Heltion's CP-Teamplate

Heltion

November 7, 2019

Catalog

1 convolution

1.1 FFT in C

LibreOJ108: maxn = 100000, time = 117ms, memory = 11MB.

```
using R = double;
constexpr R PI = acos(-(R)1);
constexpr int maxn = 1 << 21;
struct C{
    R a, b;
    C operator + (const C& B)const{return {a + B.a, b + B.b};}
    C operator - (const C& B)const{return {a - B.a, b - B.b};}
    C operator * (const C& B)const{return {a * B.a - b * B.b, a * B.b + b * B.a};}
    C operator / (const int& B)const{return {a / B, b / B};}
    C conj()const{return {a, -b};}
};
void DFT(C* a, int n, bool idft = false){
    static int r[maxn], on;
    static C w[maxn];
    if(on != n){
        for(int i = 0; i < n; i += 1){
            r[i] = i ? r[i >> 1] >> 1 | (i & 1 ? n >> 1 : 0) : 0;
            w[i] = {cos(PI * i / n), sin(PI * i / n)};
        }
        on = n;
    }
    for(int i = 0; i < n; i += 1) if(i < r[i]) swap(a[i], a[r[i]]);
    for(int m = 1; m <= n >> 1; m <<= 1)
        for(int j = 0; j < n; j += m << 1)
            for(int k = 0, p = 0; k < m; k += 1, p += n / m){
                C tmp = (idft ? w[p].conj() : w[p]) * a[j + k + m];
                a[j + k + m] = a[j + k] - tmp;
                a[j + k] = a[j + k] + tmp;
            }
    if(idft) for(int i = 0; i < n; i += 1) a[i] = a[i] / n;
}
int a[maxn], b[maxn], c[maxn];
C p[maxn];
int mul(int* a, int n, int* b, int m, int* c){
    int k = 1;
    while(k < n + m - 1) k <<= 1;
    for(int i = 0; i < k; i += 1) p[i] = {i < n ? a[i] : 0., i < m ? b[i] : 0.};
    DFT(p, k);
    for(int i = 0; i < k; i += 1) p[i] = p[i] * p[i];
    DFT(p, k, true);
    for(int i = 0; i < n + m - 1; i += 1) c[i] = p[i].b * .5 + .5;
    return n + m - 1;
}
```

1.2 FFT in Fp

LibreOJ108: maxn = 100000, time = 70ms, memory = 12MB.
Luogu3803: maxn = 1000000, time = 857ms, memory = 89MB.

```
using LL = long long;
constexpr LL mod = 998244353, G = 3;
constexpr int maxn = 1 << 21;
LL add(LL a, const LL& b){a += b; return a >= mod ? a - mod : a;}
LL sub(LL a, const LL& b){a -= b; return a < 0 ? a + mod : a;}
LL mul(const LL& a, const LL& b){return a * b % mod;}
LL power(LL a, LL r){
    LL res = 1;
```

```
    for(; r; r >>= 1, a = mul(a, a)) if(r & 1) res = mul(res, a);
    return res;
}
void DFT(LL* a, int n, bool idft = false){
    static int r[maxn], on;
    static LL w[maxn], w_[maxn];
    if(on != n){
        r[1] = n >> 1; w[0] = w_[0] = 1; w[1] = power(G, (mod - 1) / n / 2); w_[1] =
            power(w[1], mod - 2);
        for(int i = 2; i < n; i += 1){
            r[i] = r[i >> 1] >> 1 | (i & 1 ? n >> 1 : 0);
            w[i] = mul(w[i - 1], w[1]);
            w_[i] = mul(w_[i - 1], w_[1]);
        }
        on = n;
    }
    for(int i = 0; i < n; i += 1) if(i < r[i]) swap(a[i], a[r[i]]);
    for(int m = 1; m <= n >> 1; m <<= 1)
        for(int j = 0; j < n; j += m << 1)
            for(int k = 0, p = 0; k < m; k += 1, p += n / m){
                LL tmp = mul(idft ? w_[p] : w[p], a[j + k + m]);
                a[j + k + m] = sub(a[j + k], tmp);
                a[j + k] = add(a[j + k], tmp);
            }
    if(idft) for(int i = 0, _ = power(n, mod - 2); i < n; i += 1) a[i] = mul(a[i], _);
}
LL A[maxn], B[maxn], C[maxn];
int mul(LL* A, int n, LL* B, int m, LL* C){
    int k = 1;
    while(k < n + m - 1) k <<= 1;
    DFT(A, k);
    DFT(B, k);
    for(int i = 0; i < k; i += 1) C[i] = mul(A[i], B[i]);
    DFT(C, k, true);
    return n + m - 1;
}
```

