Inverse in O(1)

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

using namespace std;

constexpr int mod = 1e9 + 696969;
constexpr int n = pow(mod, 1.0 / 3) + 1;
constexpr int n2 = n * n;
constexpr int mod_n = mod / n;

int p[n2 + 1];
int f[n2 + 1];
int inv[mod_n + 1];

void precalc() {
	for (int y = 1; y < n; y++) {
		for (int x = 0; x <= y; x++) {
			int i = x * n2 / y;
			if (!p[i]) {
				p[i] = x * n + y;
			}
		}
	}

	int f_cnt = 0;
	for (int i = 0; i <= n2; i++) {
		if (p[i]) {
			f[f_cnt++] = p[i];
		}
		p[i] = f_cnt;
	}

	inv[1] = 1;
	for (int i = 2; i <= mod_n; i++)
		inv[i] = mod - (long long) (mod / i) * inv[mod % i] % mod;
}

int inverse(int a) {
	int i = p[(long long) a * n2 / mod];
	int x = f[i] / n;
	int y = f[i] % n;
	int u = a * y - mod * x;

	if (abs(u) > mod_n) {
		i--;
		x = f[i] / n;
		y = f[i] % n;
		u = a * y - mod * x;
	}

	assert(abs(u) <= mod_n);
	return (long long) y * (u < 0 ? mod - inv[-u] : inv[u]) % mod;
}

mt19937 rng(2137);

int randint(int a, int b) {
	return uniform_int_distribution<int>(a, b)(rng);
}

int main() {
	precalc();

	for (int x = 1; x < mod; x++) {
		int x_inv = inverse(x);
		assert((long long) x * x_inv % mod == 1 || gcd(x, mod) > 1);
	}
	return 0;
}
```