CSE 566 Spring 2023

IInstructor: Mingfu Shao

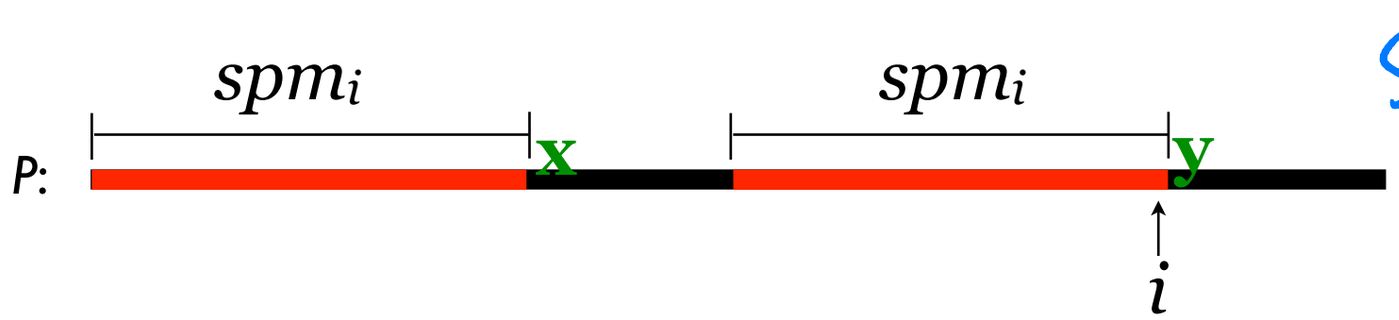
KMP and Suffix-Tree

((Slides copied edited from these by Dr. Carl Kingsford))

Knuth-Morris-Pratt

Knuth-Morris-Pratt (KMP)

Def. $spm_i(P)$ = the length of the longest substring of P that *ends* at $i > 1$ and matches a prefix of P **and** such that $P[i+1] \neq P[spm_i + 1]$. ("*spm*" stands for suffix, prefix, mismatch.)



$spm_i < i$

- P = "atcabatcab": spm_4 = 1, spm_9 = 1
- P = "alfalfa": spm_5 = 0, spm_7 =
- P = "photophosphorescent": spm_8 = 3, spm_{10} = 0

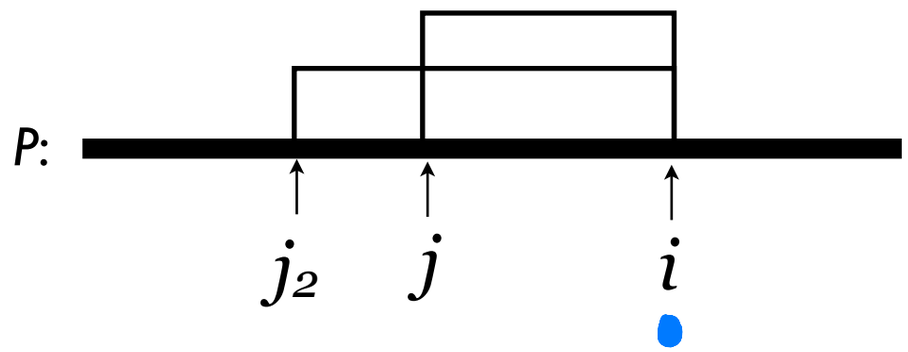
8

How should we define if $i = |P|$?

