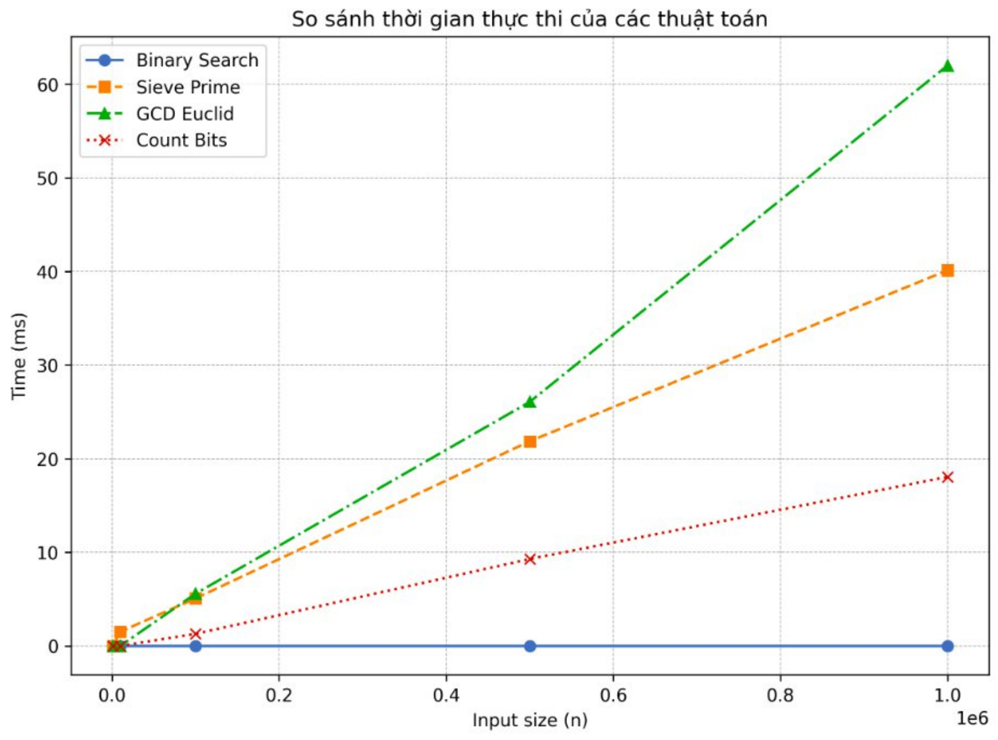
Algorithm Timing Results

Algorithm	Case	Input (n)	Time (ms)
Binary Search	Best	1000	0
	Average	1000	0
	Worst	1000	0
Sieve Prime	Best	1000	0
	Average	1000	0
	Worst	1000	0
GCD (Euclid)	Best	1000	0
	Average	1000	0
	Worst	1000	0
Count Bits	Best	1000	0
	Average	1000	0
	Worst	1000	0
Binary Search	Best	10000	0
	Average	10000	0
	Worst	10000	0
Sieve Prime	Best	10000	0
	Average	10000	0
	Worst	10000	0
GCD (Euclid)	Best	10000	0
	Average	10000	0
	Worst	10000	0
Count Bits	Best	10000	0
	Average	10000	0
	Worst	10000	0
Binary Search	Best	100000	0
	Average	100000	0
	Worst	100000	0
Sieve Prime	Best	100000	0
	Average	100000	0
	Worst	100000	5.352
GCD (Euclid)	Best	100000	0
	Average	100000	0
	Worst	100000	0
Count Bits	Best	100000	0
	Average	100000	0
	Worst	100000	0
Binary Search	Best	500000	0
	Average	500000	0
	Worst	500000	0
Sieve Prime	Best	500000	0
	Average	500000	2.018
	Worst	500000	13.081
GCD (Euclid)	Best	500000	0
	Average	500000	0

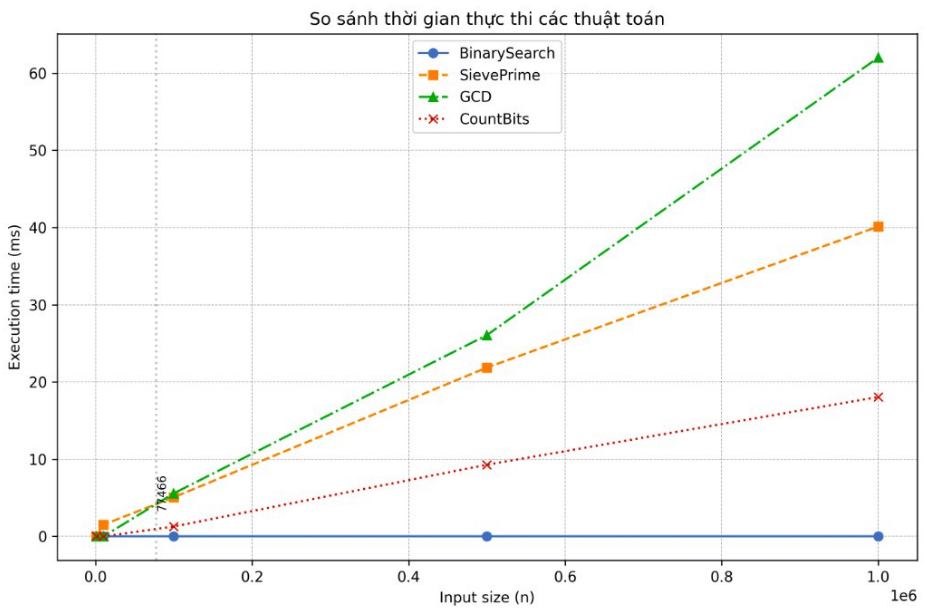
	Worst	500000	0
Count Bits	Best	500000	0
	Average	500000	0
	Worst	500000	0
Binary Search	Best	1000000	0
	Average	1000000	0
	Worst	1000000	0
Sieve Prime	Best	1000000	0
	Average	1000000	2.041
	Worst	1000000	42.152
GCD (Euclid)	Best	1000000	0
	Average	1000000	0
	Worst	1000000	0
Count Bits	Best	1000000	0
	Average	1000000	0
	Worst	1000000	0