2 data structure

2.1 01Trie

Luogu3369: $n = 100000$, time $= 67$ms, memory $= 5$MB.

```

constexpr int maxn = 120000;
constexpr int maxb = 32;
struct _01Trie{
    int count[maxn * maxb], ch[maxn * maxb][2], size = 1;
    int in[maxb], out[maxb];
    void encode(unsigned x){
        x ^= 1U << 31;
        for(int i = 0; i < maxb; i += 1) in[i] = x >> (maxb - 1 - i) & 1;
    }
    int decode(){
        unsigned res = 0;
        for(int i = 0; i < maxb; i += 1) res = res << 1 | out[i];
        return res ^ 1U << 31;
    }
    void insert(int x){
        encode(x);
        int p = 1;
```

```
        for(int c : in){
            if(not ch[p][c]) ch[p][c] = size += 1;
            count[p = ch[p][c]] += 1;
        }
    }
    void erase(int x){
        encode(x);
        int p = 1;
        for(int c : in) if(not count[p = ch[p][c]]) return;
        p = 1;
        for(int c : in) count[p = ch[p][c]] -= 1;
    }
    int rank(int x){
        encode(x);
        int p = 1, res = 1;
        for(int c : in) res += c ? count[ch[p][0]] : 0, p = ch[p][c];
        return res;
    }
    int k_th(int k){
        int p = 1;
        for(int &c : out)
            if(k > count[ch[p][0]]) k -= count[ch[p][0]], p = ch[p][c = 1];
            else p = ch[p][c = 0];
        return decode();
    }
    int prev(int x){
        encode(x);
        int p = 1, q = 0;
        for(int c : in){
            if(c and count[ch[p][0]]) q = p;
            p = ch[p][c];
        }
        if(not q) return -INT_MAX;
        copy(in, in + maxb, out);
        p = 1;
        for(int& c : out)
            if(q and p != q) p = ch[p][c];
            else if(p == q) q = 0, p = ch[p][c = 0];
            else p = ch[p][c = not not count[ch[p][1]]];
        return decode();
    }
    int next(int x){
        encode(x);
        int p = 1, q = 0;
        for(int c : in){
            if(not c and count[ch[p][1]]) q = p;
            p = ch[p][c];
        }
        if(not q) return INT_MAX;
        copy(in, in + maxb, out);
        p = 1;
        for(int& c : out)
            if(q and p != q) p = ch[p][c];
            else if(p == q) q = 0, p = ch[p][c = 1];
            else p = ch[p][c = not count[ch[p][0]]];
        return decode();
    }

};
```

