NANYANG TECHNOLOGICAL UNIVERSITY

SC2001/ CX2101: Algorithm Design and Analysis

Week 12

Huang Shell Ying

Please log into Teams as we would like to conduct 2 polls in this lecture.

Pseudo-polynomial time

- (Slide 77): An algorithm is polynomial time if it is a polynomial function of the size of the input.
  - All sorting algorithms we covered: E.g MergeSort, Complexity: O(nlgn). Size of input: n+1
  - Graph algorithms: E.g. Dijkstra's, complexity: $O(n^2)$, Size of input: $1+n+n^2$
- (Slide 77): An algorithm is pseudo-polynomial time if it is a polynomial function of the value of the input.
  - DP for Knapsack: complexity: O(nC). Size of input: 2+2n. Complexity function is a function of both n and C, where C is the value of the 2^{nd} input number

Pseudo-polynomial time

- Poll 1

  DP for chain matrix multiplication: complexity: $O(n^3)$. Is the algorithm polynomial or pseudo-polynomial?

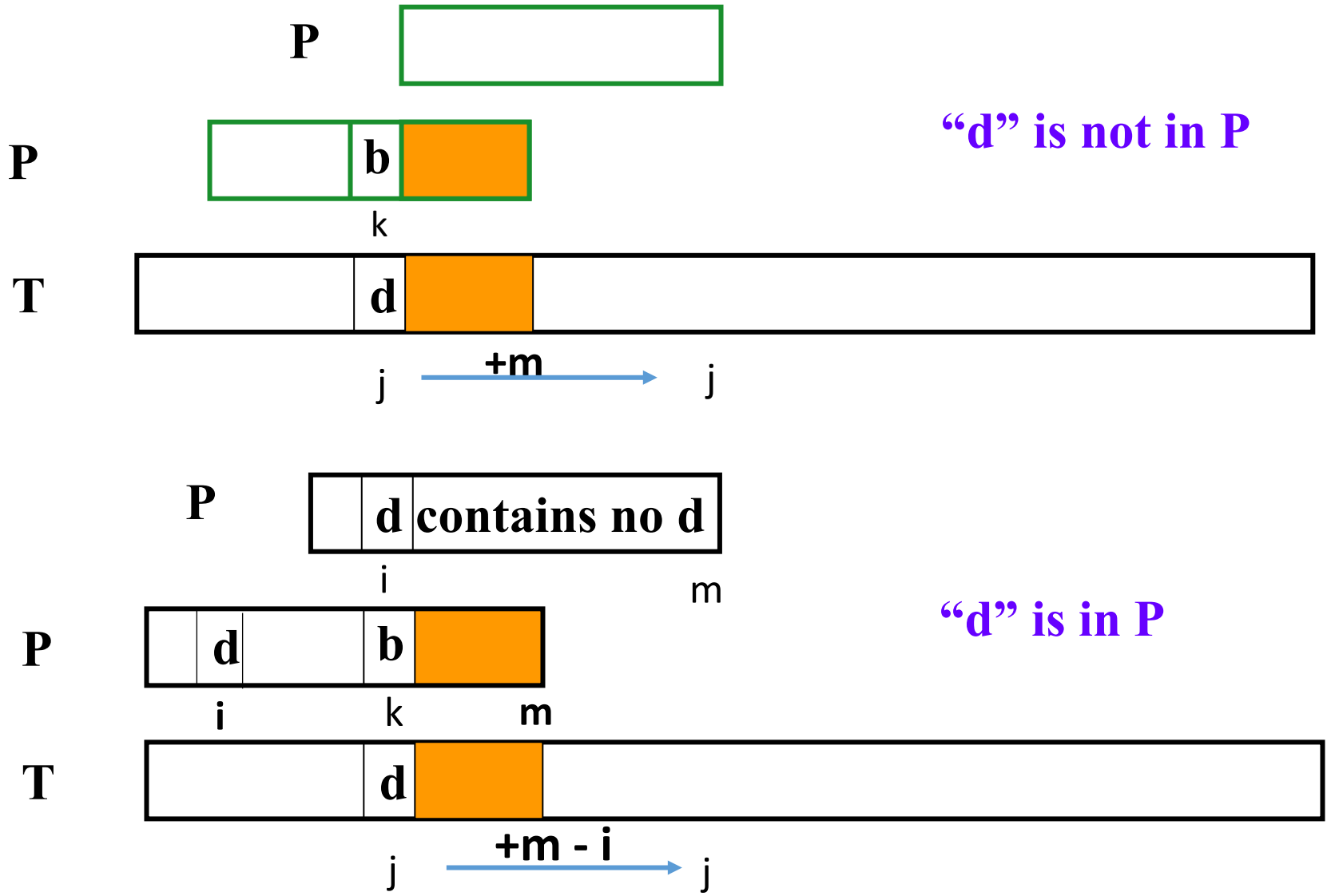
- Poll 2

  DP for Fibonacci function F(n): complexity: O(n). Is the algorithm polynomial or pseudo-polynomial?

