

Deepak and Gautam are having a discussion on a new type of number that they call Coding Blocks Number or CB Number. They use following criteria to define a CB Number.

- 0 and 1 are not a CB number.
- 2,3,5,7,11,13,17,19,23,29 are CB numbers.
- Any number not divisible by the numbers in point 2( Given above) are also CB numbers.

179

$n = 179$   
x

$n = 19$

Deepak said he loved CB numbers.Hearing it, Gautam throws a challenge to him.

Gautam will give Deepak a string of digits. Deepak's task is to find the number of CB numbers in the string.

CB number once detected should not be sub-string or super-string of any other CB number.

Ex- In 4991, both 499 and 991 are CB numbers but you can choose either 499 or 991, not both.

Further, the CB number formed can only be a sub-string of the string.

Ex - In 481, you can not take 41 as CB number because 41 is not a sub-string of 481.

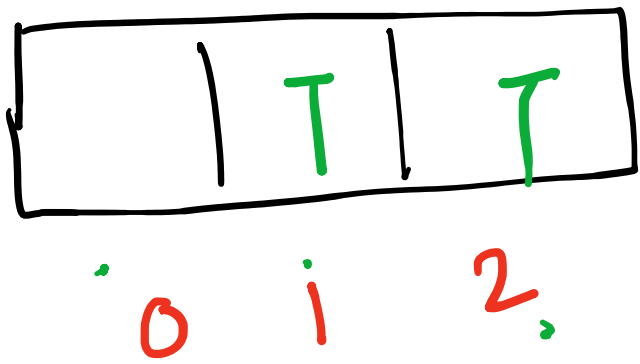
Handwritten calculations and diagrams illustrating the logic for finding CB numbers in a string. It shows various numbers being tested for divisibility by the set {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}. For example, 127 is shown as a prime number (not divisible by any of the factors), while 123 is shown as divisible by 3. A diagram shows a string "81615" with sub-strings "816", "8161", and "81615" being tested. "816" is marked as not a CB number (x), "8161" is marked as a CB number (checkmark), and "81615" is marked as not a CB number (x) because it contains "8161".

Handwritten diagrams and calculations. A diagram shows a string "01234" with a box highlighting "234". Below it, a box contains "T T" under "2 3". To the right, a calculation shows "127" is not divisible by 2 or 7. Another calculation shows "01234" is not divisible by 2 or 7. A box contains "2, 4" and "617".

```
public static void printallsubstring(String s) {
    int c = 0;
    boolean[] visited = new boolean[s.length()];
    for (int len = 1; len <= s.length(); len++) {
        for (int j = len; j <= s.length(); j++) {
            for (int i = j - len; i < j; i++) {
                long x = Long.parseLong(s.substring(i, j));
                if (isCB_Numbers(x) && !visited[i] && !visited[j-1]) {
                    c++;
                    for (int k = i; k < j; k++) {
                        visited[k] = true;
                    }
                }
            }
        }
    }
    System.out.println(c);
}

public static boolean isvisited(boolean[] visited, int si, int ei) {
    for (int i = si; i <= ei; i++) {
        if (visited[i] == true) {
            return false;
        }
    }
    return true;
}
```

127  
012



len=1  
j=1  
i=j-1=0

len=2  
j=2  
i=j-2=0

j=2  
i=j-1=1  
j=3  
i=j-2=1

j=3  
i=j-2=1  
len=3  
j=3  
i=j-3=0

0 2  
2 2