2.2 persisting 01Trie

Luogu3835: $n = 500000$, time $= 599$ms, memory $= 176$MB.

```
constexpr int maxn = 540000;
constexpr int maxb = 32;
struct Persisting_01Trie{
    int root[maxn], count[maxn * maxb], ch[maxn * maxb][2], size;
    int in[maxb], out[maxb];
    void encode(unsigned x){
        x ^= 1U << 31;
        for(int i = 0; i < maxb; i += 1) in[i] = x >> (maxb - 1 - i) & 1;
    }
    int decode(){
        unsigned res = 0;
        for(int i = 0; i < maxb; i += 1) res = res << 1 | out[i];
        return res ^ 1U << 31;
    }
    void insert(int u, int v, int x){
        encode(x);
        int pu = root[u], pv = root[v] = size += 1;
        for(int c : in){
            ch[pv][c ^ 1] = ch[pu][c ^ 1];
            pu = ch[pu][c];
            pv = ch[pv][c] = size += 1;
            count[pv] = count[pu] + 1;
        }
    }
    void erase(int u, int v, int x){
        encode(x);
        int pu = root[u], pv = root[v] = pu;
        for(int c : in) if(not count[pu = ch[pu][c]]) return;
        pu = root[u], pv = root[v] = size += 1;
        for(int c : in){
            ch[pv][c ^ 1] = ch[pu][c ^ 1];
            pu = ch[pu][c];
            pv = ch[pv][c] = size += 1;
            count[pv] = count[pu] - 1;
        }
    }
    int rank(int u, int v, int x){
        encode(x);
        int pu = root[v] = root[u], res = 1;
        for(int c : in) res += c ? count[ch[pu][0]] : 0, pu = ch[pu][c];
        return res;
    }
    int k_th(int u, int v, int k){
        int pu = root[v] = root[u];
        for(int &c : out)
            if(k > count[ch[pu][0]]) k -= count[ch[pu][0]], pu = ch[pu][c = 1];
            else pu = ch[pu][c = 0];
        return decode();
    }
    int prev(int u, int v, int x){
        encode(x);
        int pu = root[v] = root[u], pv = 0;
        for(int c : in){
            if(c and count[ch[pu][0]]) pv = pu;
            pu = ch[pu][c];
        }
        if(not pv) return -INT_MAX;
        copy(in, in + maxb, out);
```

```
        pu = root[u];
        for(int& c : out)
            if(pv and pu != pv) pu = ch[pu][c];
            else if(pu == pv) pv = 0, pu = ch[pu][c = 0];
            else pu = ch[pu][c = not not count[ch[pu][1]]];
        return decode();
    }
    int next(int u, int v, int x){
        encode(x);
        int pu = root[v] = root[u], pv = 0;
        for(int c : in){
            if(not c and count[ch[pu][1]]) pv = pu;
            pu = ch[pu][c];
        }
        if(not pv) return INT_MAX;
        copy(in, in + maxb, out);
        pu = root[u];
        for(int& c : out)
            if(pv and pu != pv) pu = ch[pu][c];
            else if(pu == pv) pv = 0, pu = ch[pu][c = 1];
            else pu = ch[pu][c = not count[ch[pu][0]]];
        return decode();
    }

};
```

3 geometry

3.1 convex hull

```
using R = double;
constexpr int maxn = 100000;
struct P{
    R x, y;
    P operator - (const P& B){return {x - B.x, y - B.y};}
}p[maxn], h[maxn + 1];
R cross(const P& A, const P& B){return A.x * B.y - A.y * B.x;}
int convex_hull(P *p, P* h, int n){
    sort(p, p + n, [](const P& A, const P& B){
        return A.x == B.x ? A.y < B.y : A.x < B.x;
    });
    int hn = 0;
    for(int i = 0; i < n; i += 1){
        while(hn >= 2 and cross(p[i] - h[hn - 2], h[hn - 1] - h[hn - 2]) >= 0) hn -= 1;
        h[hn ++] = p[i];
    }
    int tmp = hn;
    for(int i = n - 1; ~i; i -= 1){
        while(hn > tmp and cross(p[i] - h[hn - 2], h[hn - 1] - h[hn - 2]) >= 0) hn -=
            1;
        h[hn ++] = p[i];
    }
    return hn - 1;
}
```

4 linear algebra

4.1 matrix inverse in Fp

```cpp
using LL = long long;
constexpr int maxn = 400;
constexpr LL mod = 1000000007;
LL A[maxn][maxn], B[maxn][maxn];
LL power(LL a, LL r){
    LL res = 1;
    for(; r; r >>= 1, a = a * a % mod)
        if(r & 1) res = res * a % mod;
    return res;
}
void sub(LL &a, LL b){
    a -= b;
    if(a < 0) a += mod;
}
bool matrix_inverse(int n, LL A[][maxn], LL B[][maxn]){

    for(int i = 0; i < n; i += 1)
        for(int j = 0; j < n; j += 1) B[i][j] = i == j;
    for(int i = 0; i < n; i += 1){
        int k = -1;
        for(int j = i; j < n and not~k; j += 1) if(A[j][i]) k = j;
        if(not~k) return false;
        if(k != i) {
            for(int j = i; j < n; j += 1) swap(A[i][j], A[k][j]);
            for(int j = 0; j <= i; j += 1) swap(B[i][j], B[k][j]);
        }
        LL inv = power(A[i][i], mod - 2);
        for(int j = i; j < n; j += 1) A[i][j] = A[i][j] * inv % mod;
        for(int j = 0; j <= i; j += 1) B[i][j] = B[i][j] * inv % mod;
        for(int j = 0; j < n; j += 1) if(j != i){
            LL tmp = A[j][i];

            for(int k = i; k < n; k += 1) sub(A[j][k], tmp * A[i][k] % mod);
            for(int k = 0; k <= i; k += 1) sub(B[j][k], tmp * B[i][k] % mod);
        }
    }
    return true;
}
```

