Exact Matching #2

Today we will discuss in detail the exact ~~pattern~~ string matching problem we presented in the last lecture:

> Exact String Matching Problem: Given a (long) string T and a shorter string P, find all occurrences of P in T. Occurrences of P can overlap.

Applications:

1. Mapping second-generation sequencing short reads (~150 bp) to reference genome (~3Bbp)
2. Find word occurrences in text documents. Ctrl-S in emacs.
3. Keyword search in books, webpages. ~~Construct an inverted index~~

Example

T = ABCABCDABABCDABDABDE

P = ABCDABD

We will discuss a few algorithms to solve this problem. In their analysis we will treat 'character comparison' as the primitive operation.

Naive algorithm:

```
for i = 1, ..., |T|
    j = 1
    while j ≤ |P| and T[i+j-1] == P[j]:
        j += 1
    if j == |P|+1: report occurrence at position i
```

How many character comparisons: $|T||P| \Rightarrow O(|T||P|)$

Q: what is the worst case?

To improve this we should use information in previous loops when comparing P to $T[k\ldots]$. $\quad j = 1, \ldots, k-i$

What can we infer about $T[k\ldots]$ from previous loops:

ABC ABC DAB ...
ABC DAB D
ABC DABC
ABC DAB

The similarities of substrings in P allow us to make inferences about unnecessary comparisons.

First approach is to preprocess P to find similarities

Review

S: ABCDABD

Prefix of S: substring of S starting at position 1:

A, AB, ABC, ABCDAB

Suffix of S: substring of S ending at position |S|:

D, BD, ABD, BCDABD

Z-algorithm (Gusfield Ch. 1): $O(|T| + |P|)$

Fundamental Preprocessing

Def. Given string S and position $i > 1$, $z_i(s)$ is the length of the longest substring of S starting at position i that matches a prefix of S.

$S[i, \ldots, i+z_i] = S[1, \ldots, z_i]$

$|\alpha| = z_i$

ABCDABD

$z_5 = 2$

How do we use z_i to solve the exact string matching problem?

Linear time alg. to compute Z-scores gives us linear time alg. for exact matching

S = P$T

Positions in T that $z_i(S) = |P|$ & $i > |P|+1$ => P occurs at $T[i - |P| - 1]$

$P = S[1, \ldots, |P|] = S[i, \ldots, i+|P|] = T[i-|P|-1, \ldots, i-1]$

compute Z-values gives us

Step 1: z_2

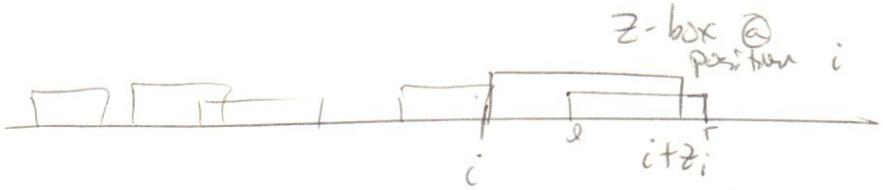
→ Directly compare $S[2,\dots]$ and $S[1,\dots]$ until first mismatching position

i.e., compute z_2 directly

Step $k = 2, \dots,$

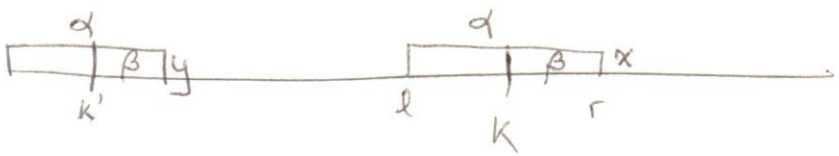
→ Assume $z_2, \dots, z_{k-1}$ are computed

Def: Z-box at position i is interval $i, \dots, i+z_i$



Def: r: end of right-most Z-box found through $2, \dots, k-1$

l: start of right-most Z-box found through $2, \dots, k-1$

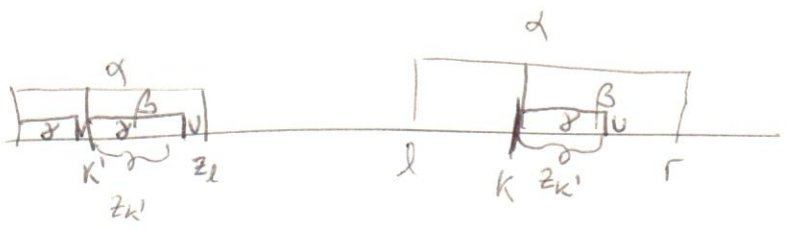


$x \neq y$

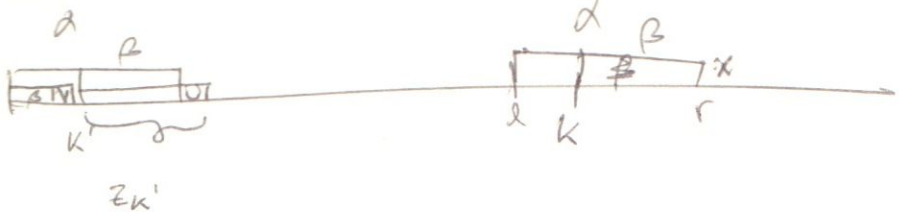
Case 2: $k \leq r$

Case 2a: $z_{k'} < |\beta| \Rightarrow z_k = z_{k'}$

$u \neq v$

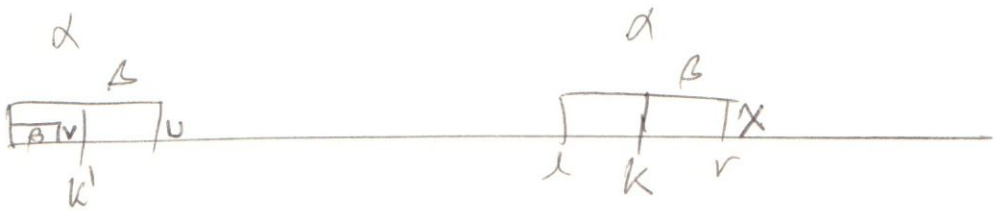


Case 2b:



$U = V$
$X \neq U$ #
$\Rightarrow X \neq V$

Case 2c: $z_{k'} = |\beta| \Rightarrow z_k = |\beta| + q$



$U \neq V$
$X \neq U$
$X \stackrel{?}{=} V$

compare $S[r+1, \ldots]$ to $S[\beta+1, \ldots]$ until first mismatch (suppose q matches)

$r_k = r_{k-1} + q$
$l = k$

Case 1 $k > r$

compare $S[k, \ldots]$ to $S[1, \ldots]$ until first mismatch (suppose $q > 0$ matches)

$r_k = k + q$
$l = k$

Obs: $r_k \geq r_{k-1}$ for all k!!

Analysis:

Correctness: by construction

Runtime: # matching + # mismatching
$|S|$ $|S|$

$O(|S|)$

Ste Next class:

KMP

Boyer-Moore

Also $O(|P|+|T|)$. B-M is better in practice, KMP will help you in project 1.

Step

Compression with Z-scores

De if $z_i = 0$ send $S[i]$

o.w. send z_i instead of $S[i, \ldots i+z_i]$

Def

Case

Case