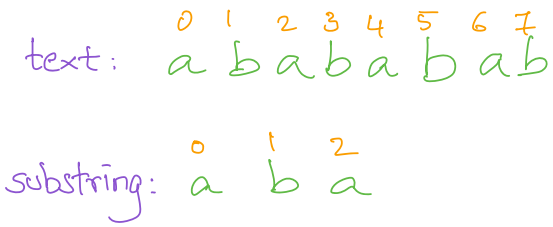
The code is printing the count of the substrings, Guess the output. *

```java
public class Main {
    public static void main(String[] args) {
        String text = "abababab";
        String substring = "aba";
        int count = 0;

        for (int i = 0; i <= text.length() - substring.length(); i++) {
            if (text.substring(i, i + substring.length()).equals(substring)) {
                count++;
            }
        }

        System.out.println(count);
    }
}
```

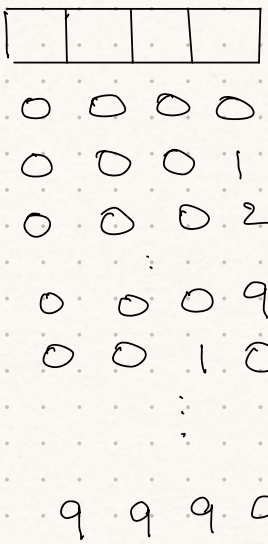


- 2
- 3
- 4
- 5

Bruteforce Algorithms

Recap:
- Array Indexing
- Substrings
- Subarrays
- Subsequences.

Bruteforce (search) algorithm: "Try all possible solutions (systematically)."



Systematically try all possible solutions

10,000 possible solutions!

Features of Bruteforce algorithms

- Blind search : No strategy involved. (naive)
- Guaranteed to find solution (if it exits)
- Often easiest to understand & implement
- Mostly very slow. (take a lot of time to run)

4 digits = 10^4 possible soln

5 digits = 10^5 " "

Pattern (Structure) of a Brute force algorithm :

- Generate all possible candidate solutions
- For each candidate solution,
  - check if it satisfies constraints of the problem

(opt) - pick the best solution.

Ex 1: Find all the divisors (factors) of a number n

$n = 16$ $\quad n < 10{,}000$

i	n % i
1	16 % 1 = 0 ✓
2	16 % 2 = 0 ✓
3	16 % 3 = 1
4	16 % 4 = 0 ✓
⋮	
16	16 % 16 = 0 ✓

Generating all possible solutions

1, 2, 4, 8, 16.

```
for i = 1 to n{
    if (n % i == 0){
        print(i)
    }
}
```

Ex 2: Find the lowest common multiple (l.c.m) of two numbers n & m.

e.g: $n = 6$

$m = 8$

multiples of 6 : 6, 12, 18, 24, 30, 36,

multiples of 8 : 8, 16, 24, 32, 40, 48...

i

Generating

max(n, m)

⋮

6×8 = 48 $n * m$

check if i is a multiple of both 6 & 8.

1. Generate all possible solutions
2. Check if it satisfies constraints (if it is a multiple of both n & m)

$n = 6$, $m = 8$

```
for (int i = Math.max(n, m) ; i <= n*m ; i++) {
    if((i%n == 0) && (i%m == 0)){
        S.O.P(i);
        break;
    }
}
```

i	$i \le 6*8$ (48)	$(i\%n == 0) \&\& (i\%m == 0)$
8	$8 \le 48$ ✓	$8\%6$ ✗
9	$9 \le 48$ ✓	$9\%6 == 0$ ✗ && $(9\%8 == 0)$
i	$i \le 48$	$i\%6$ & $i\%8$

e.g : $n = 2$, $m = 3$.

$i = \max(n,m)$, $i \le n*m$ ($2*3$, 6)

i
3
4
5
6

Count Such Pairs

	0	1	2	3	4
A	3	0	6	2	7

A.length = 5

k = 9

{3,6}, 3+6 = 9

{2,7} 2+7 = 9

Output: 2

Bruteforce:

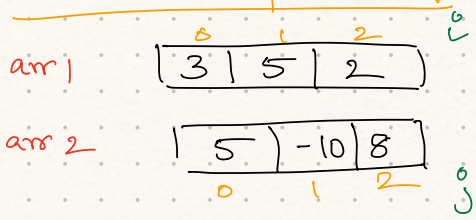
1. Generate all possible pairs
2. For each pair,
   check if its sum == k

```
int count = 0;
for(int i=0; i < A.length-1; i++) {
    for (int j = i+1 ; j< A.length; j++){
        // {A[i], A[j]} is the current pair
        if (A[i] + A[j] == k)
            count++;
    }
}
```

A[i], A[j]

i	j	Pairs	Sum	
0	1	3, 0	3	
0	2	3, 6	9	✓
0	3	3, 2	5	
0	4	3, 7	10	
1	2	0, 6	6	
1	3	0, 2	2	
1	4	0, 7	7	
2	3	6, 2	8	
2	4	6, 7	13	
3	4	2, 7	9	✓

Intersection of Arrays



"For every element in arr1"
- check if it appears in arr2.

```
for (i=0; i<n; i++) {
    x = arr1[i]
    //Check if x appears in arr2.
    for (j=0; j<n; j++) {
        if (x == arr2[j])        // if (arr1[i] == arr2[j]) {
            S.O.P(x);                 S.O.P (arr1[i]);
            break;                    break
    }                            }
}
```

i	j	j<n	x == arr2[j]	j++	i++
0	0		3 == 5	1	
	1		3 == -10	2	1
	2		3 == 8	3	
	3	3<3 ✗			
1					

x = ~~3~~ 5