IUST - 8 silandr - Notebook

Contents

1 Sublime Config

```
"shell_cmd": "g++ ${file_name} && gnome-terminal -- bash -c './a.out;
    read'"
```

2 Template

```
#pragma GCC optimize("Ofast,unroll-loops,fast-math")
#include<bits/stdc++.h>
using namespace std;
typedef long long ll ;
#define pll pair<ll , ll >
#define all(x) (x).begin(),(x).end()
#define SZ(x) (ll)(x).size()
#define X   first
#define Y   second
#define mp  make_pair
#define pii pair<int , int>
#define vec vector
#define file_io freopen("input.txt", "r", stdin);freopen("output.txt",
     "w", stdout);
#define migmig ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define pb push_back
#define ld long double
// BIG p : 1000000000000037 , 100000000003
ll poww(ll a, ll b, ll md) {
    return (!b ? 1 : (b & 1 ? a * poww(a * a % md, b / 2, md) % md :
        poww(a * a % md, b / 2, md) % md));
}
const int maxn = 1000*100+5 ;
const ll inf = 9223372036854775807 ;
const ll mod = 1e9 + 7 ;
const int lg = 20 ;

int main()
{
        migmig ;

}
```

3 C++

3.1 STL

```
Bitset:
bitset<4> b3{"0011"};
b3.set(position_t(1), true);
b3[2] = true;

Custom Sorted Set:
struct AbsoluteValueComparator {
    bool operator()(int a, int b) const
    {
        return abs(a) < abs(b);
    }
};
set<int, AbsoluteValueComparator> mySet;

lower_bound on vector:
ll pos = lower_bound(Rs.begin() , Rs.end() , value) - Rs.begin() ;

Compress:
sort(all(v))
v.erase(unique(all(v)) , v.end())
```

3.2 Compress

```
sort(all(v)) ; v.resize(unique(all(v)) - v.begin())
```

3.3 Random

```
mt19937 Rnd;
int Rand(int l, int r)
{
    return (Rnd() % (r - l + 1) + l);
}
Rnd.seed(time(0));
```

3.4 Set Intersection

```
#include<bits/stdc++.h>

int main () {
  int first[] = {5,10,15,20,25}; // Should be sorted
  int second[] = {10, 20, 30, 40, 50}; // Should be sorted
  std::vector<int> v(10);                       // 0  0  0  0  0  0  0
       0  0  0
  std::vector<int>::iterator it;

  it=std::set_intersection (first, first+5, second, second+5, v.begin
      ()); // 10 20 0  0  0  0  0  0  0  0
  v.resize(it-v.begin());                       // 10 20
  return 0;
}
```

3.5 Ordered Set

```
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template <class T> using Tree = tree<T , null_type, less<T> ,
    rb_tree_tag , tree_order_statistics_node_update>;
order_of_key
find_by_order
```

3.6 Overload

```cpp
#include <iostream>
using namespace std;

struct MyInt {
    int value;
    MyInt(int v = 0) : value(v) {}

    MyInt operator+(const MyInt& rhs) const { return MyInt(value + rhs
        .value); }
    // Same as - * /

    MyInt& operator++() { ++value; return *this; } // Prefix increment
    MyInt& operator--() { --value; return *this; } // Prefix decrement

    MyInt& operator+=(const MyInt& rhs) { value += rhs.value; return *
        this; }
    // same as -= *= /= %=

    bool operator==(const MyInt& rhs) const { return value == rhs.
        value; }
    // Same as != < <= > >=

    MyInt operator&(const MyInt& rhs) const { return MyInt(value & rhs
        .value); }
    // Same as | ^ ~

    MyInt& operator&=(const MyInt& rhs) { value &= rhs.value; return *
        this; }
    // Same as |= ^= <<= >>=

    MyInt operator<<(const int shift) const { return MyInt(value <<
        shift); }
    MyInt operator>>(const int shift) const { return MyInt(value >>
        shift); }

    // Address-of operators
    MyInt* operator&() { return this; }
    const MyInt* operator&() const { return this; }
};

int main() {
    return 0;
}
```

4 Implementation

4.1 Token

```cpp
vector<string> token(string& s, string& del)
{
    // string inp = "Hello,World;This|is.GeeksForGeeks pam";
    // string del = ",;.| ";
    // vector<string> ans = token(inp, del);
    vector<string> ans;
    int start = 0, last = 0;
    while ((last = s.find_first_of(del, start)) != string::npos)
        {
        if (last != start)
            ans.pb(s.substr(start, last - start));
        start = last + 1;
    }
    if (start != s.length())
        ans.pb(s.substr(start));
    return ans;
}
```

4.2 Convert(String-Double)

```cpp
double d = 3.0;
string str = to_string(d);
double D = stod(str)
```

5 Math - Number Theory

5.1 Combine

```cpp
const ll mod = 1e9 + 7;

ll power(ll a, ll b)
{
    if(b == 0) return 1;
    return power((a * a) % mod, b / 2)  * (b % 2 ? a : (ll)1) % mod;
}

ll comb(ll k, ll n)
{
    ll ans = 1;
    for(int i = 1; i <= k; i++) ans = ans * i % mod;
    ans = power(ans, mod - 2);
    for(int i = n; i > n - k; i--) ans = ans * i % mod;
    return ans;
}
```