1. X
2. ✓
3. does not matter

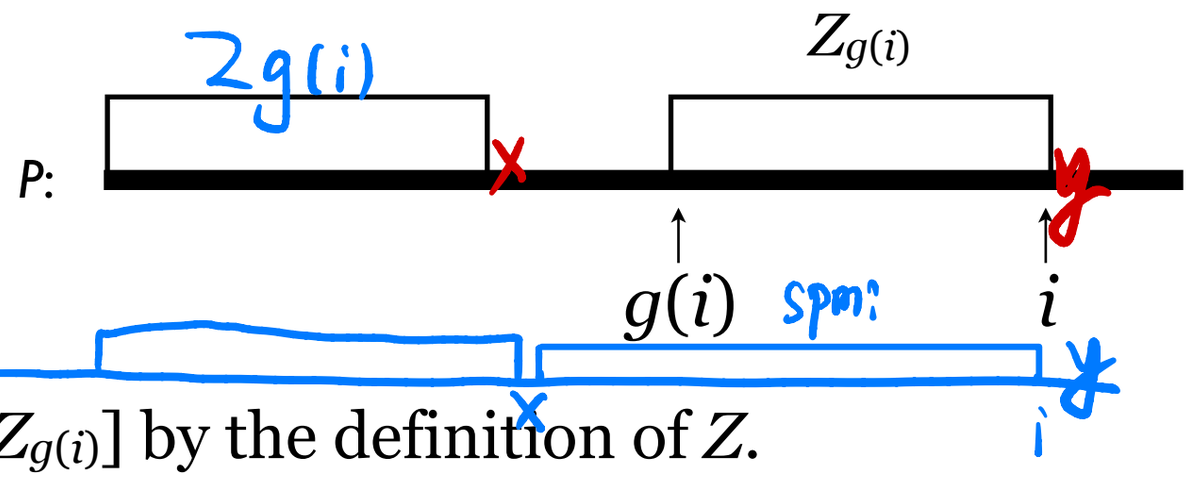
Computing spm from Z

$g(i)$ = min {j : right-end of the Z-box starting at j is i}, or 0 if no such Z-box.



Thm. $spm_i = Z_{g(i)}$ if $g(i) > 0$ otherwise 0

Proof.



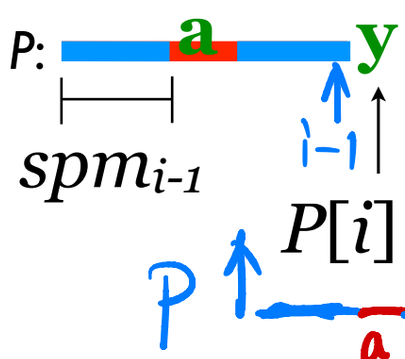
$P[g(i)..i] = P[1..Z_{g(i)}]$ by the definition of Z.

Also, $P(i+1) \neq P[Z_{g(i)}+1]$, otherwise $Z_{g(i)}$ would be bigger.

So, $spm_i \geq Z_{g(i)}$. But it can't be longer, because otherwise $g(i)$ would be smaller.

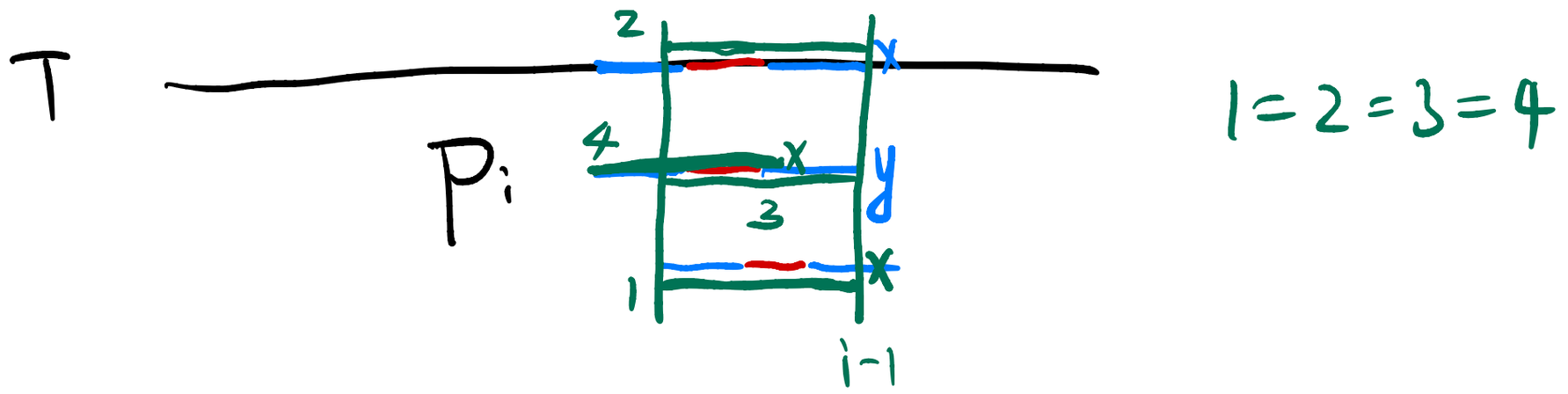
Key Idea of KMP

- Upon mismatch:
  - naive algorithm: shift P to the right by 1
  - KMP: shift P to the right by more than 1

