杂

初始

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

using i64 = long long;

void solve () {
}
// 修一下爆没爆int
// 多测

signed main () {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int t = 1;
    cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
```

对拍

- 一共 4 个文件：
  - `baoli.cpp`
  - `std.cpp`
  - `data.cpp`
    - 关键

```
std::mt19937
rng(std::chrono::steady_clock::now().time_since_epoch().count());

int gen(int min, int max) {
    std::uniform_int_distribution<long long> dis(min, max);
    return dis(rng);
}

shuffle(v.begin(), v.end(), rng);
```

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

using i64 = long long;

std::mt19937
rng(std::chrono::steady_clock::now().time_since_epoch().count());

int gen(int min, int max) {
    std::uniform_int_distribution<long long> dis(min, max);
    return dis(rng);
}

int  main () {
    std::ios::sync_with_stdio (false);
    std::cin.tie (nullptr) ;
    vector<int> a;
    shuffle(a.begin(), a.end(), rng);

    return 0 ;
}
```

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

    void solve() {
    ```
    ```

```
        for (int i = 1; ; i += 1) {
            system("data.exe > in.txt");
            system("std.exe < in.txt > std.txt");
            system("baoli.exe < in.txt > baoli.txt");
            if (system("fc std.txt baoli.txt")) {
                cout << "case " << i << " Wrong Answer" << endl;
                system("pause");
            } else {
                cout << "case " << i << " Accepted Answer" << endl;
            }
        }
    }

    signed main() {
        ios::sync_with_stdio(0);
        cin.tie(0);
        signed t = 1;
    //    cin >> t;
        while (t --) {
            solve();
        }
        return 0;
    }
```

简易版取模类

```
template<typename T>
T power(T x, long long b) {
    T res = 1;
    while (b) {
        if (b & 1) res *= x;
        x *= x;
        b >>= 1;
    }
    return res;
}
```

```cpp
template<int P>
struct mod_int {
    int x;
    static int mod;
    mod_int() : x{} {}
    mod_int(long long x) : x(norm(x % getMod())) {}

    int norm(int x) {
        if (x >= P) x -= P;
        if (x < 0) x += P;
        return x;
    }

    static void setMod(int x) {
        mod = x;
    }
    static int getMod() {
        return (P > 0 ? P : mod);
    }
    mod_int operator-() {
        return -x;
    }

    mod_int &operator+=(mod_int rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    mod_int &operator-=(mod_int rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    mod_int &operator*=(mod_int rhs) {
        x = 1ll * x * rhs.x % getMod();
        return *this;
    }

    mod_int inv() {
```

```
        return power(*this, P - 2);
    }
    mod_int &operator/=(mod_int rhs) {
        x = 1ll * x * rhs.inv().x % getMod();
        return *this;
    }

    friend mod_int operator+(mod_int lhs, mod_int rhs) {
        return lhs += rhs;
    }
    friend mod_int operator-(mod_int lhs, mod_int rhs) {
        return lhs -= rhs;
    }
    friend mod_int operator*(mod_int lhs, mod_int rhs) {
        return lhs *= rhs;
    }
    friend mod_int operator/(mod_int lhs, mod_int rhs) {
        return lhs /= rhs;
    }
    friend bool operator==(mod_int lhs, mod_int rhs) {
        return lhs.x == rhs.x;
    }
    friend bool operator!=(mod_int lhs, mod_int rhs) {
        return lhs.x != rhs.x;
    }

    template<class istream>
    friend istream &operator>>(istream &input, mod_int &rhs) {
        long long x;
        input >> x;
        rhs = x;
        return input;
    }
    template<class ostream>
    friend ostream &operator<<(ostream &output, mod_int rhs) {
        return output << rhs.x;
    }
};
```

```
template<>
int mod_int<0>::mod = 998244353;

constexpr int P = 1e9 + 7;
using Z = mod_int<P>;
```

取模类丐版

```
struct Z {
    static constexpr int P = 998244353;
    int x = 0;
    Z() {}
    Z(i64 x) : x(norm(x % P)) {}
    int norm(int x) {
        if (x >= P) {
            x -= P;
        }
        if (x < 0) {
            x += P;
        }
        return x;
    }
    Z operator-() {
        return -x;
    }
    Z &operator+=(Z rhs) {
        x = norm(x + rhs.x);
        return *this;
    }
    Z &operator-=(Z rhs) {
        x = norm(x - rhs.x);
        return *this;
    }
    Z &operator*=(Z rhs) {
        x = 1ll * x * rhs.x % P;
        return *this;
    }
```

```
    friend Z operator+(Z lhs, Z rhs) {
        return lhs += rhs;
    }
    friend Z operator-(Z lhs, Z rhs) {
        return lhs -= rhs;
    }
    friend Z operator*(Z lhs, Z rhs) {
        return lhs *= rhs;
    }
    friend istream &operator>>(istream &cin, Z &rhs) {
        i64 x;
        cin >> x;
        rhs = x;
        return cin;
    }
    friend ostream &operator<<(ostream &cout, Z rhs) {
        return cout << rhs.x;
    }
};
```

debug.h

```
template<typename A, typename B>
ostream &operator<<(ostream &cout, const pair<A, B> &p) {
    return cout << '(' << p.first << ", " << p.second << ')';
}
template<typename Tp, typename T = typename
    enable_if<!is_same<Tp, string>::value, typename Tp::value_type>::type>
ostream &operator<<(ostream &cout, const Tp &v) {
    cout << '{';
    string sep;
    for (const T &x : v)
        cout << sep << x, sep = ", ";
    return cout << '}';
}
```

```
void Output() { cerr << endl; }
template<typename Head, typename... Tail>
void Output(Head H, Tail... T) {
    cerr << ' ' << H; Output(T...);
}

# define ps cerr << "YES" << endl
# define debug(...) \
         cerr << "(" << #__VA_ARGS__ << "):" << endl,\
         Output(__VA_ARGS__)
```

hash

```
struct Hash {
  static uint64_t splitmix64(uint64_t x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }

  size_t operator()(uint64_t x) const {
    static const uint64_t FIXED_RANDOM =
        chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }

  // 针对 std::pair<int, int> 作为主键类型的哈希函数
  size_t operator()(pair<uint64_t, uint64_t> x) const {
    static const uint64_t FIXED_RANDOM =
        chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(x.first + FIXED_RANDOM) ^
           (splitmix64(x.second + FIXED_RANDOM) >> 1);
  }
};
```