5.2 Most Divisors

```
<= 1e2: 60 with 12 divisors
<= 1e3: 840 with 32 divisors
<= 1e4: 7560 with 64 divisors
<= 1e5: 83160 with 128 divisors
<= 1e6: 720720 with 240 divisors
<= 1e7: 8648640 with 448 divisors
<= 1e8: 73513440 with 768 divisors
<= 1e9: 735134400 with 1344 divisors
<= 1e10: 6983776800 with 2304 divisors
<= 1e11: 97772875200 with 4032 divisors
<= 1e12: 963761198400 with 6720 divisors
<= 1e13: 9316358251200 with 10752 divisors
<= 1e14: 97821761637600 with 17280 divisors
<= 1e15: 866421317361600 with 26880 divisors
<= 1e16: 8086598962041600 with 41472 divisors
<= 1e17: 74801040398884800 with 64512 divisors
<= 1e18: 897612484786617600 with 103680 divisors

//number of primes
30: 10
```

```
60: 17
100: 25
1000: 168
10000: 1229
100000: 9592
1000000: 78498
10000000: 664579
```

5.3 Number Theory

```
ll _gcd(ll a , ll b)
{
        if (a < b) swap(a, b);
    if(b==0) return a;
    a%=b;
    return _gcd(b,a);
}
// BIG p : 1000000000000037 , 100000000003s
ll poww(ll a, ll b, ll md) {
    return (!b ? 1 : (b & 1 ? a * poww(a * a % md, b / 2, md) % md :
        poww(a * a % md, b / 2, md) % md));
}
```

5.4 NTT

```
inline void fix(ll &x){
        if (x<0) x+=mod;
        if (x>=mod) x-=mod;
}
ll powmod(ll a, ll b){
        ll res=1;
        for (; b; b>>=1, a=a*a%mod) if (b&1) res=res*a%mod;
        return res;
}
ll nCr(int n, int r){
        if (r<0 || r>n) return 0;
        return F[n]*I[r]%mod*I[n-r]%mod;
}

void NTT(ll* F, int n, bool inv){
        int lg=ceil(log2(n));
        n=1<<lg;
        //debug2(n, lg)
        for (int i=0; i<n; i++) rev[i]=(rev[i>>1]>>1) | ((i&1)<<(lg-1)
            );
        for (int i=0; i<n; i++) if (i<rev[i]) swap(F[i], F[rev[i]]);
        for (int len=1; len<n; len<<=1){
            ll wn=powmod(3, (mod-1)/(2*len));
            if (inv) wn=powmod(wn, mod-2);
            for (int i=0; i<n; i+=2*len){
                ll w=1;
                for (int j=i; j<i+len; j++){
                    ll x=(F[j] + F[j+len]*w)%mod;
                    ll y=(F[j] - F[j+len]*w)%mod;
                    F[j]=x;
                    F[j+len]=y;
                    w=w*wn%mod;
                }
            }
        }
        if (inv){
            ll invn=powmod(n, mod-2);
            for (int i=0; i<n; i++) F[i]=F[i]*invn%mod;
        }
}
```

6 DP

6.1 SOS

```
// iterate over all the masks
for (int mask = 0; mask < (1<<n); mask++){
        F[mask] = A[0];
    // iterate over all the subsets of the mask
    for(int i = mask; i > 0; i = (i-1) & mask){
        F[mask] += A[i];
    }
```

6.2 Matrice

```
struct MAT {
        long long  n , m , A[200][200] ;
        MAT (const long long  &n2 ,const long long  &m2 ){
            n = n2 , m = m2 ;
            for (int i = 0 ; i < n ; i++)
                for (int j = 0 ; j < m ; j++)
                    A[i][j] = 0 ;
        }
        MAT operator * (const MAT &B){
            MAT C = MAT(n , m) ;
            for (int i = 0 ; i < n ; i++){
                for (int j = 0 ; j < m ; j++){
                    for (int k = 0 ; k < m ; k++)
                        C.A[i][j] = (C.A[i][j] + 1ll *
                            A[i][k] * B.A[k][j] % (
                            mod-1)) % (mod - 1) ;
                }
            }
            return (C) ;
        }
        void eq(const ll (&a)[5][5] , int n , int m )
        {
            for(int i = 0 ; i < n ; i ++ )
            {
                for(int j = 0 ; j < m ; j ++ )
                {
                    A[i][j] = a[i][j] ;
                }
            }
        }

        MAT operator + (const MAT &B){
            MAT C = MAT(n , m) ;
            for (int i = 0 ; i < n ; i++){
                for (int j = 0 ; j < m ; j++)
                    C.A[i][j] = (A[i][j] + B.A[i][j]) %
                        mod  ;
```

```
                }
                return (C) ;
        }
        void  printt (){
                for (int i = 0 ; i < n ; i++){
                        for (int j = 0 ; j < m ; j++ )
                                cout<<A[i][j]<<" ";
                        cout<<"\n" ;
                }
        }
        MAT operator ^ (long long  x){
                MAT R = MAT(n , m) ;
                for (int i = 0 ; i < n ; i++)
                        R.A[i][i] = 1 ;
                MAT T = MAT(n , m) ;
                for (int i = 0 ; i < n ;i++)
                        for (int j = 0 ;j < m ;j++)
                                T.A[i][j] = A[i][j] ;
                while (x > 0){
                        if (x%2)
                                R = R * T ;
                        T = T * T ;
                        x/=2 ;
                }
                return R ;
        }
};
```

