Binary Exponentiation

What is Binary Exponentiation?

> **Binary Exponentiation** or **Exponentiation by squaring** is the process of calculating a number raised to the power another number (AB) in **Logarithmic** time of the exponent or power, which speeds up the execution time of the program.

Why to Use Binary Exponentiation?

> Whenever we need to calculate (AB), we can simple calculate the result by taking the **result** as 1 and multiplying **A** for exactly **B** times. The time complexity for this approach is O(B) and will fail when values of B in order of 1**0**8** **or greater. This is when we can use Binary exponentiation because it can calculate the result in O(log(B)) time complexity, so we can easily calculate the results for larger values of B in order of **10****18**** or less.

Idea Behind Binary Exponentiation:

When we are calculating (AB), we can have 3 possible positive values of B:

- *Case 1:* If B = 0, whatever be the value of A, our result will be 1.
- **Case 2:*** If B is an even number, then instead of calculating (AB), we can calculate ((A^2)^(B/2)) and the result will be same.
- ****Case 3:**** If B is an odd number, then instead of calculating (AB), we can calculate $(A * (A^{(B-1)/2})^2)$,

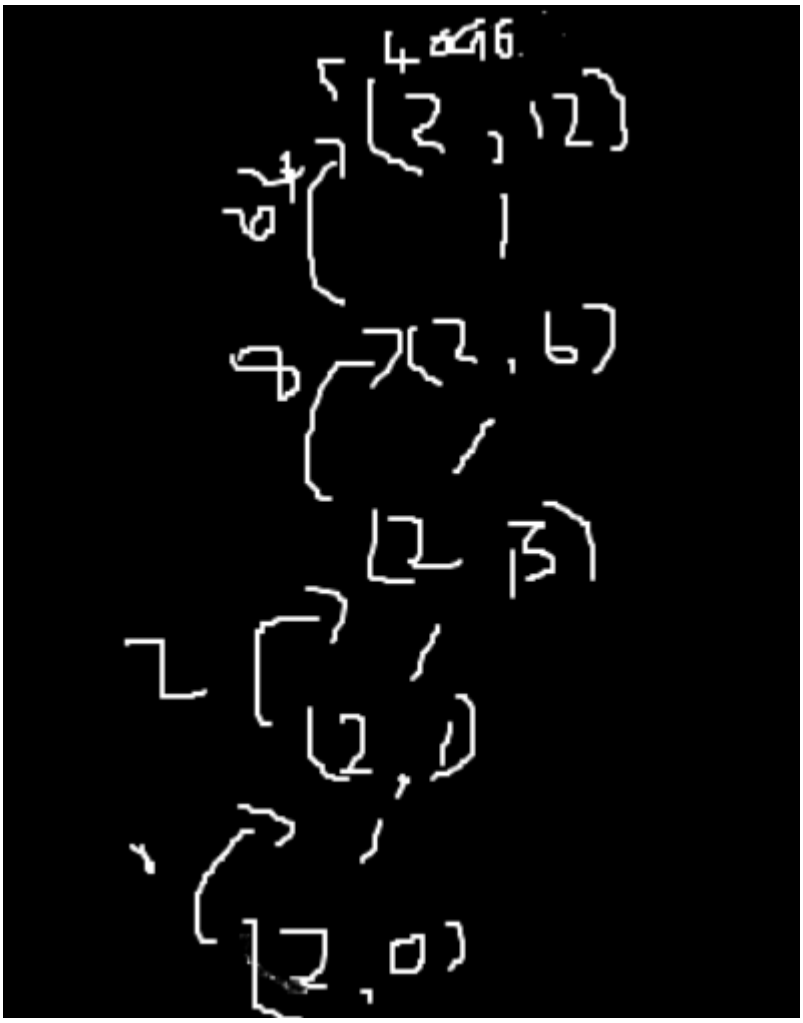
Recursive Implementation of Binary Exponentiation:

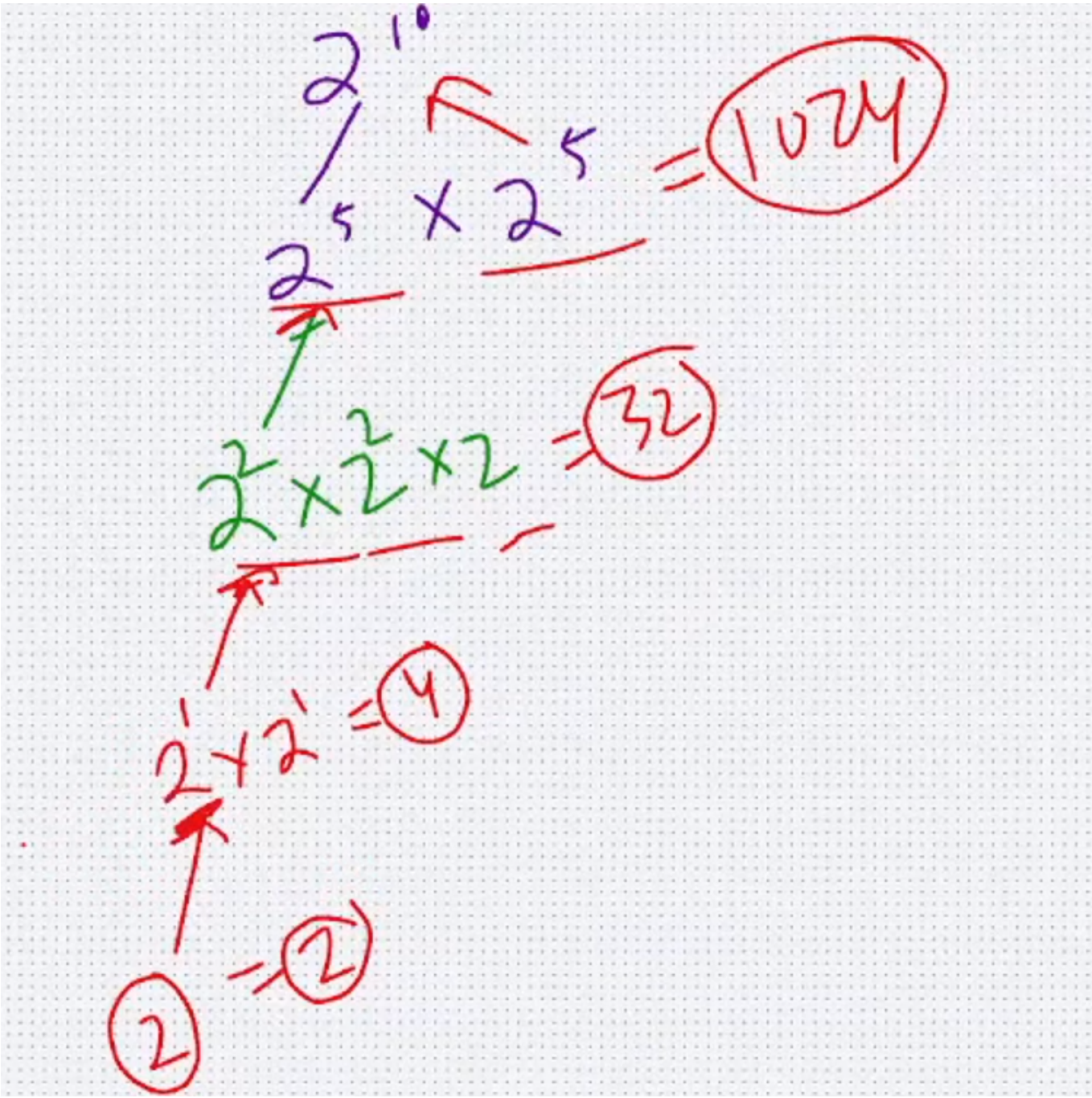
`import java.io.*;` class GFG {
`static long power(long A, long B) { `if (B == 0)` return 1;`

```
    long res = power(A, B / 2);

    if (B % 2 == 1)
        return res * res * A;
    else
        return res * res;
```

```
    }

    public static void main(String[] args) {
        System.out.println(power(2, 12));
    }
```

}





Iterative Implementation of Binary Exponentiation:

`public class Main {` public static long power(long a, long b) {
`long result = 1;` while (b > 0) {
`if ((b & 1) == 1) {` result = *a;*`
`}` *a* = a;
`b >>= 1;` }
` return result;` }

```
public static void main(String[] args) {
    System.out.println(power(2, 12));
```

```
}
```

}

Compute a large number modulo M:

(A B) mod M = ((A mod M) * (B mod M)) mod M

`import java.util.*;

`public class Main {` // Constant representing the modulo value
`static final int mod = 1000000007;

```
// Function to calculate the power of a number (a) raised to the power of b
modulo mod
public static long power(long a, long b) {
    long result = 1;
    while (b > 0) {
        // If the current bit of b is set, multiply the result by a
        if ((b & 1) == 1)
            result = (result * a) % mod;

        // Square the value of a and reduce it modulo mod
        a = (a * a) % mod;

        // Right shift b to move to the next bit
        b >>= 1;
    }
    return result;
}

public static void main(String[] args) {
    // Output the result of 2^42 modulo mod
    System.out.println(power(2, 42));
}
```

}