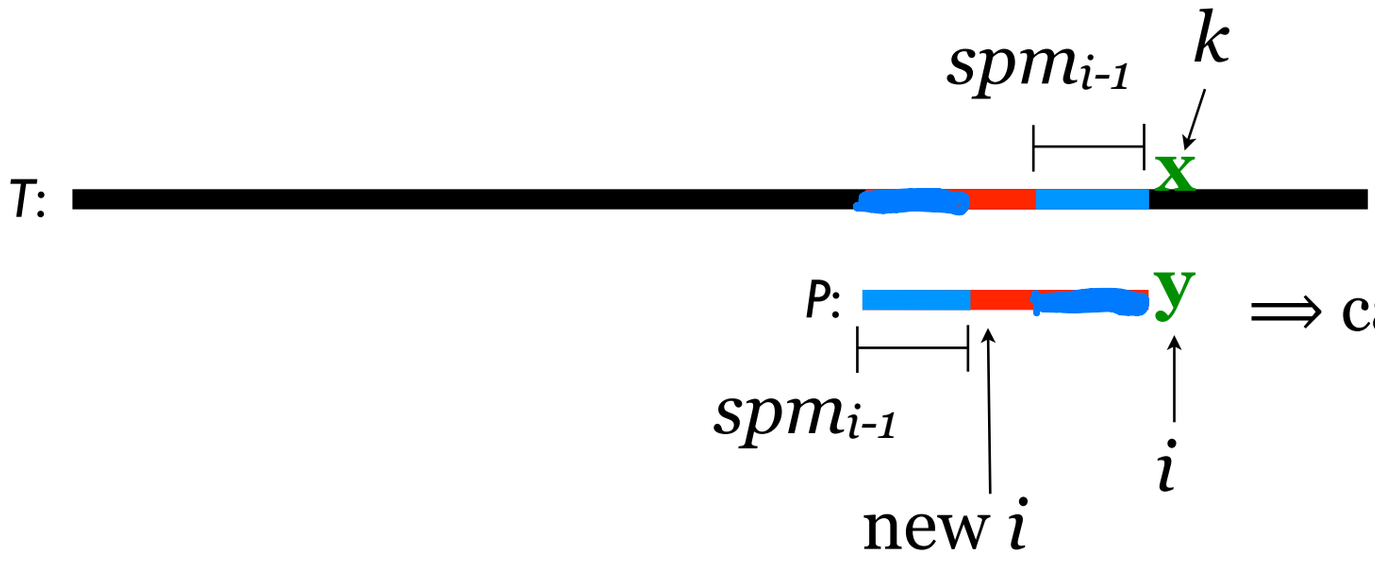


$\Longrightarrow$ can shift by: i - spm_{i-1}
AND compare the next letter

- Why it's safe, i.e., why shifting less always fails?



KMP



$\Longrightarrow$ can set new i to $spm_{i-1}+1$

K never decrease

```
k = i = 1     // ptrs into T and P, respectively
while |P| - i ≤ |T| - k:
    while P[i] = T[k] and i ≤ |P|: k++ and i++; // compare P/T
    if i = |P| + 1: print "occur at", k - |P|   // found
    if i = 1: k++; // mismatch at start, increase k (shift by 1)
    else: i = spm_{i-1} + 1; // "shift more", even if i=|P|+1
```

$|T| \geq |P| + k - i$

(i gets decreased)

$1 + spm_{i-1} \leq i - 1 + 1 = i$

KMP Running Time

Above procedure runs in O(|*T*|) time :

- #(iterations of outer while-loop) ≤ |T| since P is shifted by ≥ 1 each time
- total #mismatches ≤ #(iterations of outer while-loop) ≤ |T|
- total #matches ≤ |T|, since k is increased after each match

Comparing Z and KMP

- building Z array / spm: O(|P|+|T|) vs O(|P|)
- main procedure: O(|T|) vs O(|T|)

Z KMP

Z KMP

O(|P|)+O(T)
=O(|P|+|T|)