7 Data Structure

7.1 Fenwick 2D

```
map<pair<int,int>,int>tree;
int MAXX = 1e5 + 5, MAXY = 1e5+5;

void update( int x, int y, int val )
{
    while( x <= MAXX )
    {
        int now = y;
        while( now <= MAXY )
        {
            tree[{x,now}] += val;
            now += now&(-now);
        }
        x += x&(-x);
    }
}

int get( int x, int y )
{
    int ans = 0;
    while( x > 0 )
    {
        int now = y;
        while( now > 0 )
        {
            if( tree.find({x,now}) != tree.end() )
                ans += tree[{x,now}];
            now -= now&(-now);
        }
        x -= x&(-x);
    }
    return ans;
}
```

7.2 Fenwick Pro

```
inline void add(int ind , ll i , ll v){
        for(i ++ ; i < mod ; i += i & -i){
                fen[ind][i] += v ;
                //cout <<" bomb " << i << endl ;
        }
}
inline void ADD(ll l , ll r , ll x){
        add(0 , l , x) ;
        add(0 , r+1 , -1 * x) ;
        add(1 , l , x * (l - 1ll)) ;
        add(1 , r + 1 , -1 * x * r) ;
}
ll get(int ind , ll i){
        ll res =0 ;
        for(i ++ ; i > 0 ; i -= i & -i){
                if(fen[ind].find(i) != fen[ind].end())
                        res += fen[ind][i] ;
        }
        return res ;
}
ll GET(ll i){
        return get(0 , i) * i - get(1 , i) ;
}
```

7.3 HLD

```
void predfs(int v , int p){
        sz[v] = 1 ; if(p != -1) h[v] = h[p] + 1; par[v] = p ;
        for(auto u : adj[v]){
                if(u != p){
                        predfs(u , v) ;
                        if(bc[v] == -1 || sz[u] > sz[bc[v]]) bc[v] = u
                                ;
                        sz[v] += sz[u] ;
                }
        }
}
void dfs(int v , int p , int id){
        t.pb(v) ;
        st[v] = cnt ;
        cnt++ ;
        if(id) hd[v] = v ;
        else hd[v] = hd[p] ;
        if(bc[v] != -1){
                dfs(bc[v] , v , 0) ;
        }
        for(auto u : adj[v]) {
                if(u != p && u != bc[v]) dfs(u , v , 1) ;
        }
}
```

```
void getp(int v){
        while(v != -1){
                path.pb({st[hd[v]] , st[v] + 1}) ;
                v = par[hd[v]] ;
        }
}
```

7.4 MO's Algo

```
bool cmp(Query A, Query B)
{
  if (A.l / S    = B.l / S) return A.l / S < B.l / S;
  return A.r > B.r
}
```

8 String

8.1 Hash

```
ll hsh(int l , int r){
        return (h[r] - h[l] * pt[r-l] % mod + mod) % mod ;
}
```

8.2 Z-Function

```
const ll maxn = 2e5 + 10;
ll z[maxn];
//  Pattern matching : maximum character matching of start at index i
    to prefix
//  text && pattern -----> s = pattern + '$' + text
// z[i] = number of match prifix start index i

void z_function(string s)
{
    ll left = 0, right = 0;
    for(int i = 1; i < s.size(); i++)
        if(i <= right && z[i - left] < right - i + 1)
            z[i] = z[i - left];
        else
        {
            if(i <= right)
                left = i;
            else
                left = right = i;
            while(right < s.size() && s[right] == s[right - left])
                right++;
            z[i] = right - left;
            right--;
        }
}
```

8.3 Ahoo

```
//Aho
void AddTrie(){
    int v=0;
    for (char ch:S){
        if (!C[v][ch-'a']) C[v][ch-'a']=++ts;
        v=C[v][ch-'a'];
    }
    ted[v]++;
}
void BuildAho(){
    for (int i=0; i<SGM; i++) if (C[0][i]) Q[R++]=C[0][i];
    while (L<R){
        int v=Q[L++];
        ted[v]++;
        for (int i=0; i<SGM; i++){
            if (!C[v][i]) C[v][i]=C[F[v]][i];
            else{
                F[C[v][i]]=C[F[v]][i];
                Q[R++]=C[v][i];
            }
        }
    }
}//
for(int i = 1 ; i <= ts  ; i++ ) adj[f[i]].pb(i) ;
```