Recap on Boyer-Moore Algorithm

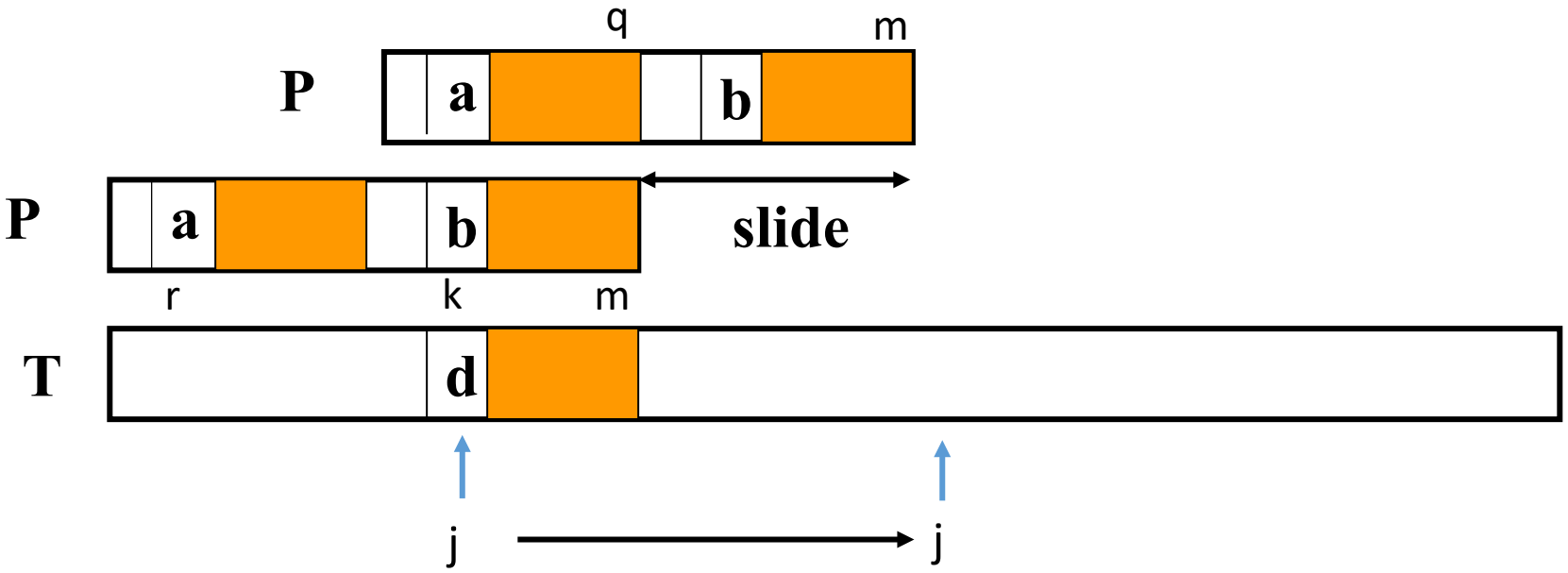
- It is a very efficient algorithm for string searching
- The text being scanned is T with *n* characters
- The pattern we are looking for is P with *m* characters
- Process the text T[1..*n*] from left to right
- Scan the pattern P[1..*m*] **from right to left**
- Preprocessing to generate two tables based on which to slide the pattern as much as possible after a mismatch
- It performs even better with long patterns

Boyer-Moore: Preprocessing to compute charJump



Boyer-Moore: Preporocessing to compute matchJump

Case 1: The matching suffix occurs earlier in the pattern, but preceded by a different character