5 miscellaneous

5.1 check

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(int argc, char** argv){
    if(argc > 1) srand(atoi(argv[1]));
    //...
}
```

```
make $1
make $2
make $3
i=0
while true
do
    ./$3 $i > $1.in
    ./$1 < $1.in > $1.out
    ./$2 < $1.in > $2.out
    if(diff $1.out $2.out)
```

```
        then
            echo "OK at "$i
        else
            echo "WA at "$i
            break
    fi
    let "i++"
done
```

5.2 makefile

```
%:%.cpp
    g++ $< -o $@ -O2 -std=c++14
```

6 number theory

6.1 Cipolla

```
using LL = long long;
LL power(LL a, LL r, const LL& p){
    LL res = 1;
    for(; r; r >>= 1, a = a * a % p)
        if(r & 1) res = res * a % p;
    return res;
}
struct Fp2{
    LL x, y, w, p;
    Fp2 operator * (const Fp2& B)const{
        assert(w == B.w and p == B.p);
        Fp2 res = {0, 0, w, p};
        res.x = (x * B.x + y * B.y % p * w) % p;
        res.y = (x * B.y + y * B.x) % p;
        return res;
    }
};
LL sqrt(LL n, LL p){
    if(n == 0) return 0;
    if(power(n, (p ^ 1) >> 1, p) != 1) return -1;
    LL x, w;
    do x = rand(), w = (x * x + p - n) % p;
    while(power(w, (p ^ 1) >> 1, p) == 1);
    Fp2 a = {x, 1, w, p}, res = {1, 0, w, p};
    for(LL r = (p + 1) >> 1; r; r >>= 1, a = a * a)
        if(r & 1) res = res * a;
    return res.x;
}
```

6.2 Du's method

```
using LL = long long;
constexpr int maxp = 120000;
constexpr int maxn = 1 << 22;
constexpr LL mod = 1000000007;
int p[maxn], ps[maxp], pg[maxp], pn, m;
bool cp[maxn];
int phi[maxn];
LL v[maxp << 1], s[maxp << 1], sphi[maxn], on;
LL add(LL x, const LL& y){x += y; return x >= mod ? x - mod : x;}
```

```
LL sub(LL x, const LL& y){x -= y; return x < 0 ? x + mod : x;}
LL id(LL n){return n <= m ? ps[n] : pg[on / n];}
LL sfg(LL n){n %= mod; return n * (n + 1) / 2 % mod;}//...
LL sg(LL n){return n % mod;}//...
LL S(LL n){
    m = sqrt(on = n) + 1;
    phi[1] = sphi[1] = 1;
    for(int i = 2; i < maxn; i += 1){
        if(not cp[i]) p[pn ++] = i, phi[i] = i - 1;
        for(int j = 0; j < pn and i * p[j] < maxn; j += 1){
            cp[i * p[j]] = true;
            if(i % p[j] == 0){
                phi[i * p[j]] = phi[i] * p[j];
                break;
            }
            phi[i * p[j]] = phi[i] * (p[j] - 1);
        }
        sphi[i] = add(sphi[i - 1], phi[i]);
    }
    int gn = 0;
    for(LL L = 1, R; L <= n; L = R + 1){
        R = n / (n / L);
        if(n / L <= m) ps[n / L] = gn += 1;
        else pg[R] = gn += 1;
        v[gn] = n / L;
    }
    for(int i = gn; i; i -= 1)
        if(v[i] < maxn) s[i] = sphi[v[i]];
        else{
            s[i] = sfg(v[i]);
            for(LL L = 2, R; L <= v[i]; L = R + 1){
                R = v[i] / (v[i] / L);
                s[i] = sub(s[i], sub(sg(R), sg(L - 1)) * s[id(v[i] / L)] % mod);
            }
        }
    return s[1];
}
```

6.3 extended BSGS

```
LL gcd(LL a, LL b){
    return b ? gcd(b, a % b) : a;
}
LL exBSGS(LL a, LL b, LL p){
    LL res = 0, k = 1, d;
    if(b == 1) return 0;
    if(not a) return b ? -1 : 1;
    for(;k != b and (d = gcd(a, p)) != 1; res += 1){
        if(b % d) return -1;
        p /= d, b /= d;
        k = k * (a / d) % p;
    }
    if(k == b) return res;
    unordered_map<LL, LL> mp;
    LL x = 1, y, M = sqrt(p) + 1;
    for(int i = 0; i < M; i += 1, x = x * a % p) mp[b * x % p] = i;
    y = k * x % p;
    for(int i = 1; i <= M; i += 1, y = y * x % p)
        if(mp.count(y)) return res + i * M - mp[y];
    return -1;
}
```

6.4 extended CRT