8.4 SuffixArray

```
ll n , p[maxn] , h[lg][maxn] , sa[maxn] , lcp[maxn] , mark[maxn] , ps[
    maxn] , l[maxn] , r[maxn] , cp , ans ;
string s , t ;
stack<int> st ;
bool cmp(int i , int j){
        if(h[cp-1][i] < h[cp-1][j]) return true;
        if(h[cp-1][i] > h[cp-1][j]) return false ;
        if(i + (1 << (cp-1)) > n + 1 || j + (1 << (cp-1)) > n + 1)
            return i < j ;
        return h[cp-1][i + (1 << (cp-1))] < h[cp-1][j + (1 << (cp-1))]
             ;
}
void SA(){
        for(int i = 1 ; i <= n ; i ++ ) p[i] = i , h[0][i] = s[i];
        for(cp = 1 ; cp < lg ; cp++ ){
                sort(p +1 , p + n + 1 , cmp) ;
                h[cp][p[1]] =1 ;
                for(int i = 2 ; i <= n ; i ++ ) h[cp][p[i]] = h[cp][p[
                    i-1]] + cmp(p[i-1] , p[i]) ;
        }
}
void Lcp(){
        for(int i = 1 , k = 0 ; i <= n ; i ++ ){
                if(sa[i] == n) continue ;
                for(int j = p[sa[i] + 1] ; k <= n && s[i+k] == s[j+k]
                    ; ) k++ ;
                lcp[sa[i]] = k ;
                if(k) k -- ;
        }
}
```

9 Graph

9.1 Dijkastra

```
void dijkstra(ll source, ll n)
{
    // graph --> v[source].pb({destination, edge})
    // dist[i] --> distance(sources to i)
    ll dist[maxn];
    fill(dist, dist + maxn, inf);
    set<pair<ll, pii>> st;
    dist[source] = 0;
    st.insert({dist[source], {source, source}});
    while (st.size())
    {
        pair<ll, pii> now = (*st.begin());
        st.erase(now);
        dist[now.S.S] = min(dist[now.S.S], dist[now.S.F] + now.F);
        for (auto i : v[now.S.S])
            if(dist[now.S.S] + i.S < dist[i.F])
                st.insert({i.S, {now.S.S, i.F}});
    }
}
```

9.2 Scc

```
void sfd(int v , int c)
{
        comp[v] = c ;
        for(auto u : radj[v])
        {
                if(comp[u] == 0)
                {
                        sfd(u , c) ;
                }
        }
}

void dfs(int v)
{
        mark[v] = 1 ;
        for(auto u : adj[v])
        {
                if(mark[u.X] == 0 )
                {
                        dfs(u.X) ;
                }
        }
        t.pb(v) ;
}

reverse(t.begin() , t.end()) ;
for(auto v : t)
        {
                if(comp[v] == 0 ) sfd(v , cnt++) ;
        }
```

9.3 2sat

```
//harki mesle i khodesh 2 * i notesh 2 * i + 1 he
//yale u -> v yani agar u yek baseh v ham bayad yek bashe
vector<int> out[maxn * 2] , in[maxn * 2] , topol , adj[maxn];

bool visited[maxn * 2];
int color[maxn * 2] , c , val[maxn] , r[maxn] ;

void add_edge(int v , int u)
{
        out[v].pb(u);
        in[u].pb(v);
}

void dfs(int v)
{
        visited[v] = 1;
        for(auto u : out[v])
                if(!visited[u])
                        dfs(u);
        topol.pb(v);
}

void sfd(int v)
{
        visited[v] = 1;
        color[v] = c;
        for(auto u : in[v])
                if(!visited[u])
                        sfd(u);
}
for(int i = 0; i < 2 * n; i++)
                if(!visited[i])
                        dfs(i);
reverse(topol.begin() , topol.end());
memset(visited , 0 , sizeof visited);
for(int v = 1 ; v <= 2 * m + 1 ; v ++ )
                if(!visited[v])
                        sfd(v) , c++;
        for(int i = 1; i <= m; i++)
                if(color[2 * i] == color[2 * i + 1])
                        return cout << "NO" << endl , 0;
```

9.4 Centroid

```
void plant(int v , int p = -1)
{
        sz[v] = 1;
        for(auto u : adj[v])
        if(u != p && !hide[u])
        {
                plant(u , v);
                sz[v] += sz[u];
        }
}//
int find_centroid(int v , int n , int p = -1)
{
    bool found = 1;
        while(found)
    {
        found = 0;
        for(auto u : adj[v])
                if(u != p && !hide[u] && sz[u] * 2 > n)
            {
                found = 1;
                p = v;
```

```
                v = u;
                break;
            }
    }
        return v;
}
void solve(int v , int h = 0)
{
        plant(v);
        v = find_centroid(v , sz[v]);
        cout<<v <<"  level " << h << endl ;
        hide[v] = 1;
        for(auto u : adj[cent])
        if(!hide[u])
                solve(u , h + 1);
}
```

9.5 Dinitz

```
Dinic:
int from[MAXE], to[MAXE], cap[MAXE], prv[MAXE], head[MAXN], pt[MAXN],
    ec;
void addEdge(int u, int v, int uv, int vu = 0){
        from[ec] = u, to[ec] = v, cap[ec] = uv, prv[ec] = head[u],
            head[u] = ec++;
        from[ec] = v, to[ec] = u, cap[ec] = vu, prv[ec] = head[v],
            head[v] = ec++;
}
int lv[MAXN], q[MAXN];
bool bfs(int source, int sink){
        memset(lv, 63, sizeof(lv));
        int h = 0, t = 0;
        lv[source] = 0;
        q[t++] = source;
        while (t-h){
                int v = q[h++];
                for (int e = head[v]; ~e; e = prv[e])
                        if (cap[e] && lv[v] + 1 < lv[to[e]]){
                                lv[to[e]] = lv[v] + 1;
                                q[t++] = to[e];
                        }
        }
        return lv[sink] < 1e8;
}
int dfs(int v, int sink, int f = 1e9){
        if (v == sink || f == 0)
                return f;
        int ret = 0;
        for (int &e = pt[v]; ~e; e = prv[e])
                if (lv[v]+1 == lv[to[e]]){
                        int x = dfs(to[e], sink, min(f, cap[e]));
                        cap[e] -= x;
                        cap[e^1] += x;
                        ret += x;
                        f -= x;
                        if (!f)
                                break;
                }
        return ret;
}
int dinic(int source, int sink){
        int ret = 0;
        while (bfs(source, sink)){
                memcpy(pt, head, sizeof(head));
                ret += dfs(source, sink);
        }
        return ret;
}

        memset(head, -1, sizeof(head));
//mohem
```

