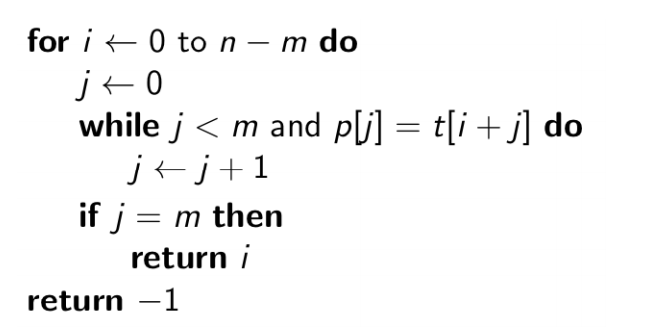
Brute force string matching



Q. Trace the brute force string matching algorithm on the following input: The path p is ‘needle’ and the text t is ‘there\_need\_not\_be\_any’. How many comparisons (successful and unsuccessful are made)?

Solution:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | h | e | r | e | \_ | n | e | e | d | \_ | n | o | t | \_ | b | e | \_ | a | n | y |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | e | e | d | l | e |

Initially we compare n and t, that’s unsuccessful match so we would increment value of i.

So we compare n and t -> NOT A MATCH (1st comparison)

Increment i

So we compare n and h -> NOT A MATCH – 2

Increment i

We would then compare n and e -> NOT A MATCH – 3

Increment i

We would then compare n and r -> NOT A MATCH – 4

Increment i

We would then compare n and e -> NOT A MATCH – 5

Increment i

We would then compare n and \_ -> NOT A MATCH – 6

Increment i

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | h | e | r | e | \_ | n | e | e | d | \_ | n | o | t | \_ | b | e | \_ | a | n | y |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | e | e | d | l | e |

We would then compare n and n -> MATCH -7

Increment j and i+j so that P[j] and T[i+j] would be incremented so the position in the text and pattern would be incremented subsequently.

We would then compare e and e -> MATCH - 8

Increment j and i+j

We would then compare e and e -> MATCH – 9

Increment j and i+j

We would then compare d and d -> MATCH - 10

Increment j and i+j

We would then compare l and \_ -> NOT A MATCH – 11

So then we would increment i so that it would point to e after n and that’s from where the comparison would begin.

We would then compare n and e -> NOT A MATCH - 12

Increment i

We would then compare n and e -> NOT A MATCH - 13

Increment i

We would then compare n and d -> NOT A MATCH - 14

Increment i

We would then compare n and \_ -> NOT A MATCH - 15

Increment i

We would then compare n and n -> MATCH - 16

So we would then increment j and i+j.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | h | e | r | e | \_ | n | e | e | d | \_ | n | o | t | \_ | b | e | \_ | a | n | Y |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | e | e | d | l | e |

We would then compare o and e -> NOT A MATCH -17

Increment i.

We would then compare n and o -> NOT A MATCH – 18

Increment i.

We would then compare n and t -> NOT A MATCH - 19

Increment i.

We would then compare n and \_ -> NOT A MATCH – 20

Increment i.

We would then compare n and b -> NOT A MATCH – 21

Increment i.

We would increment i from 0 to n-m, so the value n = 21 and m = 6 so n-m = 21-6 = 15.Only until index 15 do we increment i and only until that index in the text do we make any comparisons.

**So a total of 21 comparisons would be made.**

**Q. Assume that we have a text consisting of one million zeroes. For each of these patterns, determine how many character comparisons the brute force string matching algorithm would make:**

**(a) 010001**

**Soln : We would compare from left to right, so Initially 0 in the pattern would match with 0 in the text, but the next value in the pattern which is 1 would not match with the next 0 in the text. So for each 0 in the text, each time we are making 2 comparisons.**

**n-m would be 1,000,000 – 5 = 9,99,995 (i would be incremented from 0 to n-m).So for each in 1 mill zeroes we make two comparisons, so that’s 2\*9,99,995 which will be equal to 1,999,990 comparisons.**

**Algorithm in action:**

T[i+j],i=0,j=0

**T(x): 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (1 million zeros )**

0 1 2 3 4 <- index i . (now i=0)

P[j],j=0

**P(x): 0 1 0 0 0 1**

0 1 2 3 4 <- index j . (now j=0)

**Match between 0 and 0 so increment j. i is still at index 0.**

**i+j would be at position 1 in the text and as j =1.i is still 0. Next we do comparison between 0 and 1 which is false. So we would increment the value of i to be 1.**

T[i+j] i=0,j=1

**T(x): 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (1 million zeros)**

0 1 2 3 4 <- index i . (now i=0)

P[j],j=1

**P(x): 0 1 0 0 0 1**

0 1 2 3 4 <- index j . (now j=0)

**Not a match or it’s a mismatch. So we would increment i so now i=1 and reset j as j=0.**

**So every 0 in the text(with index i), we would do 2 comparisons and this done until n-m index in the text, which is equal to 1million – 5 = 9,99,995 .**

**So for 9,99,995 characters this would be equal to 9,99,995\*2 = 1,999,990.**

**(b) 000101**

**First 3 successful comparisons and then one unsuccessful comparison (TOTAL OF 4 COMPARISONS) this repeated for n-m zeroes in the text.**

**So we would have 1mil – 5 = 9,99,995\*4 = 3,999,980 comparisons.**

**(c)011101**

**COMPARISON DONE FROM LEFT TO RIGHT.**

**First successful comparison(between 0 in the pattern and 0 in the text and then unsuccessful between 1 in the pattern and 0 in the text. )**

**This is repeated for n-m characters in the text, which is 1million – 5 = 9,99,995 characters.**

**So a total of 9,99,995\*2 = 1,999,990 comparisons are done.**

**Q.** Give an example of a text of length n and a pattern of length m, which together constitute a worst-case scenario for the brute-force string matching algorithm. How many character comparisons, as a function of n and m, will be made for the worst-case example? What is the value of m (the length of the pattern) that maximises this function? i.e. What is the worst case pattern length?

Solution:

**The worst case happens when we would have a text of length of n consisting of a character c which is repeated n times, together with a pattern of length m, consisting of m-1 occurrences of c followed by a single character that would be different from c.**

**In this case the loop would be traversed n-m+1 time, and each time m character comparisons are made before failure would be detected.**

**Altogether we would have**

**(n-m+1)m = (n+1)m-m^2 comparisons.**

**As a function of m, this has its maximal value of**

**d/dm((n+1)m-m^2) = n+1-2m**

**Maximal value is when d/dm is equal to zero or n+1-2m is equal to zero.**

**We have n+1-2m = 0 or n+1 = 2m or m is roughly equal to n/2 or the length of the pattern is roughly equal to half the length of the text.**