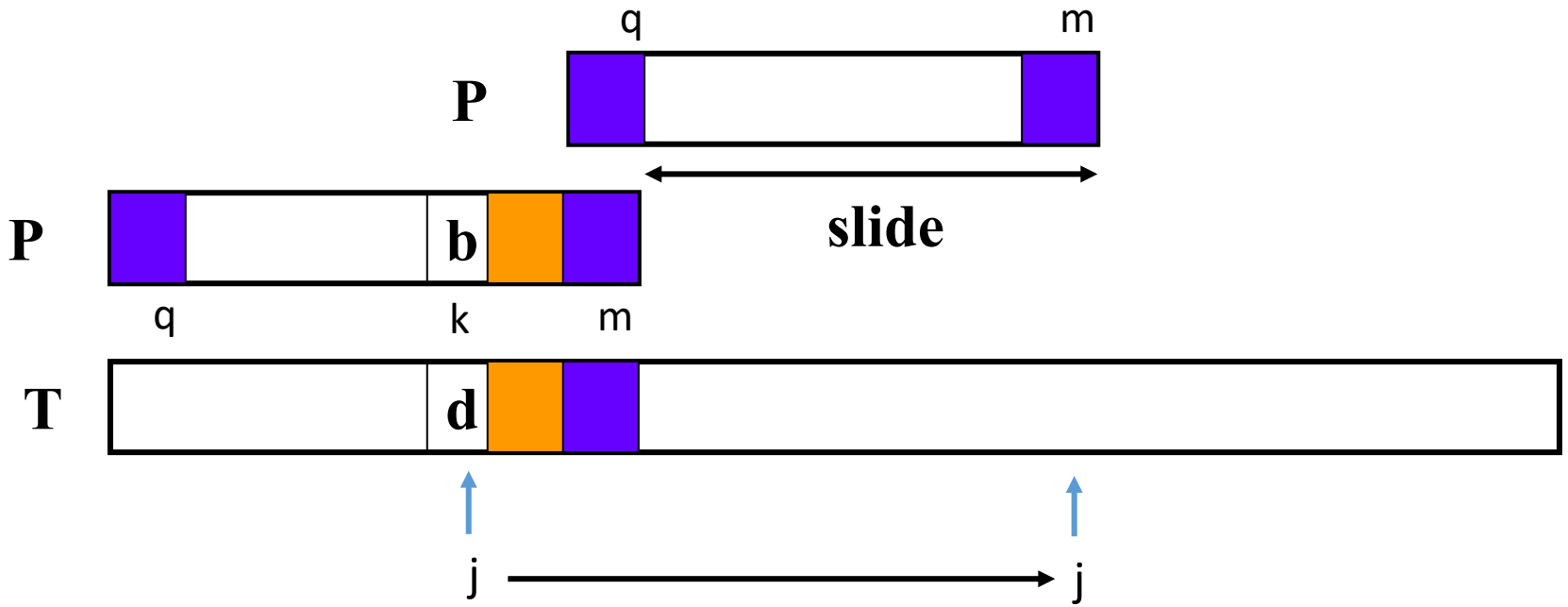


matchJump[k] = m – k + slide[k]

slide[k] = m - q (P[r] ≠ P[k])

i.e. matchJump[k] = no. of characters matched + slide[k]

Case 2: Only part of the matching suffix occurs at the beginning of the pattern (a prefix).

