

对拍

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    // For Windows
    // 对拍时不开文件输入输出
    // 当然，这段程序也可以改写成批处理的形式
    while (true) {
        system("data.exe > test.in"); // 数据生成器将生成数据写入输入文件
        system("solve.exe < test.in > solve.out"); // 获取程序1输出
        system("std.exe < test.in > std.out"); // 获取程序2输出
        if (system("diff solve.out std.out")) {
            // 该行语句比对输入输出
            // fc返回0时表示输出一致，否则表示有不同处
            system("pause"); // 方便查看不同处
            return 0;
            // 该输入数据已经存放在test.in文件中，可以直接利用进行调试
        }
    }
}
```

```
import os
tc = 0
os.system("g++ std.cpp -o std.exe")
os.system("g++ solve.cpp -o solve.exe")

while True:
    os.system("python data.py > data.in")
    os.system("std.exe < data.in > std.out")
    os.system("solve.exe < data.in > solve.out");
    if(os.system("diff std.out solve.out")):
        print("WA")
        exit(0)
    else:
        tc += 1
        print("AC #%d" %(tc))
```

random.randrange(10, 100 - 1) 闭区间

`__int128` 快读快写 O2/O3加速

kun神的最爱

```

#pragma GCC optimize("O2")
#pragma GCC optimize("O3")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC target("avx,avx2,fma")
#pragma GCC target("sse4,popcnt,abm,mmx")

using i128 = __int128;
// 重载输入运算符以支持__int128类型
std::istream &operator>>(std::istream &is, __int128 &val) {
    std::string str; is >> str; val = 0;
    bool neg = false;
    if (str[0] == '-') neg = true, str = str.substr(1);
    for (char &c: str) val = val * 10 + c - '0';
    if (neg) val = -val;
    return is;
}

//重载输出运算符以支持__int128类型
std::ostream &operator<<(std::ostream &os, __int128 val) {
    if (val < 0) os << "-", val = -val;
    if (val > 9) os << val / 10;
    os << static_cast<char>(val % 10 + '0');
    return os;
}

__int128 read() {
    __int128 x=0, f=1;
    char c=getchar();
    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;
}

void write(__int128 x) {
    if(x<0) putchar('-'), x=-x;
    if(x>9) write(x/10);
    putchar(x%10+'0');
}

```

单测多测

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

#define ranges std::ranges
#define views std::views

using u32 = unsigned;
using i64 = long long;
using u64 = unsigned long long;

using pii = std::pair<int, int>;
using a3 = std::array<int, 3>;
using a4 = std::array<int, 4>;

const int N = 1e6;
const int MAXN = 1e6 + 10;
const int inf = 1e9;
// const int mod = 1e9 + 7;
const int mod = 998244353;

std::mt19937_64 rng(std::chrono::steady_clock::now().time_since_epoch().count());

void solve() {

}

signed main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0), std::cout.tie(0);
    int t = 1; // cin >> t;
    while (t--) {
        solve();
        std::cout << '\n';
    }
    return 0;
}
```

取模类

```
struct Mi64 {  
    i64 value;  
    Mi64() {}  
    Mi64(i64 value) { this->value = value; }  
    Mi64 operator * (const i64 w) {  
        return value * w % mod;  
    }  
    Mi64 operator * (const Mi64 w) {  
        return value * w.value % mod;  
    }  
    Mi64 operator + (const Mi64 w) {  
        return (value + w.value) % mod;  
    }  
    Mi64 operator - (const Mi64 w) {  
        return((value - w.value) % mod + mod) % mod;  
    }  
    bool operator == (const Mi64 w) {  
        return value == w.value;  
    }  
}
```

```

using i64=long long;
template<class T>
constexpr T power(T a, i64 b) {
    T res = 1;
    for (; b; b /= 2, a *= a) {
        if (b % 2) {
            res *= a;
        }
    }
    return res;
}

constexpr i64 mul(i64 a, i64 b, i64 p) {
    i64 res = a * b - i64(1.L * a * b / p) * p;
    res %= p;
    if (res < 0) {
        res += p;
    }
    return res;
}

template<i64 P>
struct MLong {
    i64 x;
    constexpr MLong() : x{} {}
    constexpr MLong(i64 x) : x{norm(x % getMod())} {}

    static i64 Mod;
    constexpr static i64 getMod() {
        if (P > 0) {
            return P;
        } else {
            return Mod;
        }
    }
}

constexpr static void setMod(i64 Mod_) {
    Mod = Mod_;
}

constexpr i64 norm(i64 x) const {
    if (x < 0) {
        x += getMod();
    }
    if (x >= getMod()) {
        x -= getMod();
    }
}

