CSE 566 Spring 2023

Searching & Constructing Suffix Array

Instructor: Mingfu Shao

Suffix Array

S = banana$

SA(s) = (7, 6, 4, 2, 1, 5, 3)

$\boxed{\$ < a < b < n}$

1 banana$	**7** $
2 anana$	**6** a$
3 nana$	**4** ana$
4 ana$	**2** anana$
5 na$	**1** banana$
6 a$	**5** na$
7 $	**3** nana$

Longest Common Prefix (LCP) Array

S = banana$

SA(s) = (7, 6, 4, 2, 1, 5, 3)

LCP(s) = (0, 1, 3, 0, 0, 2)

1	banana$
2	anana$
3	nana$
4	ana$
5	na$
6	a$
7	$

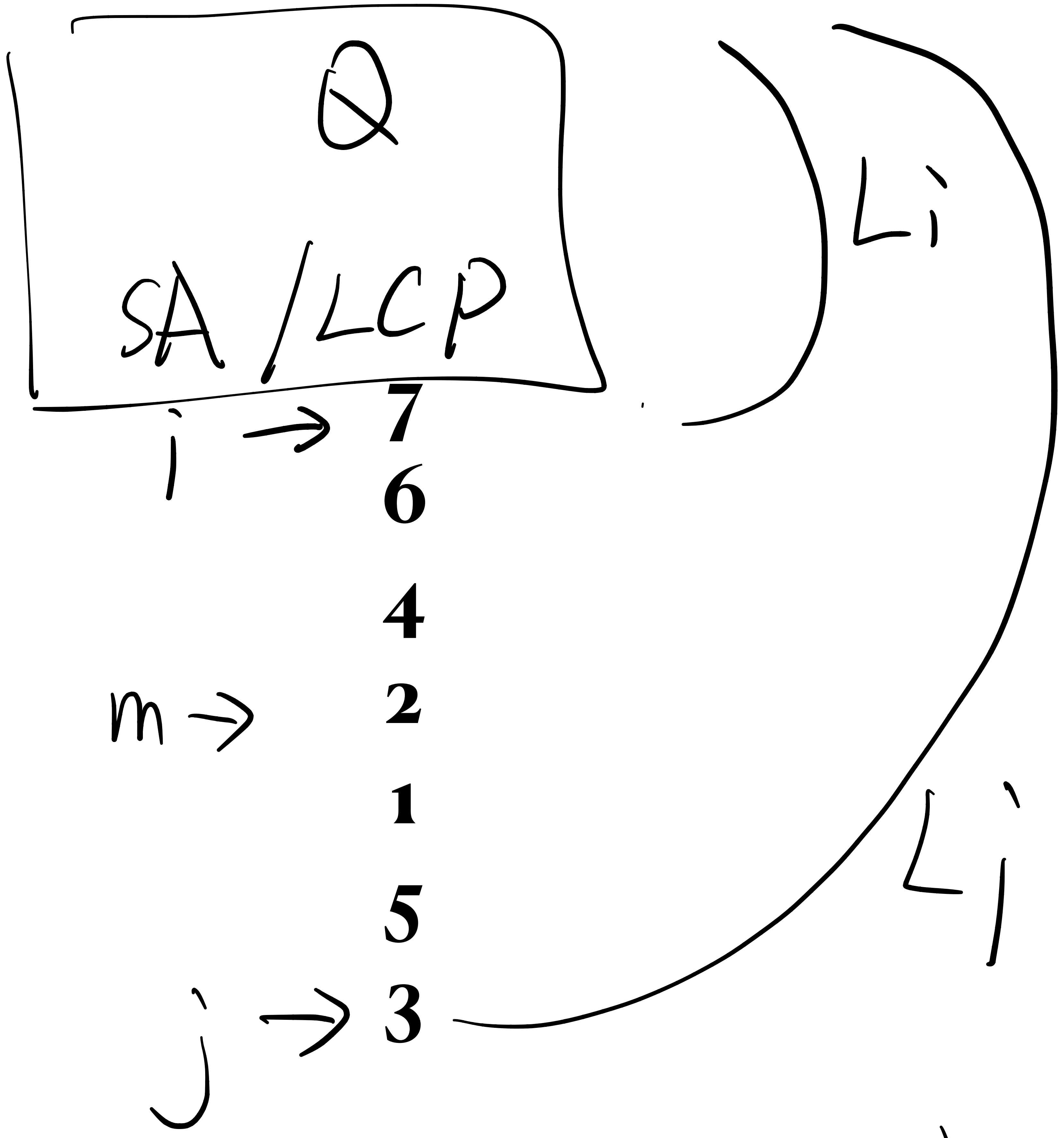
LCP	SA	Suffix
	7	$
0		
	6	a$
1		
	4	ana$
3		
	2	anana$
0		
	1	banana$
0		
	5	na$
2		
	3	nana$

Searching for a query

- Searching questions:
  - Decide if query q Q is the substring of S.
  - Find the longest common substring (LCS) of S and Q such that the LCS starts from Q[1].
- Essentially: find the position of query Q in the sorted list of suffixes
- Key idea: binary search!
- Naive implementation: O(|q| log |S|).