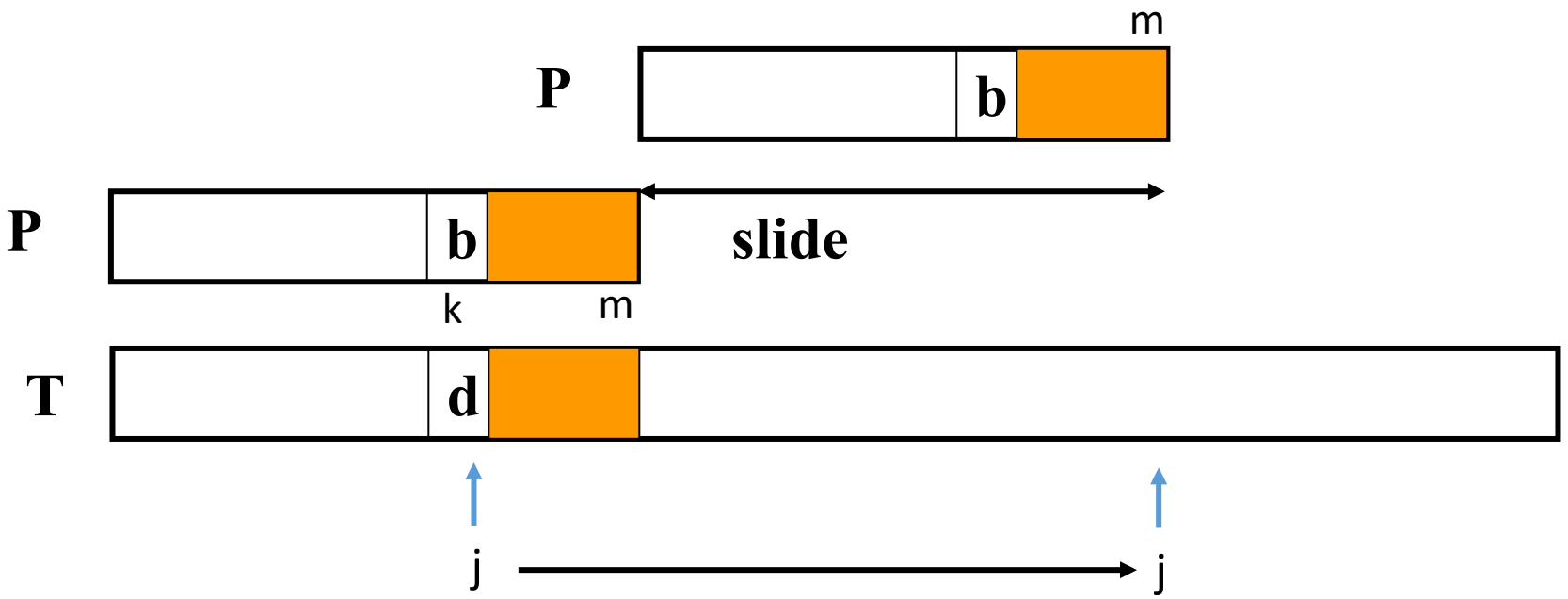


matchJump[k] = m – k + slide[k]

slide[k] = m – q

i.e. matchJump[k] = no. of characters matched + slide[k]

Case 3: There is no other occurrence of the matching suffix in the pattern. (Case 1 and Case 2 do not happen)



matchJump[k] = m – k + slide[k]

slide[k] = m (q = 0)

i.e. matchJump[k] = no. of characters matched + slide[k]

Note: Slide[m] = 1

Example 1 of finding matchJump arrays

List the values in the *charJump* and *matchJump* arrays for the Boyer-Moore Algorithm for the following pattern:

A A B A A C A A B A B A

```
void computeJumps(char [] P, int m,
                  int alpha, int [] charJump)
{   char ch;  int k;
      for (ch = 0; ch < alpha; ch++)
            charJump[ch] = m;
      for (k = 1; k <= m; k++)
                  charJump[ P[k] ] = m - k;
}
```

charJump[A] = 0
charJump[B] = 1
charJump[C] = 6
charJump[X] = 12
X is any character in the character set not A, B, C

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

											1

Note: Slide[m] = 1

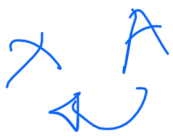
Matched = 0

Slide[12] = 1

not count as next one is B

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

										5	1



Matched = 1

Slide[11] = 12-8=4

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

									13	5	1

Matched = 2

Slide[10] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

								5	13	5	1

Matched = 3

Slide[9] = 12-10=2

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12
							15	5	13	5	1

not found

Matched = 4

Slide[8] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12
						16	15	5	13	5	1

nf

Matched = 5

Slide[7] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

					17	16	15	5	13	5	1

Matched = 6

Slide[6] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

				18	17	16	15	5	13	5	1

Matched = 7

Slide[5] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

			19	18	17	16	15	5	13	5	1

Matched = 8

Slide[4] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

		20	19	18	17	16	15	5	13	5	1

Matched = 9

Slide[3] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

	21	20	19	18	17	16	15	5	13	5	1

Matched = 10

Slide[2] = 12-1=11

A	A	B	A	A	C	A	A	B	A	B	A
1	2	3	4	5	6	7	8	9	10	11	12

22	21	20	19	18	17	16	15	5	13	5	1

Matched = 11

Slide[1] = 12-1=11

reminder

try out

Example 2 of finding matchJump arrays

List the values in the *charJump* and *matchJump* arrays for the Boyer-Moore Algorithm for the following pattern:

A S T R A C A S T R A

charJump[A] = 0
charJump[R] = 1
charJump[T] = 2
charJump[S] = 3
charJump[C] = 5
charJump[X] = 11

