Lecture 15

COL 351: Analysis and Design of Algorithms

Lecture 15

String Matching Problem

Given: String S = [s_1,…, s_n] and a pattern P = [p_1, …, p_k], represented as arrays of size n, k. (Here $k < n$).

Find: Does there exists a sub-string of S that is identical to P.

Examples:

S = "cuckoo hashing is efficient"

P = "hash"

Yes

S = "cuckoo hashing is efficient"

P = "hash-table"

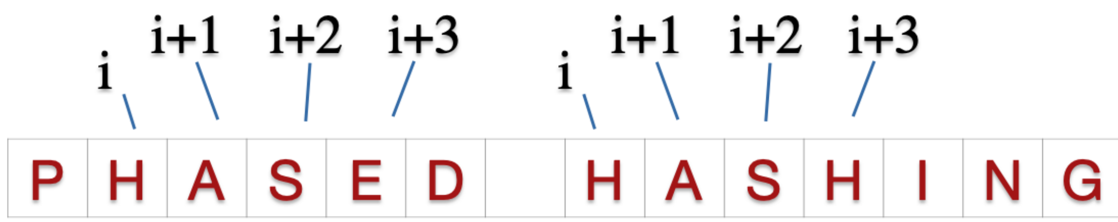
No

String Matching Problem

Given: String S = [s_1,…, s_n] and a pattern P = [p_1, …, p_k], represented as arrays of size n, k. (Here $k < n$).

Find: Does there exists a sub-string of S that is identical to P.

```
For i = 1 to n:
    Flag = True
    For j = 0 to k-1:
        If S[i + j] ≠ P[1+ j] then Flag= False
    If (Flag) Return True
Return False
```

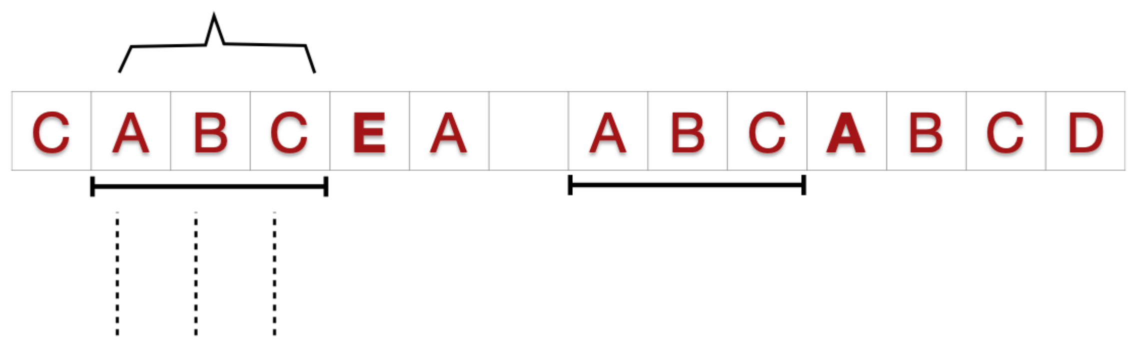


$O(nk)$ time algorithm

Special Scenario: All characters in " pattern P " are different!

```
i, j ← 1;
While (i ≤ n):
    If S[i] = P[j]):
        If (j = k): Return True
        Increment i and j by 1;
    Else :
        j ← 1;
```

The substring matched with prefix of P contains only one copy of P[1]=A as no characters are repeated in P