Bảng so sánh khi thay đổi input :

STT	Input	Binary Search (ms)	Sieve Prime(ms)	GCD(ms)	Count Bits(ms)
0	1000	0	0.000	0.000	0.000
1	10000	0	1.515	0.000	0.000
2	100000	0	5.073	5.579	1.286
3	500000	0	21.863	26.068	9.282
4	1000000	0	40.126	62.008	18.057



Biểu đồ so sánh thời gian thực thi của các thuật toán.



Biểu đồ xác định điểm giao.

CODE :

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

using namespace std;

using namespace chrono;


int binarySearch(vector<int> &arr, int x) {

    int left = 0, right = arr.size() - 1;

    while (left <= right) {

        int mid = (left + right) / 2;

        if (arr[mid] == x)

            return mid;

        else if (arr[mid] < x)

            left = mid + 1;

        else

            right = mid - 1;
```

```
  }
    return -1;
}

void snt(int n, vector<int> &primes) {
    primes.clear();
    vector<bool> is_prime(n + 1, true);
    is_prime[0] = is_prime[1] = false;
    for (int i = 2; i * i <= n; i++) {
        if (is_prime[i]) {
            for (int j = i * i; j <= n; j += i)
                is_prime[j] = false;
        }
    }
    for (int i = 2; i <= n; i++) {
        if (is_prime[i]) primes.push_back(i);
    }
}

long long gcdEuclid(long long a, long long b) {
    if (b == 0)
        return a;
    return gcdEuclid(b, a % b);
}

int countBits(int n) {
    int count = 0;

    while (n) {

        count += n & 1;

        n >>= 1;

    }

    return count;

}


template <typename Func>

double measureTime(Func f) {

    auto start = high_resolution_clock::now();

    f();

    auto end = high_resolution_clock::now();

    duration<double, milli> elapsed = end - start;

    return elapsed.count();

}


int main() {

    ios::sync_with_stdio(false);

    cin.tie(nullptr);


    srand(time(0));


    vector<int> input_sizes = {1000, 10000, 100000, 500000, 1000000};


    cout << left << setw(20) << "Algorithm"
```

```
        << setw(15) << "Case"
        << setw(15) << "Input(n)"
        << setw(15) << "Time (ms)" << "\n";

    for (int n : input_sizes) {
        vector<int> arr(n);
        iota(arr.begin(), arr.end(), 1);
        vector<int> primes;

        double t_bs_best = measureTime([&]() { binarySearch(arr, arr[n / 2]); });
        double t_bs_avg = measureTime([&]() { binarySearch(arr, arr[n / 4]); });
        double t_bs_worst = measureTime([&]() { binarySearch(arr, n + 1); });

        cout << setw(20) << "Binary Search"
            << setw(15) << "Best" << setw(15) << n << setw(15) << t_bs_best << "\n"
            << setw(20) << "" << setw(15) << "Average" << setw(15) << n << setw(15) <<
t_bs_avg << "\n"
            << setw(20) << "" << setw(15) << "Worst" << setw(15) << n << setw(15) <<
t_bs_worst << "\n";

        double t_sieve_best = measureTime([&]() { snt(1000, primes); });
        double t_sieve_avg = measureTime([&]() { snt(n / 10, primes); });
        double t_sieve_worst = measureTime([&]() { snt(n, primes); });

        cout << setw(20) << "Sieve Prime"
            << setw(15) << "Best" << setw(15) << n << setw(15) << t_sieve_best << "\n"
            << setw(20) << "" << setw(15) << "Average" << setw(15) << n << setw(15) <<
t_sieve_avg << "\n"
            << setw(20) << "" << setw(15) << "Worst" << setw(15) << n << setw(15) <<
t_sieve_worst << "\n";

        double t_gcd_best = measureTime([&]() { gcdEuclid(n, n); });
        double t_gcd_avg = measureTime([&]() { gcdEuclid(n, rand() % n + 1); });
        double t_gcd_worst = measureTime([&]() { gcdEuclid(21, 13); });

        cout << setw(20) << "GCD (Euclid)"
            << setw(15) << "Best" << setw(15) << n << setw(15) << t_gcd_best << "\n"
            << setw(20) << "" << setw(15) << "Average" << setw(15) << n << setw(15) <<
t_gcd_avg << "\n"
            << setw(20) << "" << setw(15) << "Worst" << setw(15) << n << setw(15) <<
t_gcd_worst << "\n";

        double t_bits_best = measureTime([&]() { countBits(1); });
        double t_bits_avg = measureTime([&]() { countBits(rand() % n + 1); });
        double t_bits_worst = measureTime([&]() { countBits(INT_MAX); });

        cout << setw(20) << "Count Bits"
            << setw(15) << "Best" << setw(15) << n << setw(15) << t_bits_best << "\n"
            << setw(20) << "" << setw(15) << "Average" << setw(15) << n << setw(15) <<
t_bits_avg << "\n"
            << setw(20) << "" << setw(15) << "Worst" << setw(15) << n << setw(15) <<
t_bits_worst << "\n";
    }
    return 0;
```

```
}
```