A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11

										1

A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11

									5	1

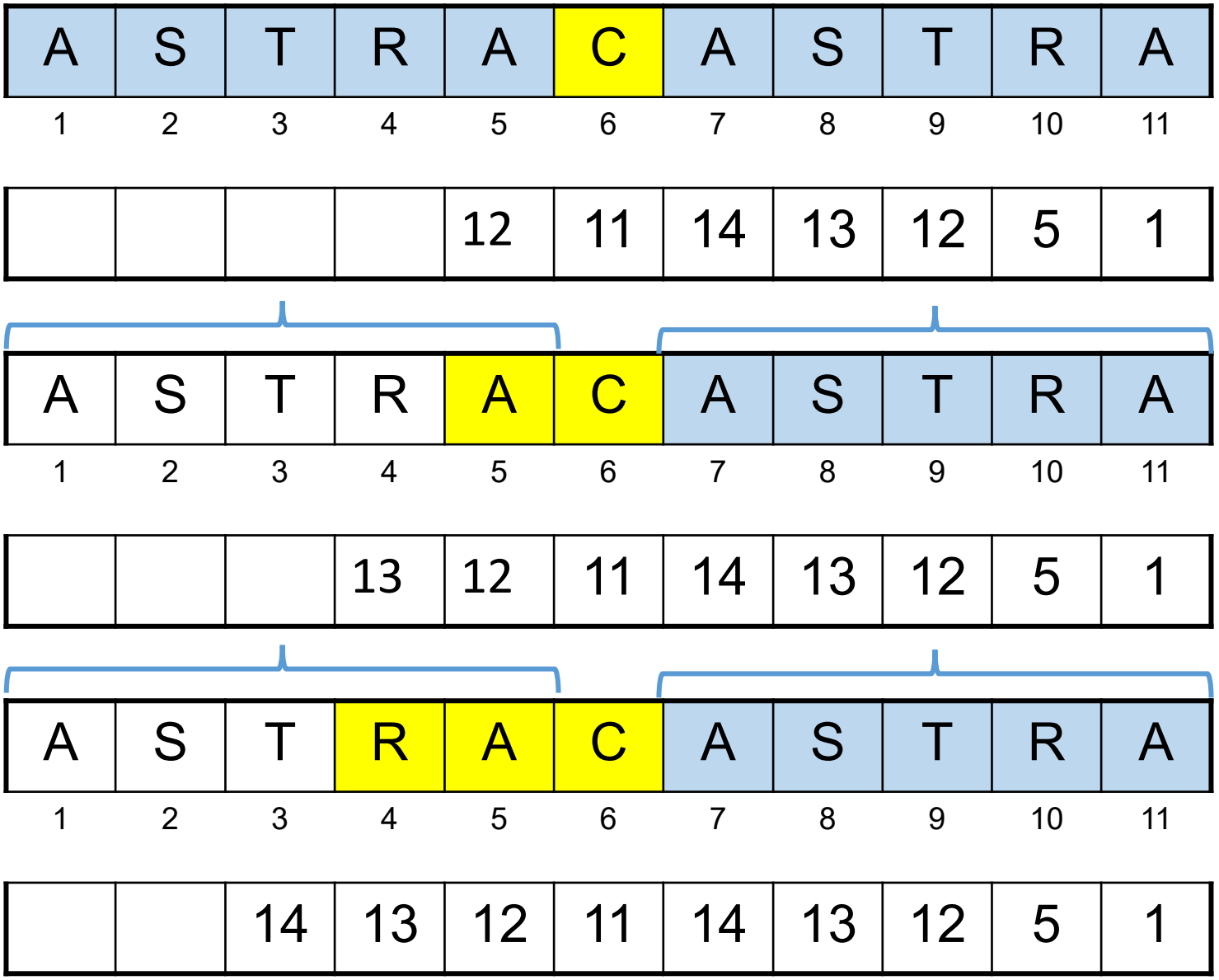
A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11