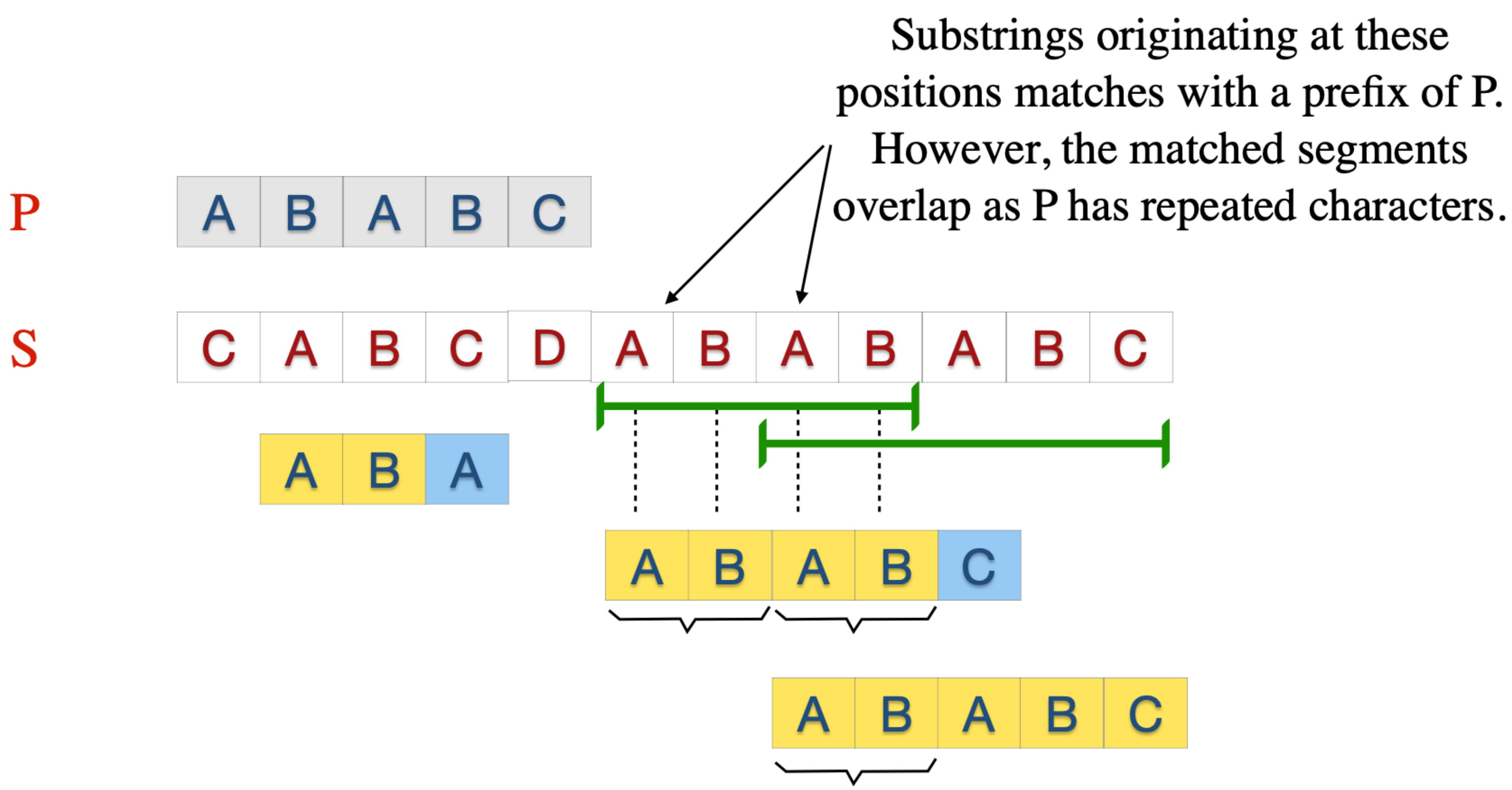
If (S[i] = P[j]): $j \leftarrow 2$;

Increment i by 1;

Return False;

O(n) time algorithm for special scenario

An Example where P has repeated characters



Key Idea to obtain Linear time algorithm - Pattern preprocessing.

Sub-Problem

Given: String a pattern $P = [p_1, \ldots, p_k]$, represented as arrays of size k.

Find: A Table of size k satisfying

Table[i] := The length of longest non-trivial prefix of P[1, i] that is also a suffix of P[1, i]

Examples:

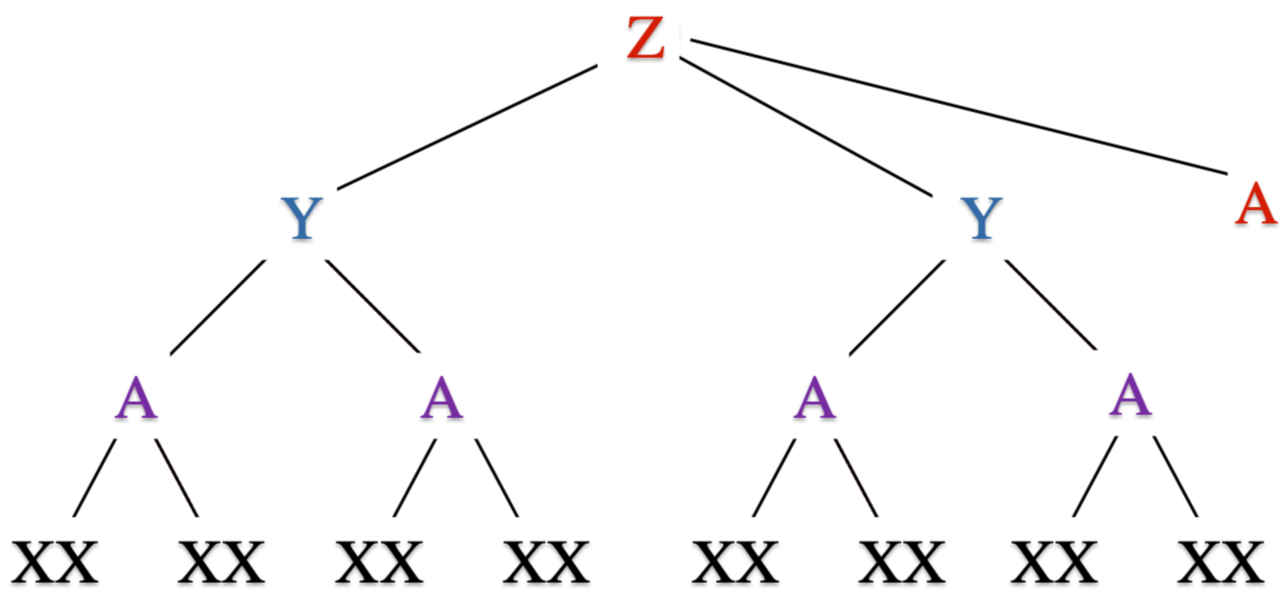
i	1	2	3	4	5	6
P[i]	A	B	C	A	B	B
Table[i]	0	0	0	1	2	0

i	1	2	3	4	5	6	7	8	9
P[i]	A	A	B	A	A	B	A	A	A
Table[i]	0	1	0	1	2	3	4	5	2

Home-work

i	1	2	3	4	5	6	7	8	9
P[i]	A	B	C	A	B	C	A	B	C
Table[i]									

We will study this example to understand the intuition for an O(k) time algorithm to solve our sub-problem.



i	1	2	3	4	5	6	7	8	9	10	11	12												
P[i]	X	X	A	X	X	Y	X	X	A	X	X	Z	X	X	A	X	X	Y	X	X	A	X	X	A
Table[i]	0	1	0	1	2	0	1	2	3	4	5	0	1	2	3	4	5	6	7	8	9	10	11	

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	…									22	i	i+1
P[i]	X	X	A	X	X	Y	X	X	A	X	X	Z	X	X	A	X	X	Y	X	X	A	X	X	A

Table[i]	0	1	0	1	**2**	0	1	2	3	4	**5**	0	1	2	3	4	5	6	7	8	9	10	11

- Suppose we have computed Table values upto an index $i = 23$, and want to compute Table[$i + 1$].
- Observe Table[23] =11. Thus, 11 is length of longest identical suffix-prefix of P[1, 23].
- Now, Z = P[11+1] $\neq$ P[23+1] = A, therefore, Table[23+1] cannot be 12.
  We compute length of longest identical suffix-prefix of P[1, 23] smaller than 11. This is just P[11] = 5.
- Again, Y=P[5+1]$\neq$P[23+1]=A.
  Therefore, we compute length of longest identical suffix-prefix of P[1, 23] smaller than 5. This is just P[5] = 2.
- Compare P[2+1] and P[23+1]. Both are A. Thus, Table[23+1] is 2+1=3.

HomeWork

Complete the entries of Table below by applying the algorithm stated in previous slide, verify the answers manually.

i	1	2	3	4	5	6	7	8	9	10

P[i]	A	A	A	B	A	A	A	A	A	B
Table[i]										