9.6 Eularian Tour

```
void tour(int v , int b)
{
        if(b == 0) M[v] = 1 ;
        while( adj[v][b] .size())
        {
                auto k = adj[v][b].back() ; int u = k.X , id = k.Y ;
                adj[v][b].pop_back() ;
                if(mark[id] == 0)
                {
                        mark[id]  = 1 ;
                        tour(u , 1 -  b) ;
                }
        }
        ans.pb(v) ;
        a.pb({v , b}) ;
}
```

9.7 Min Cost Max Flow

```
template<typename F, typename C, int MAXN, int MAXM>
struct MinCostMaxFlow {
        struct Edge {
                int from, to;
                F cap;
                C cost;
        };


        Edge E[2 * MAXM];
        int m, par[MAXN], s, t;
        C dist[MAXN], cost = 0;
        F mn[MAXN], flow = 0;
        vector<int> adj[MAXN];

        inline void add_edge(int u, int v, F cap, C cost) {
                adj[u].push_back(m);
                E[m++] = {u, v, cap, cost};
                adj[v].push_back(m);
                E[m++] = {v, u, 0, -cost};
        }

        inline void SPFA() {
                fill(dist, dist + MAXN, numeric_limits<C>::max());
                fill(par, par + MAXN, -1);
                queue<int> q;
```

```
                dist[s] = 0;
                q.push(s);
                mn[s] = numeric_limits<F>::max();

                while (!q.empty()) {
                        int v = q.front();
                        q.pop();

                        for (int id : adj[v]) {
                                int u = E[id].to;
                                if (!E[id].cap) continue;

                                if (dist[u] > dist[v] + E[id].cost) {
                                        dist[u] = dist[v] + E[id].cost
                                                ;
                                        q.push(u);
                                        par[u] = id;
                                        mn[u] = min(mn[v], E[id].cap);
                                }
                        }
                }
        }

        inline F solve() {
                SPFA();

                if (par[t] == -1) return 0;
                F c = mn[t], v = t;
                flow += c;
                cost += c * dist[t];

                while (v != s) {
                        int id = par[v];
                        E[id].cap -= c;
                        E[id ^ 1].cap += c;
                        v = E[id].from;
                }

                return c;
        }

        inline pair<F, C> max_flow(int _s, int _t) {
                s = _s, t = _t;
                while (true) {
                        F c = solve();
                        if (!c) break;
                }

                return {flow, cost};
        }
};

const ll MAXN = 402;
MinCostMaxFlow<int, double, MAXN * 2, MAXN * MAXN> flow;
```