```
LL exgcd(LL a, LL b, LL& x, LL& y){
    if(not b) return x = 1, y = 0, a;
    LL d = exgcd(b, a % b, x, y), t = x;
    return x = y, y = t - a / b * y, d;
}
bool excrt(int n, LL* m, LL* r, LL& M, LL& R){
    __int128 IM = 1, IR = 0;
    for(int i = 0; i < n; i += 1){
        LL x, y, d = exgcd(IM, m[i], x, y);
        if((r[i] - IR) % d) return false;
        IR += ((r[i] - IR) * x / d) % (m[i] / d) * IM;
        IM = IM / d * m[i];
        IR = (IR % IM + IM) % IM;
    }
    M = IM;
    R = IR;
    return true;
}
```

6.5 extended Lucas

```
using LL = long long;
struct PLL{LL x, y;};
LL power(LL a, LL r, LL p){
    LL res = 1;
    for(; r; r >>= 1, a = a * a % p) if(r & 1) res = res * a % p;
    return res;
}
LL inverse(LL a, PLL p){
    return power(a, p.y / p.x * (p.x - 1) - 1, p.y);
}
PLL fact(LL n, PLL p){
    if(n == 0) return {0, 1};
    PLL res = fact(n / p.x, p);
    res.x += n / p.x;
    for(LL i = 1; i <= n % p.y; i += 1) if(i % p.x) res.y = res.y * i % p.y;
    if((p.x == 4 or p.y != 2) and ((n / p.y) & 1)) res.y = p.y - res.y;
    return res;
}
LL exLucas(LL m, LL n, PLL p){
    PLL fm = fact(m, p), fn = fact(n, p), fm_n = fact(m - n, p);
    LL res = fm.y * inverse(fn.y, p) % p.y * inverse(fm_n.y, p) % p.y;
    for(int i = 0; i < fm.x - fn.x - fm_n.x and res; i += 1) res = res * p.x % p.y;
    return res;
}
LL exLucas(LL m, LL n, LL p){
    vector<PLL> vp;
    LL np = p;
    for(int i = 2; i * i <= np; i += 1) if(np % i == 0){
        vp.push_back({i, 1});
        for(; np % i == 0; np /= i) vp.back().y *= i;
    }
    if(np > 1) vp.push_back({np, np});
    vector<LL> r;
    for(auto p : vp) r.push_back(exLucas(m, n, p));
    LL res = 0;
    for(int i = 0; i < (int)r.size(); i += 1)
        res = (res + r[i] * (p / vp[i].y) % p * inverse(p / vp[i].y, vp[i])) % p;
    return res;
```

```
}
```

6.6 Min 25's method

```
using LL = long long;
constexpr int maxn = 120000;
constexpr LL mod = 1000000007;
int p[maxn], pn, ps[maxn], pg[maxn], m;
bool cp[maxn];
LL sp[3][maxn], g[3][maxn << 1], v[maxn << 1], on;
LL add(LL x, const LL& y){x += y; return x >= mod ? x - mod : x;}
LL sub(LL x, const LL& y){x -= y; return x < 0 ? x + mod : x;}
LL s1(LL n){n %= mod; n = n * (n + 1) / 2 % mod; return n ? n - 1 : mod - 1;}
LL s2(LL n){n %= mod; n = n * (n + 1) % mod * (2 * n + 1) % mod * 166666668 % mod;
    return n ? n - 1 : mod - 1;}
LL id(LL n){return n <= m ? ps[n] : pg[on / n];}
LL S(LL n, int k){
    if(k == 0){
        m = sqrt(on = n) + 1;
        for(int i = 2; i <= m; i += 1){
            if(not cp[i]){
                p[pn += 1] = i;
                sp[1][pn] = add(sp[1][pn - 1], i);
                sp[2][pn] = add(sp[2][pn - 1], (LL)i * i % mod);
                //...
            }
            for(int j = 1; j <= pn and i * p[j] <= m; j += 1){
                cp[i * p[j]] = true;
                if(i % p[j] == 0) break;
            }
        }
        int gn = 0;

