Worst case

Competitive Programming: Core Skills

Artur Riazanov

SPbSU

Introduction

- In programming competitions time limits are pretty tight.

Introduction

- In programming competitions time limits are pretty tight.
- The whole point of most of the problems is to do the thing quickly.

Introduction

- In programming competitions time limits are pretty tight.
- The whole point of most of the problems is to do the thing quickly.
- But how to measure the quickness? How to predict how long a program will run? Let's find out!

Example

Substring

Given two strings s and t check if s is a substring of t.

Example

Substring

Given two strings s and t check if s is a substring of t.

Input: $s = \mathtt{abac}$; $t = \mathtt{abacabad}$

Output: Yes: abacabad

Example

Substring

Given two strings s and t check if s is a substring of t.

Input: $s = \mathtt{abac}$; $t = \mathtt{abacabad}$

Output: Yes: abacabad

Input: $s = \mathtt{cac}$; $t = \mathtt{abacabad}$

Output: No.

Example

Substring

Given two strings s and t check if s is a substring of t.

Input: $s = \mathrm{abac}$; $t = \mathrm{abacabad}$

Output: Yes: abacabad

Input: $s = \mathrm{cac}$; $t = \mathrm{abacabad}$

Output: No.

Input: $s = \mathrm{abab}$; $t = \mathrm{abacabab}$

Output: Yes: abacabab

Algorithm

Denote the length of s as n and the length of t as m. If $n > m$, then the answer is always no.
The easiest solution for this problem is to check for each substring of t of length n if it equals to s.

Examples

$s = \texttt{abab}$; $t = \texttt{abacabab}$;

Examples

$s =$ abab; $t =$ abacabab;

a b a b

a b a c a b a b

0 operations

Examples

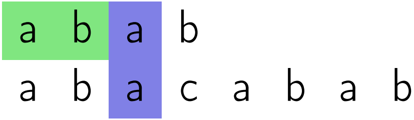
$s =$ abab; $t =$ abacabab;

a	b	a	b				
a	b	a	c	a	b	a	b

1 operation

Examples

$s = \mathtt{abab}$; $t = \mathtt{abacabab}$;



a b a b

a b a c a b a b

2 operations

Examples

$s =$ abab; $t =$ abacabab;

a	b	a	b				
a	b	a	c	a	b	a	b

3 operations

Examples

s = abab; t = abacabab;

a	b	a	b				
a	b	a	c	a	b	a	b

$4 = 4$

Examples

s = abab; t = abacabab;

	a	b	a	b			
a	b	a	c	a	b	a	b

$4 + 1 = 5$

Examples

$s = \texttt{abab}$; $t = \texttt{abacabab}$;

		a	b	a	b		
a	b	a	c	a	b	a	b

$4 + 1 + 2 = 7$

Examples

$s =$ abab; $t =$ abacabab;

			a	b	a	b	
a	b	a	c	a	b	a	b

$4 + 1 + 2 + 1 = 8$

Examples

$s = \text{abab}$; $t = \text{abacabab}$;

				a	b	a	b
a	b	a	c	a	b	a	b

$$4 + 1 + 2 + 1 + 4 = 12$$

Examples

$s = \texttt{abac}$; $t = \texttt{abacabad}$;

Examples

$s = \text{abac}$; $t = \text{abacabad}$;

a b a c

a b a c a b a d

$4 = 4$

We instantly got the match!

Variation

- Number of the operations can vary.

Variation

- Number of the operations can vary.
- If the program works quickly on the sample tests or even on some custom tests, that doesn't mean it'll always work this way.

Variation

- Number of the operations can vary.
- If the program works quickly on the sample tests or even on some custom tests, that doesn't mean it'll always work this way.
- Usually your program is expected to work on the worst possible test.

Variation

- Number of the operations can vary.
- If the program works quickly on the sample tests or even on some custom tests, that doesn't mean it'll always work this way.
- Usually your program is expected to work on the worst possible test.
- In the worst possible test for the substring problems
  - The answer is "No" hence the algorithm will check every position.
  - At every position the algorithm will compare symbols until the very end.

Examples

$s = \mathrm{aaab}$; $t = \mathrm{aaaaaaaa}$;

Examples

$s = \text{aaab}$; $t = \text{aaaaaaaa}$;

a	a	a	b				
a	a	a	a	a	a	a	a

$4 = 4$

Examples

$s =$ aaab; $t =$ aaaaaaaa;

```
  a a a b
a a a a a a a a
```

$4 + 4 = 8$

Examples

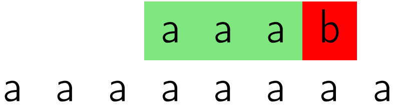
$s = \text{aaab}$; $t = \text{aaaaaaaa}$;

		a	a	a	b		
a	a	a	a	a	a	a	a

$4 + 4 + 4 = 12$

Examples

$s = \text{aaab}$; $t = \text{aaaaaaaa}$;



a a a b

a a a a a a a a

$4 + 4 + 4 + 4 = 16$

Examples

$s = \text{aaab};\ t = \text{aaaaaaaa};$

a a a b

a a a a a a a a

$4 + 4 + 4 + 4 + 4 = 20$

Conclusion

- Worst possible test is not always just biggest possible test.

Conclusion

- Worst possible test is not always just biggest possible test.
- It could be hard to explicitly construct it.

Conclusion

- Worst possible test is not always just biggest possible test.
- It could be hard to explicitly construct it.
- We have to develop a technique to estimate the number of operations algorithm does on the worst case which doesn't require explicit constuction.