10 Geometry

10.1 Geometry 1

```
#include <bits/stdc++.h>
#include <bits/extc++.h>
using namespace std;

#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define trav(a, x) for(auto& a : x)
#define all(x) x.begin(), x.end()
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef vector<int> vi;
typedef vector<ll> vl;
typedef long double ld;

const ll nils = 1000000007;

template <class T> int sgn(T x) { return (x > 0) - (x < 0); }
template<class T>
struct Point {
  typedef Point P;
  T x, y;
  explicit Point(T x=0, T y=0) : x(x), y(y) {}
  bool operator<(P p) const { return tie(x,y) < tie(p.x,p.y); }
  bool operator==(P p) const { return tie(x,y)==tie(p.x,p.y); }
  P operator+(P p) const { return P(x+p.x, y+p.y); }
  P operator-(P p) const { return P(x-p.x, y-p.y); }
  P operator*(T d) const { return P(x*d, y*d); }
  P operator/(T d) const { return P(x/d, y/d); }
  T dot(P p) const { return x*p.x + y*p.y; }
  T cross(P p) const { return x*p.y - y*p.x; }
  T cross(P a, P b) const { return (a-*this).cross(b-*this); }
  T dist2() const { return x*x + y*y; }
  double dist() const { return sqrt((double)dist2()); }
  // angle to x-axis in interval [-pi, pi]
  double angle() const { return atan2(y, x); }
  P unit() const { return *this/dist(); } // makes dist()=1
  P perp() const { return P(-y, x); } // rotates +90 degrees
  P normal() const { return perp().unit(); }
  // returns point rotated 'a' radians ccw around the origin
  P rotate(double a) const {
    return P(x*cos(a)-y*sin(a),x*sin(a)+y*cos(a)); }
  friend ostream& operator<<(ostream& os, P p) {
    return os << "(" << p.x << "," << p.y << ")"; }
};

struct Angle {
  int x, y;
  int t;
  Angle(int x, int y, int t=0) : x(x), y(y), t(t) {}
  Angle operator-(Angle b) const { return {x-b.x, y-b.y, t}; }
  int half() const {
    assert(x || y);
    return y < 0 || (y == 0 && x < 0);
  }
  Angle t90() const { return {-y, x, t + (half() && x >= 0)}; }
  Angle t180() const { return {-x, -y, t + half()}; }
  Angle t360() const { return {x, y, t + 1}; }
};
bool operator<(Angle a, Angle b) {
  // add a.dist2() and b.dist2() to also compare distances
```

```
  return make_tuple(a.t, a.half(), a.y * (ll)b.x) <
         make_tuple(b.t, b.half(), a.x * (ll)b.y);
}

template<class P> bool onSegment(P s, P e, P p) {
  return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
}

template<class P> vector<P> segInter(P a, P b, P c, P d) {
  auto oa = c.cross(d, a), ob = c.cross(d, b),
       oc = a.cross(b, c), od = a.cross(b, d);
  // Checks if intersection is single non-endpoint point.
  if (sgn(oa) * sgn(ob) < 0 && sgn(oc) * sgn(od) < 0)
    return {(a * ob - b * oa) / (ob - oa)};
  set<P> s;
  if (onSegment(c, d, a)) s.insert(a);
  if (onSegment(c, d, b)) s.insert(b);
  if (onSegment(a, b, c)) s.insert(c);
  if (onSegment(a, b, d)) s.insert(d);
  return {all(s)};
}

template<class T>
T polygonArea2(vector<Point<T>>& v) {
  T a = v.back().cross(v[0]);
  rep(i,0,sz(v)-1) a += v[i].cross(v[i+1]);
  return a;
}

vi num, st;
vector<vector<pii>> ed;
int Time;
int comps = 0;
int bridges = 0;
int dfs(int at, int par) {
  int me = num[at] = ++Time, top = me;
  for (auto [y, e] : ed[at]) if (e != par) {
    if (num[y]) {
      top = min(top, num[y]);
      if (num[y] < me)
        st.push_back(e);
    } else {
      int si = sz(st);
      int up = dfs(y, e);
      top = min(top, up);
      if (up == me) {
        st.push_back(e);
                comps++;
        st.resize(si);
      }
      else if (up < me) st.push_back(e);
      else {bridges++;}
    }
  }
  return top;
}

void bicomps() {
  num.assign(sz(ed), 0);
  rep(i,0,sz(ed)) if (!num[i]) dfs(i, -1);
}
```

```
int n = 0;
int m = 0;
vector<vl> segments;
map<ll,int> M;
vector<Point<ll> > pts;
vector<vi> C;
vector<vi> taken;

int geti(ll x, ll y){
    ll h = x*nils + y;
    if(M.find(h) != M.end())return M[h];
    M[h] = n;
    pts.push_back(Point<ll>(x, y));
    n++;
    vi temp;
    C.push_back(temp);
    taken.push_back(temp);
    vector<pii> temp2;
    ed.push_back(temp2);
    return n-1;
}
int from;
bool comp(int i, int j){
    Point<ll> pi = pts[i]-pts[from];
    Point<ll> pj = pts[j]-pts[from];
    return Angle(pi.x, pi.y) < Angle(pj.x, pj.y);
}

map<ll,ll> I;

int index(ll i, ll to){
    ll h = i*nils+to;
    assert(I.find(h) != I.end());
    return I[h];
}

vl areas;

ll get_area(int i, int j){
    vector<Point<ll> > polygon;
    int p = i;
    int q = j;
    while(1){
        polygon.push_back(pts[i]);
        taken[i][j] = 1;
        int i2 = C[i][j];
        int j2 = index(i,i2);
        j2 = (j2+1)%sz(C[i2]);
        i = i2;
        j = j2;
        if(i == p && j == q)break;
    }

    return polygonArea2(polygon);
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
```

```
cin >> m;

rep(c1,0,m){
    ll x1,y1,x2,y2;
    cin >> x1 >> y1 >> x2 >> y2;
    segments.push_back({x1,y1,x2,y2});
    int i = geti(x1,y1);
    int j = geti(x2,y2);
    assert(i != j);
    C[i].push_back(j);
    C[j].push_back(i);
    ed[i].push_back({j, 2*c1});
    ed[j].push_back({i, 2*c1+1});
    taken[i].push_back(0);
    taken[j].push_back(0);
}

bool planar = 1;

rep(c1,0,min(m,1000)){
    rep(c2,c1+1,min(m,1000)){
        Point<ll> p1 = Point<ll>(segments[c1][0], segments[c1][1])
            ;
        Point<ll> q1 = Point<ll>(segments[c1][2], segments[c1][3])
            ;
        Point<ll> p2 = Point<ll>(segments[c2][0], segments[c2][1])
            ;
        Point<ll> q2 = Point<ll>(segments[c2][2], segments[c2][3])
            ;
        vector<Point<ll>> inter = segInter(p1,q1,p2,q2);
        if(sz(inter) == 0)continue;
        if(sz(inter) > 1){
            planar = 0;
            continue;
        }
        if(!(inter[0] == p1 || inter[0] == q1)){
            planar = 0;
        }
    }
}
assert(planar);
bicomps();
//assert(comps == 1);
assert(bridges == 0);

rep(c1,0,n){
    from = c1;
    sort(all(C[c1]), comp);

    rep(c2,0,sz(C[c1])){
        ll from2 = C[c1][c2];
        ll h = from2*nils+ll(c1);
        I[h] = c2;
    }
}

rep(c1,0,n){
    rep(c2,0,sz(C[c1])){
        if(!taken[c1][c2]){
            areas.push_back(abs(get_area(c1, c2)));
        }
    }
}
sort(all(areas));
double ans = 0.0;

rep(c1,0,sz(areas)-1){
    ans += areas[c1]*areas[c1];
}

cout << setprecision(18) << fixed << ans / 4.0 << "\n";

return 0;
}
```