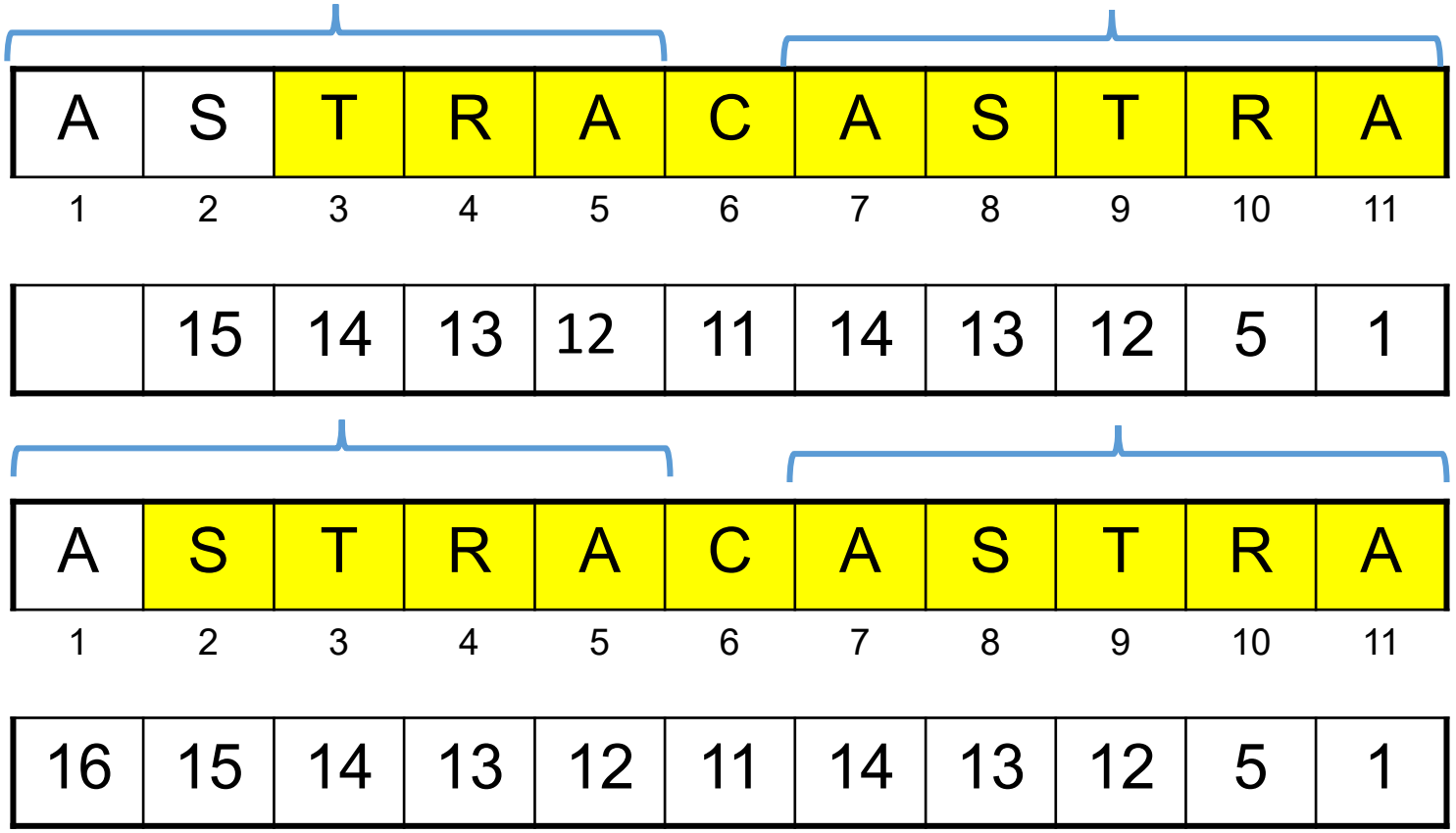
								12	5	1

A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11
							13	12	5	1

A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11
						14	13	12	5	1

A	S	T	R	A	C	A	S	T	R	A
1	2	3	4	5	6	7	8	9	10	11
					11	14	13	12	5	1


A S T R A C A S T R A
1 2 3 4 5 6 7 8 9 10 11
12 11 14 13 12 5 1
A S T R A C A S T R A
1 2 3 4 5 6 7 8 9 10 11
13 12 11 14 13 12 5 1
A S T R A C A S T R A
1 2 3 4 5 6 7 8 9 10 11
14 13 12 11 14 13 12 5 1


A S T R A C A S T R A
1 2 3 4 5 6 7 8 9 10 11
15 14 13 12 11 14 13 12 5 1
A S T R A C A S T R A
1 2 3 4 5 6 7 8 9 10 11
16 15 14 13 12 11 14 13 12 5 1

Please complete the online teaching feedback by 11.59pm, this Sunday.

Thank you!