Compilando Conocimiento

Refence

Competitive Programming

Rosas Hernandez Oscar Andrés

July 2018

Contents

Part I

Things to Learn / To Do

Chapter 1

C++

1.1 Integrals

1.1.1 int vs long vs long long

```
int minValue {-2,147,483,648};
int maxValue {2,147,483,647};

long long minValue {-9,223,372,036,854,775,808};
long long maxValue {9,223,372,036,854,775,807};

unsigned int maxValueIntUnsigned {4,294,967,295};
unsigned long long maxValueLLUnsigned
    {18,446,744,073,709,551,615};
```

1.1.2 Fixed width (int32_t, uint64_t, ...)

```
#include <cstdint>

int8_t likeChar {};
int16_t likeShort {};
int32_t likeInt {};
int64_t likeLong {};

// And the unsigned versions:
uint8_t likeChar {};
uint16_t likeShort {};
uint32_t likeInt {};
uint64_t likeLong {};
```

1.1.3 Fast I / O

```
// No merge cin & cout with scanf & printf
ios::sync_with_stdio(false);

// No merge cin / cout
cin.tie(nullptr);
```

```
template <class T>
inline void getNumberFast(T &result) {
    T number {};
    T sign {1};

    char currentDigit {getchar_unlocked()};

    while(currentDigit < '0' or currentDigit > '9') {
        currentDigit = getchar_unlocked();
        if (currentDigit == '-')  sign = -1;
    }

    while ('0' <= currentDigit and currentDigit <= '9') {
        number = (number << 3) + (number << 1);
        number += currentDigit - '0';
        currentDigit = getchar_unlocked();
    }

    if (sign) result = -number;
    else result = number;
}
```

Part II

Number Theory

Chapter 2

Primes

2.1 Sieve of Eratosthenes

2.1.1 Get the Boolean Version

```
template<typename T>
auto getIsPrime(T maxValue) -> std::vector<bool> {
    std::vector<bool> isPrime (maxValue + 1, true);
    isPrime[0] = isPrime[1] = false;

    for (T i {4}; i <= maxValue; i += 2) isPrime[i] = false;

    for (T i {3}; i * i <= maxValue; i += 2) {
        if (not isPrime[i]) continue;

        T multiple {i * i}, step {2 * i};
        while (multiple <= maxValue) {
            isPrime[multiple] = false;
            multiple += step;
        }
    }

    return isPrime;
}
```

2.1.2 Get the Vector of Primes

```
template<typename T>
auto getPrimes(T maxValue) -> std::vector<T> {
    std::vector<bool> isPrime (maxValue + 1, true);
    std::vector<T> primes {2};

    // Just to do it if you need the bools too.
    // isPrime[0] = isPrime[1] = false;
    // for (T i = 4; i <= n; i += 2) isPrime[i] = false;

    for (T i {3}; i <= maxValue; i += 2) {
        if (not isPrime[i]) continue;
        primes.push_back(i);

        T multiple {i * i}, step {2 * i};
        while (multiple <= maxValue) {
            isPrime[multiple] = false;
            multiple += step;
        }
    }

    return primes;
}
```