10.2 Geometry 2

```
double area(const vector<point>& fig) {
    double res = 0;
    for (unsigned i = 0; i < fig.size(); i++) {
        point p = i ? fig[i - 1] : fig.back();
        point q = fig[i];
        res += (p.x - q.x) * (p.y + q.y);
    }
    return fabs(res) / 2;
}

struct pt {
    long long x, y;
    pt() {}
    pt(long long _x, long long _y) : x(_x), y(_y) {}
    pt operator+(const pt &p) const { return pt(x + p.x, y + p.y); }
    pt operator-(const pt &p) const { return pt(x - p.x, y - p.y); }
    long long cross(const pt &p) const { return x * p.y - y * p.x; }
    long long dot(const pt &p) const { return x * p.x + y * p.y; }
    long long cross(const pt &a, const pt &b) const { return (a - *
        this).cross(b - *this); }
    long long dot(const pt &a, const pt &b) const { return (a - *this)
        .dot(b - *this); }
    long long sqrLen() const { return this->dot(*this); }
};

bool pointInTriangle(pt a, pt b, pt c, pt point) {
    long long s1 = abs(a.cross(b, c));
    long long s2 = abs(point.cross(a, b)) + abs(point.cross(b, c)) +
        abs(point.cross(c, a));
    return s1 == s2;
}

Area of triangle with sides a, b, c: sqrt(S *(S-a)*(S-b)*(S-c)) where
    S = (a+b+c)/2
Area of equilateral triangle: s^2 * sqrt(3) / 4 where is side lenght
Pyramid and cones volume: 1/3 area(base) * height

if p1=(x1, x2), p2=(x2, y2), p3=(x3, y3) are points on circle, the
    center is
x = -((x2^2 - x1^2 + y2^2 - y1^2)*(y3 - y2) - (x2^2 - x3^2 + y2^2 - y3
    ^2)*(y1 - y2)) / (2*(x1 - x2)*(y3 - y2) - 2*(x3 - x2)*(y1 - y2))
y = -((y2^2 - y1^2 + x2^2 - x1^2)*(x3 - x2) - (y2^2 - y3^3 + x2^2 - x3
    ^2)*(x1 - x2)) / (2*(y1 - y2)*(x3 - x2) - 2*(y3 - y2)*(x1 - x2))
```

10.3 Convex hull

10.4 Convex hull trick

```
#include <bits/stdc++.h>
using namespace std;

struct Line {
    long long m, c;
    Line(long long _m = 0ll, long long _c = 0ll) {m = (long long)_m; c
        = (long long)_c;}
    long long find_y (int x) { return (m * (long long)(x)) + c;}
    long double intersect_x (Line l) {return (long double)(l.c - c) /
        (long double)(m - l.m); }
};
deque < Line > dq;

const int MAXN = 1e6 + 12;
int n, p[MAXN], q[MAXN];
long long dp[MAXN], ans, a[MAXN];
vector < int > points;

int32_t main() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> p[i] >> q[i] >> a[i];
        points.push_back(i);
    }
    sort(points.begin(), points.end(), cmp);

    dq.push_back(Line(0, 0));
    for (int i = 0; i < n; i++) {
        int ind = points[i];
        while(dq.size() >= 2
            && dq.back().find_y(q[ind]) <= dq[dq.size() - 2].find_y(q[
                ind]))
            dq.pop_back();

        dp[i] = ((long long)(p[ind]) * (long long)(q[ind])) - (long
            long)a[ind] + dq.back().find_y(q[ind]);
        Line new_line(-1ll * p[ind], (long long)(dp[i]));

        while (dq.size() >= 2
            && dq[0].intersect_x(new_line) >= dq[0].intersect_x(dq[1])
                )
            dq.pop_front();
        dq.push_front(new_line);
        ans = max(ans, dp[i]);
    }

    cout << ans;
    return 0;
}

// Salam
struct pt {
    double x, y;
    bool typ;
    bool operator == (pt const& t) const {
        return x == t.x && y == t.y;
    }
};

int orientation(pt a, pt b, pt c) {
    double v = a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y);
    if (v < 0) return -1; // clockwise
    if (v > 0) return +1; // counter-clockwise
    return 0;
}

bool cw(pt a, pt b, pt c, bool include_collinear) {
    int o = orientation(a, b, c);
    return o < 0 || (include_collinear && o == 0);
}
bool collinear(pt a, pt b, pt c) { return orientation(a, b, c) == 0; }

void convex_hull(vector<pt>& a, bool include_collinear = false) {
    pt p0 = *min_element(a.begin(), a.end(), [](pt a, pt b) {
        return make_pair(a.y, a.x) < make_pair(b.y, b.x);
    });
    sort(a.begin(), a.end(), [&p0](const pt& a, const pt& b) {
        int o = orientation(p0, a, b);
        if (o == 0)
            return (p0.x-a.x)*(p0.x-a.x) + (p0.y-a.y)*(p0.y-a.y)
                < (p0.x-b.x)*(p0.x-b.x) + (p0.y-b.y)*(p0.y-b.y);
        return o < 0;
    });
    if (include_collinear) {
        int i = (int)a.size()-1;
        while (i >= 0 && collinear(p0, a[i], a.back())) i--;
        reverse(a.begin()+i+1, a.end());
    }

    vector<pt> st;
    for (int i = 0; i < (int)a.size(); i++) {
        while (st.size() > 1 && !cw(st[st.size()-2], st.back(), a[i],
            include_collinear))
            st.pop_back();
        st.push_back(a[i]);
    }

    if (include_collinear == false && st.size() == 2 && st[0] == st
        [1])
        st.pop_back();

    a = st;
}
```

