6	Catalan Number	7
5.7	Miller Rabin	7
5.8	PollarRho	7
5.9	PrimeFactorO(logn)	7
5.10	(1)mul	7
5.11	Josephus Problem	7
5.12	Harmonic Sum	7
6	Data Structure	8
6.1	BIT	8
6.2	Sparse Table	8
6.3	Segment Tree	8
6.4	Time Segment Tree	9
6.5	Treap	9
6.6	PBDS	9
7	String	9
7.1	SA	9
7.2	KMP	10
7.3	Single Hash	10
7.4	Double Hash	10
7.5	Trie	11
7.6	Z value	11
7.7	MinRotation	11
7.8	Manacher	11
7.9	PalTree	12
8	Python	12
8.1	Decimal	12
8.2	Fraction	12
8.3	Misc	12

1 Basic

1.1 Default code

```

1 #include<bits/stdc++.h>
1 #include<chrono> // for timing
2 #pragma GCC optimize("O3,unroll-loops")
2 #pragma target optimize("avx2,bmi,bmi2,lzcnt,popcnt")
2 #define IO ios_base::sync_with_stdio(0);cin.tie(0);cout
2 .tie(0);
2 #define pii pair<int,int>
2 #define ft first
2 #define sd second
2 #define int long long
2 #define double long double
2 #define PI acos(-1)
3 #define SZ(x) (int)x.size()
3 #define all(v) (v).begin(), (v).end()
3 #define _for(i,a,b) for(int i=(a);i<(b);++i)
4 using namespace std;
4 template<typename T>
4 ostream& operator<<(ostream& os,const vector<T>& vn){
4     for(int i=0;i<vn.size();++i)os<<vn[i]<<" ";
5     return os;
5 }
6 template<typename T>
6 ostream& operator<<(ostream& os,const set<T>& vn){
6     for(typename set<T>::iterator it=vn.begin();it!=vn.
6         end();++it)os<<*it<<" ";
6     return os;
7 }
7 mt19937 mt(hash<string>{}("Mashu_AC_Please")); //mt();
7 // mt19937 mt(chrono::steady_clock::now().
7     time_since_epoch().count());
7 // g++ a.cpp -Wall -Wshadow -fsanitize=undefined -o a.
7     exe
7 // ./a.exe
7 const int MXN=2e5+5;
8 const int INF=INT_MAX;
8 void sol() {}
8 signed main() {
8     // auto start=chrono::high_resolution_clock::now();
8     // #ifdef LOCAL
8     // freopen("input.txt","r",stdin);
8     // freopen("output.txt","w",stdout);
9     // #endif
9     IO
9     int t=1;
9     cin>>t;
10 while(t--) {sol();}
10 // auto stop = chrono::high_resolution_clock::now()
10 ;
10 // auto duration = chrono::duration_cast<chrono::
10     milliseconds>(stop - start);
10 // cerr<<"Time:"<<duration.count()<<" ms\n";
11 }

```

1.2 Misc

```

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
__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

```

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 hashes{
    size_t operator()(const Type &o) const {
        return ((hash<int>()(o.x)^(hash<string>()(o.y)
            <<1))>>1);
    }
};
//unordered_map<Type,int,hashes> 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;
}

```

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 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 ZKW Flow

```

//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 $O(V^2E^2)$ 
// 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
#define ll long long
struct zkwwflow{
    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 Hungarian

```

//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 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;

```

4.2 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;

```

4.3 2SAT

有N個 boolean 變數 $a_1 \dots a_N$
 ex: 滿足 $(\neg a_1 \text{ or } a_2) \text{ and } (a_2 \text{ or } a_3) \text{ and } (\neg a_3 \text{ or } \neg a_4)$ 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 $(p \text{ and } q)$ ，則建兩條邊
 $\text{not } p \rightarrow q$ (if p為false 則 q必為true)
 $\text{not } q \rightarrow p$ (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 $(a_i \text{ 和 } \neg a_i)$ 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 $\neg a_i$)

[illegible]

4.4 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;
```

4.5 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;

```

4.6 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;

```

4.7 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;

```

5 Math

5.1 Formulas

```

//五次方冪次和
a(n) = n^2*(n+1)^2*(2*n^2+2*n-1)/12.

```

5.2 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);

```

5.3 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;
}

```

5.4 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);
        }
    }
}
```

5.5 Factor Table

```
int arr[MXN];
void init(){
    for(int i=1;i<MXN;++i) for(int j=i;j<MXN;j+=i) arr[j]++;
}
```

5.6 Catalan Number

```
// O(N), 要記得開Long Long 跟設定 MOD
cat[0]=1; cat[1]=1;
for(int i=1 ; i<N ; i++) {
    cat[i+1] = cat[i]*(i*4+2)%MOD*qpow(i+2, MOD-2)%MOD;
}
```

5.7 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 : pimes <= 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=myspow(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;
}
```

5.8 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;
} }
```

5.9 PrimeFactorO(logn)

```
#define i64 __int64
vector<i64> ret;
void fact(i64 x) {
    if (miller_rabin(x)) {
        ret.push_back(x);
        return;
    }
    i64 f = pollard_rho(x);
    fact(f); fact(x/f);
}
```

5.10 O(1)mul

```
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;
}
```

5.11 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);
}
```

5.12 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]
    //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 Data Structure

6.1 BIT

```

//注意值域
#define lowbit(x) (x & -x)
const int N = 1e5+5;
int bit[N];
struct BIT {
    int n;
    void init(int n){this->n = n;}
    void update(int x, int val) {
        for (; x <= n; x += lowbit(x))
            bit[x] += val;
    }
    int query(int x) {
        int res = 0;
        for (; x; x -= lowbit(x))
            res += bit[x];
        return res;
    }
    int query(int L, int R) { return query(R) - query(L - 1); }
}

```

6.2 Sparse Table

```

//st[i][j]表示[i,i+2^j-1]的最值,區間最大長度為log2(n)
//i為base
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;

```

6.3 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;

```

6.4 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;
    }
}

```

6.5 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);
}

```

6.6 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)

```

7 String

7.1 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, nmzx = -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[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[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]]]=nmzx+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmzx + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[p[nsa[i]]]] = p[nsa[i]]);
}
}
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";
}
}

```

7.2 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;
}

```

7.3 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;
}

```

```
}

```

7.4 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};
}

```

7.5 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);
    }
}

```

7.6 Z value

```
// O(n)
//z[i] = lcp(s[1...], s[i...])
//1base
int z[MXN];
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;
        }
    }
}

```

7.7 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;
}

```

7.8 Manacher

```
// 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 = "abac" -> "@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+1-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];
}

```

7.9 PalTree

```
// 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?1-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;
```

8.3 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
```

8 Python

8.1 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))
```

8.2 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
```

[illegible]

[illegible]