O2优化

```
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,avx2,fma")
#pragma GCC optimize("unroll-loops")
```

快读

```
struct Input {
    using i64 = long long;
    Input() {}
    static constexpr int MAXSIZE = 1 << 20;

    char buf[MAXSIZE], *p1 = buf, *p2 = buf;
    # define isdigit(x) ('0' <= x && x <= '9')

    #define gc()                                                          \
       (p1 == p2 &&(p2 =(p1 = buf) + fread(buf, 1, MAXSIZE, stdin), p1 == p2) \
            ? EOF                                                         \
            : *p1++)

    bool blank(char ch) {
        return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t' || ch == EOF;
    }
    void tie(int x) {}
    template <typename T>
    Input &operator>>(T &x) {
        x = 0;
        bool sign = 0;
        char ch = gc();
        for (; !isdigit(ch); ch = gc())
            if(ch == '-') sign = 1;
        for (; isdigit(ch); ch = gc())
            x = (x << 3) + (x << 1) + ch - '0';
        if(sign) x = -x;
        return *this;
    }
```

```
    Input &operator>>(char &x) {
        x = ' ';
        for (; blank(x); x = gc());
        return *this;
    }
    Input &operator>>(double &x) {
        x = 0;
        double tmp = 1;
        bool sign = 0;
        char ch = gc();
        for (; !isdigit(ch); ch = gc())
            if(ch == '-') sign = 1;
        for (; isdigit(ch); ch = gc())
            x = x * 10 + ch - '0';
        if(ch == '.')
        for (ch = gc(); isdigit(ch); ch = gc())
            tmp /= 10.0, x += tmp *(ch - '0');
        if(sign) x = -x;
        return *this;
    }
    Input &operator>>(string &s) {
        s.clear();
        char ch = gc();
        for (; blank(ch); ch = gc());
        for (; !blank(ch); ch = gc()) {
            s += ch;
        }
        return *this;
    }
    # undef isdigit
    # undef gc
}input;
# define cin input
struct Output {
    struct setprecision {
        int precision;
    };
    static constexpr int MAXSIZE = 1 << 20;
```

```
    char pbuf[MAXSIZE], *pp = pbuf;
    void push(const char &c) {
        if(pp - pbuf == MAXSIZE)
            fwrite(pbuf, 1, MAXSIZE, stdout), pp = pbuf;
        *pp++ = c;\
    }
    int precision;
    Output() { precision = 6;}
    ~Output() { fwrite(pbuf, 1, pp - pbuf, stdout);}
    char stack[40];
    int top = 0;
    template<class T>
    Output &operator<<(const T &x) {
        T tmp = x;
        bool _ = tmp < 0;
        if(_) tmp *= -1;
        while(tmp) stack[++ top] = '0' + tmp % 10, tmp /= 10;
        if(_) stack[++ top] = '-';
        while(top) push(stack [top]), -- top;
        if(x == 0)push('0');
        return *this;
    }
    Output &operator<<(const string &x) {
        for (auto &u : x) push(u);
        return *this;
    }
    template<size_t N>
    Output &operator<<(const char(&x)[N]) {
        *this << string(x);
        return *this;
    }
    Output &operator<<(const char* const &x) {
        for (const char* ptr = x; *ptr != '\0'; ++ptr)
            push(*ptr);
        return *this;
    }
    Output &operator<<(const char &x) {
        push(x);
```

```
        return *this;
    }
    Output &operator<<(const bool &x) {
        push(x ? '1' : '0');
        return *this;
    }
    Output &operator<<(const double &x) {
        int intPart = static_cast<int>(x);
        *this << intPart;

        push('.');

        double decimalPart = x - intPart;
        for (int i = 0; i < precision; ++i) {
            decimalPart *= 10;
            int digit = static_cast<int>(decimalPart);
            *this << char('0' + digit);
            decimalPart -= digit;
        }
        return *this;
    }
    Output &operator<<(setprecision x) {
        precision = x.precision;
        return *this;
    }
    # undef push
}output;
# define cout output
```

u32 指针

```
/**
 * 1 MB = 1024 KB
 * 1 KB = 1024 B
 * 134'210'000    128
 * 262'144'000    256
 * 520'000'000    524
```

```
 * 1'030'000'000  1024
 * 注意事项：记得内存别开小了或者别爆了
 */

constexpr int max_size = 1030000000;
uint8_t buf[max_size];
uint8_t *head = buf;

using u32 = uint32_t;

template <class T>
struct Base {
    u32 x;
    Base(u32 x = 0) : x(x) {}
    T *operator->() {
        return (T *)(buf + x);
    }
    T &operator*() {
        return *((T *)(buf + x));
    }
    operator bool() {
        return x;
    }
    operator u32() {
        return x;
    }
    bool operator==(Base rhs) const {
        return x == rhs.x;
    }
    static Base alloc() {
        return (head += sizeof(T)) - buf;
    }
};
```