10.5 Circle

```
#define _USE_MATH_DEFINES
const int MAX_N = 2e5+10;
const int INF = 1e9;
const ld eps = 1e-8;
struct circle {
public:
  ld r;
  point o;
  circle(ld rr, ld x, ld y) {
    r = rr;
    o.x = x;
    o.y = y;
```

```
}
ld S() {
  return M_PI*r*r;
}
ld distance(point p1, point p2) { return hypot(p2.x-p1.x,p2.y-p1.y);
    }
/*
  0 = other is inside this, zero point
  1 = other is tangent inisde of this, one point
  2 = other is intersect with this, two point
  3 = other is tangent outside of this, one point
  4 = other is outside of this, zero point
*/
pair<int, vector<point> > intersect(circle other) {
  vector<point> v;
  ld sumr = other.r + r;
  ld rr = r - other.r;
  ld dis = distance(o, other.o);
  ld a = (r*r - other.r*other.r + dis*dis)/(2*dis);
  ld h = sqrt(r*r-a*a);
  point p2(o.x, o.y);
  p2.x = a*(other.o.x - o.x)/dis;
  p2.y = a*(other.o.y - o.y)/dis;
  if(is_zero(sumr-dis)) {
    v.push_back(p2);
    return make_pair(3, v);
  }
  if(is_zero(rr - dis)) {
    v.push_back(p2);
    return make_pair(1, v);
  }
  if(dis <= rr)
    return make_pair(0, v);
  if(dis >= sumr)
    return make_pair(4, v);
  point p3(p2.x + h*(other.o.y - o.y)/dis, p2.y - h*(other.o.x - o.x
      )/dis);
  point p4(p2.x - h*(other.o.y - o.y)/dis, p2.y + h*(other.o.x - o.x
      )/dis);
  v.push_back(p3);
  v.push_back(p4);
  return make_pair(2, v);
}
ld f(ld l, ld r, ld R){
  ld cosa = (l*l + r*r - R*R)/(2.0*r*l);
  ld a = acos(cosa);
  return r*r*(a - sin(2*a)/2);
}
ld intersection_area(circle c2) {
  ld l = distance(o, c2.o);
  if(l >= r + c2.r) return 0;
  else if(c2.r >= l + r) return S();
  else if(r >= l + c2.r) return c2.S();
  return f(l, r, c2.r) + f(l, c2.r, r);
  }
};
```

10.6 Point in Circle

```
// returns positive if d is outside circle ABC, positive if d is
    inside it and 0 if it's on border
int inCircle (point a, point b, point c, point d){
        if (cross(b - a, c - a) < 0)
                swap(b, c);
        int x[4][4] = {
                1, a.first, a.second, a.first * a.first + a.second * a
                    .second,
                1, b.first, b.second, b.first * b.first + b.second * b
                    .second,
                1, c.first, c.second, c.first * c.first + c.second * c
                    .second,
                1, d.first, d.second, d.first * d.first + d.second * d
                    .second
        };
        // you can replace the following with any faster way
        // of calculating determinant.
        int y[] = {0, 1, 2, 3};
        int ans = 0;
        do {
                int mul = 1;
                for (int i = 0; i < 4; i++)
                        for (int j = i + 1; j < 4; j++)
                                if (y[i] > y[j])
                                        mul *= -1;
                for (int i = 0; i < 4; i++)
                        mul *= x[i][y[i]];
                ans += mul;
        } while (next_permutation(y, y + 4));
        return ans;
}
```

10.7 Circle Line

```
typedef pair<point,point> ppp;
const ld INF = 1e18;
const ld eps = 1e-15;
ppp line_circle_intersection(point p1,point p2,point o,ld r){
  point q = dot(o-p1,p2-p1)/dist(p1,p2)*(p2-p1) + p1;
  ld d = r*r - dist(o,q);
  if(d<eps && d>-eps) return ppp(q, point(INF,INF));
  if(d<0) return ppp(point(INF,INF), point(INF,INF));
  point dif = sqrt(d/dist(p1,p2))*(p1-p2);
  return ppp(q-dif,q+dif);
}
```