Suffix Tree

Preprocessing Strings

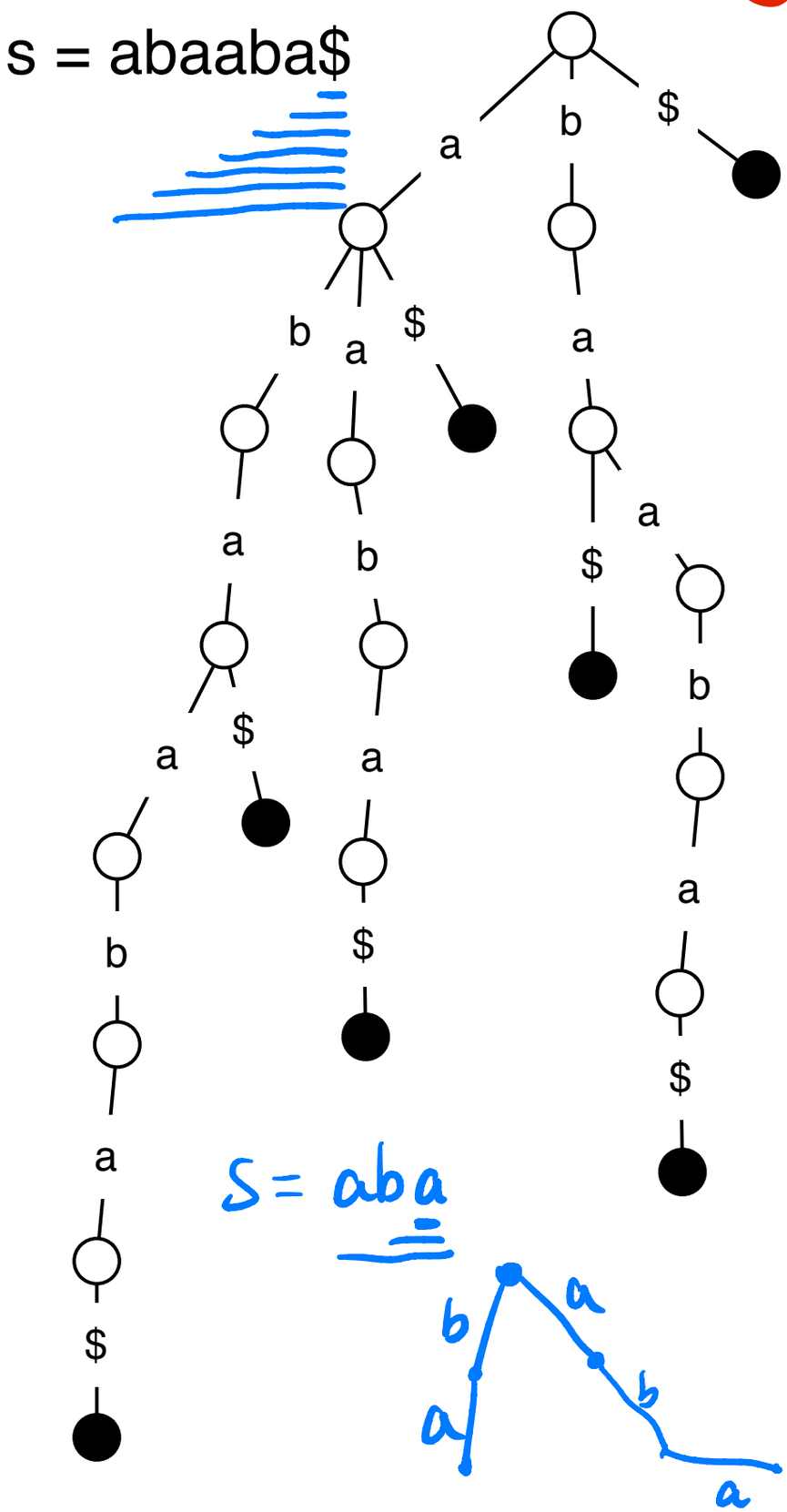
- Methods for preprocessing strings into data structures that make many questions (like searching) easy to answer:
  - Suffix Tries
  - Suffix Trees
  - Suffix Arrays
  - Borrows-Wheeler transform (BWT)

- Typical setting: A long, known, and fixed text string (like a genome) and many unknown, changing query strings.
  - To preprocess the **text** string once

- Data structures will be useful in other settings as well.

Suffix-trie(S)

Suffix Trie: Definition



S = aba

1. A tree structure representing string s.
2. Edges are labeled with letters from the alphabet Σ say {A,C,G,T,$}
3. Out-edges of the same node are labeled differently.
4. Every path from the root to a leaf represents a suffix of s.
5. Every suffix of s is represented by some path from the root to a leaf.

How many leaves will there be? |S|
Why adding $ to the end?