```

```

    }
    return x;
}
constexpr i64 val() const {
    return x;
}
explicit constexpr operator i64() const {
    return x;
}
}
constexpr MLong operator-() const {
    MLong res;
    res.x = norm(getMod() - x);
    return res;
}
constexpr MLong inv() const {
    assert(x != 0);
    return power(*this, getMod() - 2);
}
constexpr MLong &operator*=(MLong rhs) & {
    x = mul(x, rhs.x, getMod());
    return *this;
}
constexpr MLong &operator+=(MLong rhs) & {
    x = norm(x + rhs.x);
    return *this;
}
constexpr MLong &operator--(MLong rhs) & {
    x = norm(x - rhs.x);
    return *this;
}
constexpr MLong &operator/=(MLong rhs) & {
    return *this *= rhs.inv();
}
}
friend constexpr MLong operator*(MLong lhs, MLong rhs) {
    MLong res = lhs;
    res *= rhs;
    return res;
}
friend constexpr MLong operator+(MLong lhs, MLong rhs) {
    MLong res = lhs;
    res += rhs;
    return res;
}
}

```

```

friend constexpr MLong operator-(MLong lhs, MLong rhs) {
    MLong res = lhs;
    res -= rhs;
    return res;
}
friend constexpr MLong operator/(MLong lhs, MLong rhs) {
    MLong res = lhs;
    res /= rhs;
    return res;
}
friend constexpr std::istream &operator>>(std::istream &is, MLong &a) {
    i64 v;
    is >> v;
    a = MLong(v);
    return is;
}
friend constexpr std::ostream &operator<<(std::ostream &os, const MLong &a) {
    return os << a.val();
}
friend constexpr bool operator==(MLong lhs, MLong rhs) {
    return lhs.val() == rhs.val();
}
friend constexpr bool operator!=(MLong lhs, MLong rhs) {
    return lhs.val() != rhs.val();
}
};

```

```

template<>
i64 MLong<0LL>::Mod = i64(1E18) + 9;

```

```

template<int P>
struct MInt {
    int x;
    constexpr MInt() : x{} {}
    constexpr MInt(i64 x) : x{norm(x % getMod())} {}

    static int Mod;
    constexpr static int getMod() {
        if (P > 0) {
            return P;
        } else {
            return Mod;
        }
    }
};

```



```

}
constexpr static void setMod(int Mod_) {
    Mod = Mod_;
}
constexpr int norm(int x) const {
    if (x < 0) {
        x += getMod();
    }
    if (x >= getMod()) {
        x -= getMod();
    }
    return x;
}
constexpr int val() const {
    return x;
}
explicit constexpr operator int() const {
    return x;
}
constexpr MInt operator-() const {
    MInt res;
    res.x = norm(getMod() - x);
    return res;
}
constexpr MInt inv() const {
    assert(x != 0);
    return power(*this, getMod() - 2);
}
constexpr MInt &operator*=(MInt rhs) & {
    x = 1LL * x * rhs.x % getMod();
    return *this;
}
constexpr MInt &operator+=(MInt rhs) & {
    x = norm(x + rhs.x);
    return *this;
}
constexpr MInt &operator-=(MInt rhs) & {
    x = norm(x - rhs.x);
    return *this;
}
constexpr MInt &operator/=(MInt rhs) & {
    return *this *= rhs.inv();
}

