Prime Number of Set Bits in Binary Representation

Given two integers `L` and `R`, find the count of numbers in the range `[L, R]` (inclusive) having a prime number of set bits in their binary representation.

(Recall that the number of set bits an integer has is the number of `1`s present when written in binary. For example, `21` written in binary is `10101` which has 3 set bits. Also, 1 is not a prime.)

Example 1:

```
Input: L = 6, R = 10
Output: 4
Explanation:
6 -> 110 (2 set bits, 2 is prime)
7 -> 111 (3 set bits, 3 is prime)
9 -> 1001 (2 set bits , 2 is prime)
10->1010 (2 set bits , 2 is prime)
```

Example 2:

```
Input: L = 10, R = 15
Output: 5
Explanation:
10 -> 1010 (2 set bits, 2 is prime)
11 -> 1011 (3 set bits, 3 is prime)
12 -> 1100 (2 set bits, 2 is prime)
13 -> 1101 (3 set bits, 3 is prime)
14 -> 1110 (3 set bits, 3 is prime)
15 -> 1111 (4 set bits, 4 is not prime)
```

Note:

1. `L, R` will be integers `L <= R` in the range `[1, 10^6]`.
2. `R - L` will be at most 10000.

Solution 1

Java

```
class Solution {
    public int countPrimeSetBits(int l, int r) {
        Set<Integer> primes = new HashSet<>(Arrays.asList(2, 3, 5, 7, 11, 13, 17, 1
9 /*, 23, 29 */ ));
        int cnt = 0;
        for (int i = l; i <= r; i++) {
            int bits = 0;
            for (int n = i; n > 0; n >>= 1)
                bits += n & 1;
            cnt += primes.contains(bits) ? 1 : 0;
        }
        return cnt;
    }
}
```

C++

```
class Solution {
public:
    int countPrimeSetBits(int l, int r) {
        set<int> primes = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
        int cnt = 0;
        for (int i = l; i <= r; i++) {
            int bits = 0;
            for (int n = i; n; n >>= 1)
                bits += n & 1;
            cnt += primes.count(bits);
        }
        return cnt;
    }
};
```

written by alexander original link here

Solution 2

```
class Solution {
    public int countPrimeSetBits(int L, int R) {
        int cnt = 0;
        Set<Integer> listPrimes = new HashSet<>(Arrays.asList(2, 3, 5, 7, 11, 13, 17, 19, 23, 29 ));
        int[] res = countBits(R);
        for(int i=L;i<=R;i++){
            if(listPrimes.contains(res[i])){
                cnt++;
            }
        }
        return cnt;
    }

    public int[] countBits(int num) {
        if(num == 0)
            return new int[1];
        int[] dp = new int[num+1];

        dp[0] = 0;
        dp[1] = 1;

        for(int i=2;i<=num;i++){
            dp[i] = dp[i >> 1] + dp[i & 1]; //  i >> 1 is i / 2 and i & 1 is i % 2
        }
        return dp;
    }
}
```

written by ashish53v original link here

Solution 3

Ruby:

```ruby
def count_prime_set_bits(l, r)
  (l..r).sum { |i| 665772 >> i.digits(2).sum & 1 }
end
```

Python:

```python
def countPrimeSetBits(self, L, R):
    return sum(665772 >> bin(i).count('1') & 1 for i in range(L, R+1))
```

Java stream:

```java
public int countPrimeSetBits(int L, int R) {
    return IntStream.range(L, R+1).map(i -> 665772 >> Integer.bitCount(i) & 1).sum()
;
}
```

Java:

```java
public int countPrimeSetBits(int L, int R) {
    int count = 0;
    while (L <= R)
        count += 665772 >> Integer.bitCount(L++) & 1;
    return count;
}
```

C++:

```cpp
int countPrimeSetBits(int L, int R) {
    int count = 0;
    while (L <= R)
        count += 665772 >> __builtin_popcount(L++) & 1;
    return count;
}
```

written by StefanPochmann original link here