        for(LL L = 1, R; L <= n; L = R + 1){
            R = n / (n / L);
            if(n / L > m) pg[R] = gn += 1;
            else ps[n / L] = gn += 1;
            v[gn] = n / L;
            g[1][gn] = s1(v[gn]);
            g[2][gn] = s2(v[gn]);
            //...
        }

        for(int i = 1; i <= pn; i += 1){
            LL p2 = (LL)p[i] * p[i];
            for(int j = 1; j <= gn and p2 <= v[j]; j += 1){
                int x = id(v[j] / p[i]);
                g[1][j] = sub(g[1][j], p[i] * sub(g[1][x], sp[1][i - 1]) % mod);
                g[2][j] = sub(g[2][j], p2 % mod * sub(g[2][x], sp[2][i - 1]) % mod);
                //...
            }
        }
    }
    if(p[k] >= n) return 0;
    int x = id(n);
    LL res = sub(sub(g[2][x], sp[2][k]), sub(g[1][x], sp[1][k]));
    //...
    for(int i = k + 1; i <= pn and (LL)p[i] * p[i] <= n; i += 1){
        LL a = p[i], b = a;
        for(int r = 1; a <= n; r += 1, a *= p[i], b = b * p[i] % mod)
            res = add(res, b * (b - 1) % mod * (S(n / a, i) + (r != 1)) % mod);
```

```
            //...
    }
    return add(res, not k);
}
```

6.7 Pollard Rho

Miller Rabin: LibreOJ143: $T = 100000, n = 10^{18}$, time = 346ms.
Pollard Rho: Luogu4718: $T = 350, n = 10^{18}$, time = 1160ms.

```
using LL = long long;
LL gcd(LL a, LL b){
    return b ? gcd(b, a % b) : a;
}
LL mul(LL a, LL b, LL p){
    return(__int128)a * b % p;
    //return (a * b - (LL)(a / (long double)p * b + 1e-3) * p + p) % p;
}
LL power(LL a, LL r, LL p){
    LL res = 1;
    for(; r; a = mul(a, a, p), r >>= 1)
        if(r & 1) res = mul(res, a, p);
    return res;
}
bool Miller_Rabin(LL n){
    static LL p[9] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    if(n == 1) return false;
    if(n == 2) return true;
    if(not(n & 1)) return false;
    LL d = n - 1, r = 0;
    for(;not(d & 1); d >>= 1) r += 1;
    bool res = true;
    for(int i = 0; i < 9 and p[i] < n and res; i += 1){
        LL x = power(p[i], d, n);
        if(x == 1 or x == n - 1) continue;
        for(int j = 1; j < r; j += 1){
            x = mul(x, x, n);
            if(x == n - 1) break;
        }
        if(x != n - 1) res = false;
    }
    return res;
}
void Pollard_Rho(LL n){
    if(n == 1) return;
    if(Miller_Rabin(n)){
        //n is prime factor
        return;
    }
    LL d = n;
    while(d == n){
        d = 1;
        for(LL k = 1, y = 0, x = 0, s = 1, c = rand() % n; d == 1; k <<= 1, y = x, s =
            1){
            for(int i = 1; i <= k; i += 1){
                x = (mul(x, x, n) + c) % n;
                s = mul(s, abs(x - y), n);
                if(not(i % 127) or i == k){
                    d = gcd(s, n);
                    if(d != 1) break;
                }
            }
```

```
        }
    }
    Pollard_Rho(d);
    Pollard_Rho(n / d);
}
```

7 numeric

7.1 ternary search with golden section

```
constexpr double Phi = (sqrt(5) - 1) / 2;
constexpr int step = 50;
double f(double x){
    //cost function
}
void ternary_search(double f(double), double L, double R){
    //find mininum
    double mL = Phi * L + (1 - Phi) * R;
    double mR = Phi * R + (1 - Phi) * L;
    double fmL = f(mL), fmR = f(mR);
    for(int i = 0; i < step; i += 1)
        if(fmL > fmR){
            L = mL;
            mL = mR;
            fmL = fmR;
            fmR = f(mR = Phi * R + (1 - Phi) * L);
        }
        else{
            R = mR;
            mR = mL;
            fmR = fmL;
            fmL = f(mL = Phi * L + (1 - Phi) * R);
        }
}
```