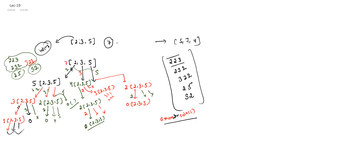
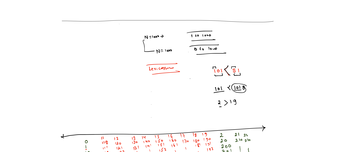


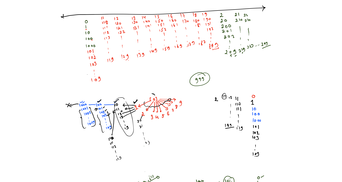
```
public static void Permutation(int[] coin, int amount, String ans) {
    if (amount == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < coin.length; i++) {
        if (amount >= coin[i]) {
            Permutation(coin, amount - coin[i], ans + coin[i]);
        }
    }
}
```

```
public static void Combination(int[] coin, int amount, List<Integer> ll, int idx) {
    if (amount == 0) {
        System.out.println(ll);
        return;
    }
    for (int i = idx; i < coin.length; i++) {
        if (amount >= coin[i]) {
            ll.add(coin[i]);
            Combination(coin, amount - coin[i], ll, i);
            ll.remove(ll.size() - 1);
        }
    }
}
```

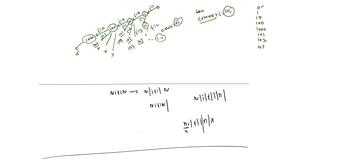
```
public static void Combination(int[] coin, int amount, List<Integer> ll, int idx, List<List<Integer>> ans) {
    if (amount == 0) {
        System.out.println(ll);
        ans.add(ll);
        return;
    }
    for (int i = idx; i < coin.length; i++) {
        if (amount >= coin[i]) {
            ll.add(coin[i]);
            Combination(coin, amount - coin[i], ll, i, ans);
            ll.remove(ll.size() - 1);
        }
    }
}
```

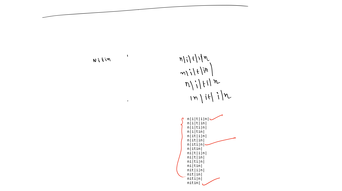


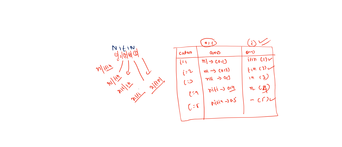
sieve of Eratosthenes



```
public static void Partition(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 1; i <= ques.length(); i++) {
        String s = ques.substring(0, i);
        Partition(ques.substring(i), ans + s + "|");
    }
}
```





sieve of Eratosthenes

```
public static int CountPrime(int n) {
    int[] prime = new int[n + 1];
    int count = 0;
    for (int i = 2; i < n; i++) {
        if (prime[i] == 0) {
            count++;
            for (int j = 2; j * i < prime.length; j++) {
                prime[j * i] = 1;
            }
        }
    }
    return count;
}
```