```

```

friend constexpr MInt operator*(MInt lhs, MInt rhs) {
    MInt res = lhs;
    res *= rhs;
    return res;
}
friend constexpr MInt operator+(MInt lhs, MInt rhs) {
    MInt res = lhs;
    res += rhs;
    return res;
}
friend constexpr MInt operator-(MInt lhs, MInt rhs) {
    MInt res = lhs;
    res -= rhs;
    return res;
}
friend constexpr MInt operator/(MInt lhs, MInt rhs) {
    MInt res = lhs;
    res /= rhs;
    return res;
}
friend constexpr std::istream &operator>>(std::istream &is, MInt &a) {
    i64 v;
    is >> v;
    a = MInt(v);
    return is;
}
friend constexpr std::ostream &operator<<(std::ostream &os, const MInt &a) {
    return os << a.val();
}
friend constexpr bool operator==(MInt lhs, MInt rhs) {
    return lhs.val() == rhs.val();
}
friend constexpr bool operator!=(MInt lhs, MInt rhs) {
    return lhs.val() != rhs.val();
}
};

template<>
int MInt<0>::Mod = 998244353;

template<int V, int P>
constexpr MInt<P> CInv = MInt<P>(V).inv();

```

```
constexpr int P = 1000000007;  
using Z = MInt<P>;
```

pbds 排名树

牛客小白96 E 完整示例

```

#include <bits/stdc++.h>
#include <bits/extc++.h>
using namespace __gnu_pbds;
using i64 = long long;
constexpr int MAXN = 1e6 + 10, inf = 1e9, mod = 1e9 + 7;
using pii = std::pair<i64, int>;
using int_rb_tree = tree<pii, null_type, std::less<pii>, rb_tree_tag, tree_order_statistics_node_update>;
i64 n, a[MAXN], is_z[MAXN], z_child[MAXN], s_child[MAXN], cj[MAXN];
std::vector<int> adj[MAXN];

pii dfs1(int x, i64 sum) {
    i64 child_sum = 0, zhic_sum = 0;
    for (int nxt : adj[x]) {
        auto [c_sum, z_sum] = dfs1(nxt, sum + a[x]);
        child_sum += c_sum;
        zhic_sum += z_sum;
    }
    if (child_sum <= a[x] && a[x] <= sum) zhic_sum++, is_z[x] = 1;
    else if (a[x] <= sum && x) {
        cj[x] = child_sum - a[x];
    }
    z_child[x] = zhic_sum;
    s_child[x] = child_sum;
    return { child_sum + a[x], zhic_sum };
}

i64 ans = 0, t = 0;
int_rb_tree rbtree;
void dfs2(int x, i64 sum) {
    int curt = t++;
    ans = std::max(ans, (i64)rbtree.order_of_key({ s_child[x] + a[x], curt }) + z_child[0] - z_child[x]);
    if (cj[x])
        rbtree.insert({ cj[x], curt });
    for (int nxt : adj[x]) {
        dfs2(nxt, sum + a[x]);
    }
    if (cj[x])
        rbtree.erase({ cj[x], curt });
}

signed main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0), std::cout.tie(0);

```

```

std::cin >> n;
for (int i = 1; i <= n; ++i) std::cin >> a[i];
for (int i = 1; i <= n; ++i) {
    int fa; std::cin >> fa;
    adj[fa].push_back(i);
}
dfs1(0, 0);
ans = z_child[0];
dfs2(0, 0);
std::cout << ans;
return 0;
}

```

div

```

i64 ceilDiv(i64 n, i64 m) {
    if (n >= 0) {
        return (n + m - 1) / m;
    } else {
        return n / m;
    }
}

```

```

i64 floorDiv(i64 n, i64 m) {
    if (n >= 0) {
        return n / m;
    } else {
        return (n - m + 1) / m;
    }
}

```