L P S - 8

{ Design and Analysis of Algorithms }

Rachit Bhalla
21BAI1869

1. Given two integers a, b, to find the greatest common factor of a and b.

Problem Statement: Design a algorithm to find greatest common factor of a and b.
Logic: The `gcd` function uses the Euclidean algorithm to find the greatest common factor of two integers `a` and `b`.If `a` is zero, then the GCF of `a` and `b` is `b`.Otherwise, we compute the remainder of `b` divided by `a` using the modulo operator (`%`). We then call the `gcd` function recursively with arguments `b % a` and `a`.
Algorithm:
1.Import all the necessary libraries.
2. The `gcd` function takes two integers `a` and `b` as input and returns their greatest common factor using the Euclidean algorithm.
3.If `a` is equal to 0, then the GCF of `a` and `b` is `b`, so we return `b`.
4. Otherwise, we compute the remainder of `b` divided by `a` using the modulo operator (`%`). We then call the `gcd` function recursively with arguments `b % a` and `a`. This is because the GCF of `a` and `b` is the same as the GCF of `b % a` and `a`.
5. In the `main` function, we prompt the user to enter two integers `a` and `b`. We then call the `gcd` function with arguments `a` and `b`, and store the result in a variable called `result`. Finally, we print out a message to the console telling the user what the GCF of `a` and `b` is.
6. Return all the possible solutions of the given problem.

Pseudocode:

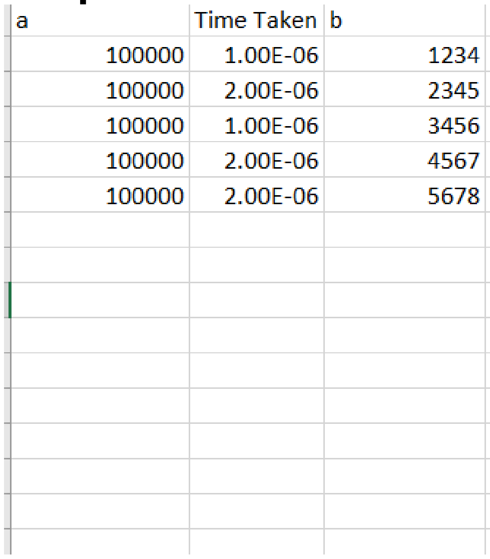
```
#include <iostream>
using namespace std;

int gcd(int a, int b) {
    if (a == 0) {
        return b;
    }
    return gcd(b % a, a);
}

int main() {
    int a, b;
    cout << "Enter two integers: ";
    cin >> a >> b;
    int result = gcd(a, b);
    cout << "The GCF of " << a << " and " << b << " is " << result <<
endl;
    return 0;
}
```

Time Complexity: O(log(min(a,b))

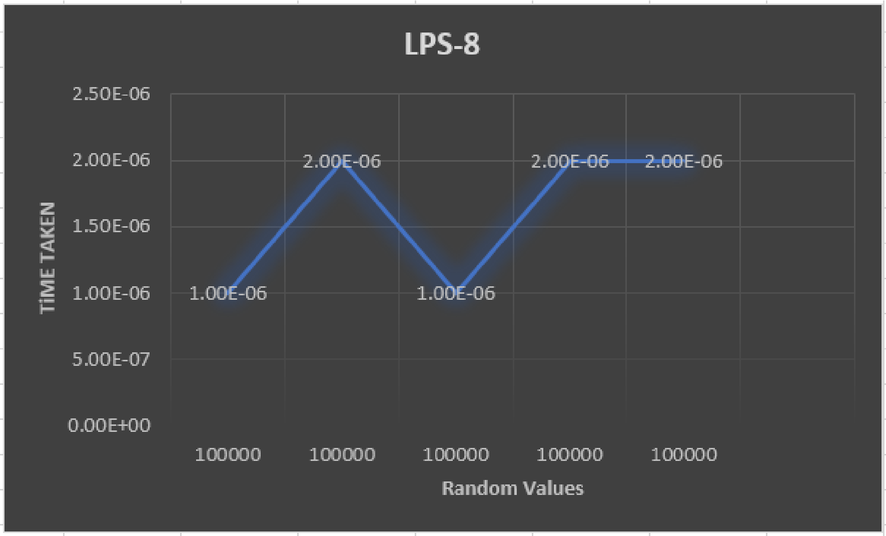
~> The time complexity of the Euclidean algorithm implemented in the `gcd` function is O(log min(a, b)), where a and b are the two input integers. This is because the algorithm repeatedly divides the larger number by the smaller number until the remainder is zero, and each division reduces the size of the larger number by at least a factor of 2. Therefore, the number of iterations required to reach a remainder of zero is proportional to log(min(a, b)). Since each iteration takes constant time, the overall time complexity of the algorithm is O(log min(a, b)).

Proof of Correctness:
Initialization:
The algorithm is initialized with two integers a and b. If a is equal to zero, then the GCF of a and b is b, which is the correct result. Otherwise, we proceed to the maintenance step.

Maintenance:
In each iteration of the algorithm, we compute the remainder of b divided by a using the modulo operator (%). We then call the gcd function recursively with arguments b % a and a. By the inductive step of the proof of correctness, this correctly computes the GCF of b % a and a. Since the GCF of b % a and a is the same as the GCF of a and b, this correctly computes the GCF of a and b.
Termination:
The algorithm terminates when a is equal to zero. At this point, we return b, which is the correct result for the GCF of a and b.
Graph:



a	Time Taken	b
100000	1.00E-06	1234
100000	2.00E-06	2345
100000	1.00E-06	3456
100000	2.00E-06	4567
100000	2.00E-06	5678


LPS-8
TIME TAKEN
2.50E-06
2.00E-06
1.50E-06
1.00E-06
5.00E-07
0.00E+00
1.00E-06
2.00E-06
1.00E-06
2.00E-06
2.00E-06
Random Values