Faster Search Algorithm

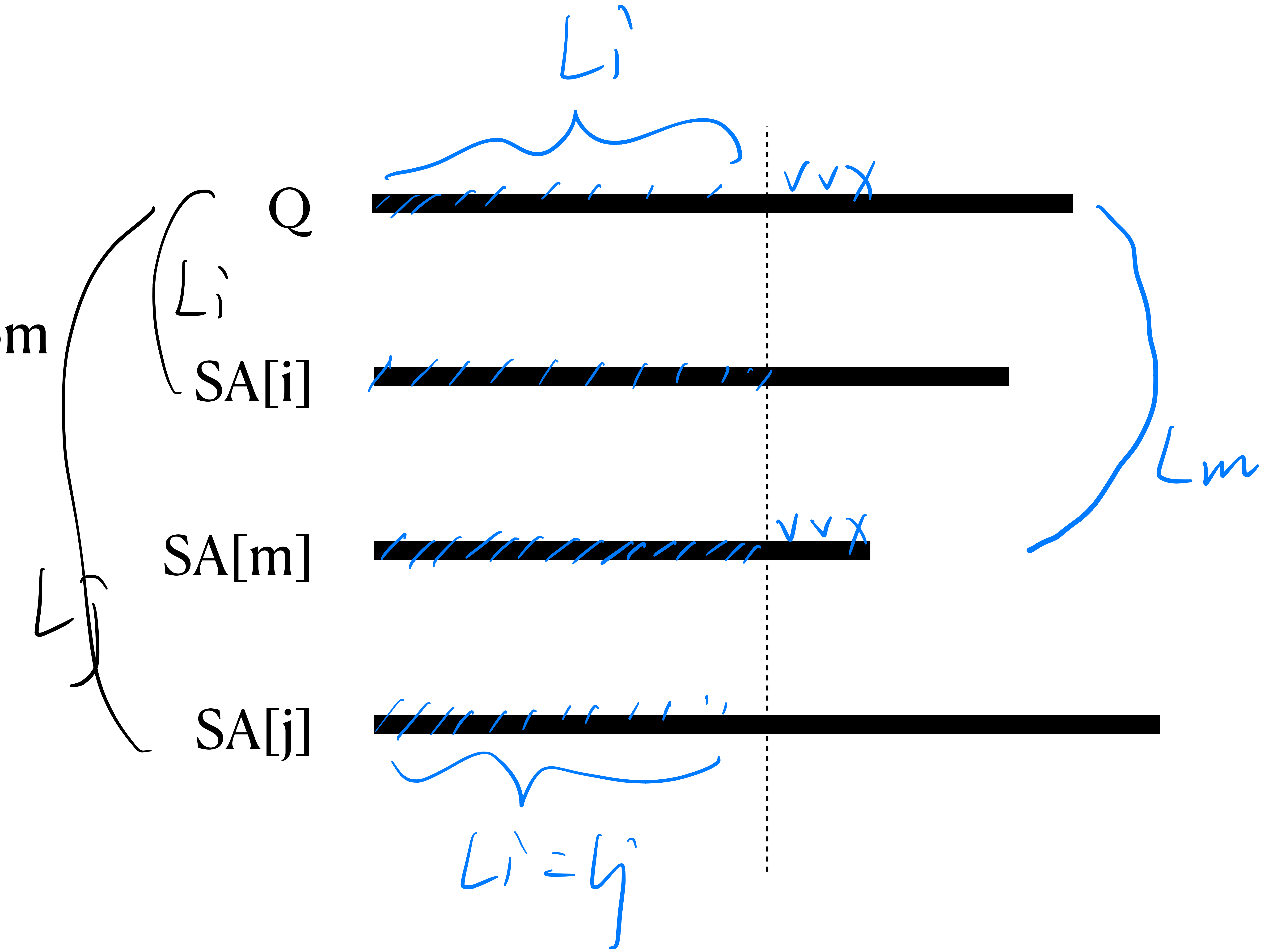
- Check if Q is less than SA[1], or Q is larger than SA[n]
- Init: i = 1 and j = n
- Compute Li := LCP(SA[i], Q) and Lj := LCP(SA[j], Q)
- FUNCTION **BS(i, j, Li, Lj)**
  - Let m = (i + j) / 2
  - Case 1: Li = Lj:
  - Case 2: Li > Lj:
  - Case 3: Li < Lj; //symmetric to Case 2
- END FUNCTION



Running Time: $O(\log|S| + O(|Q|))$

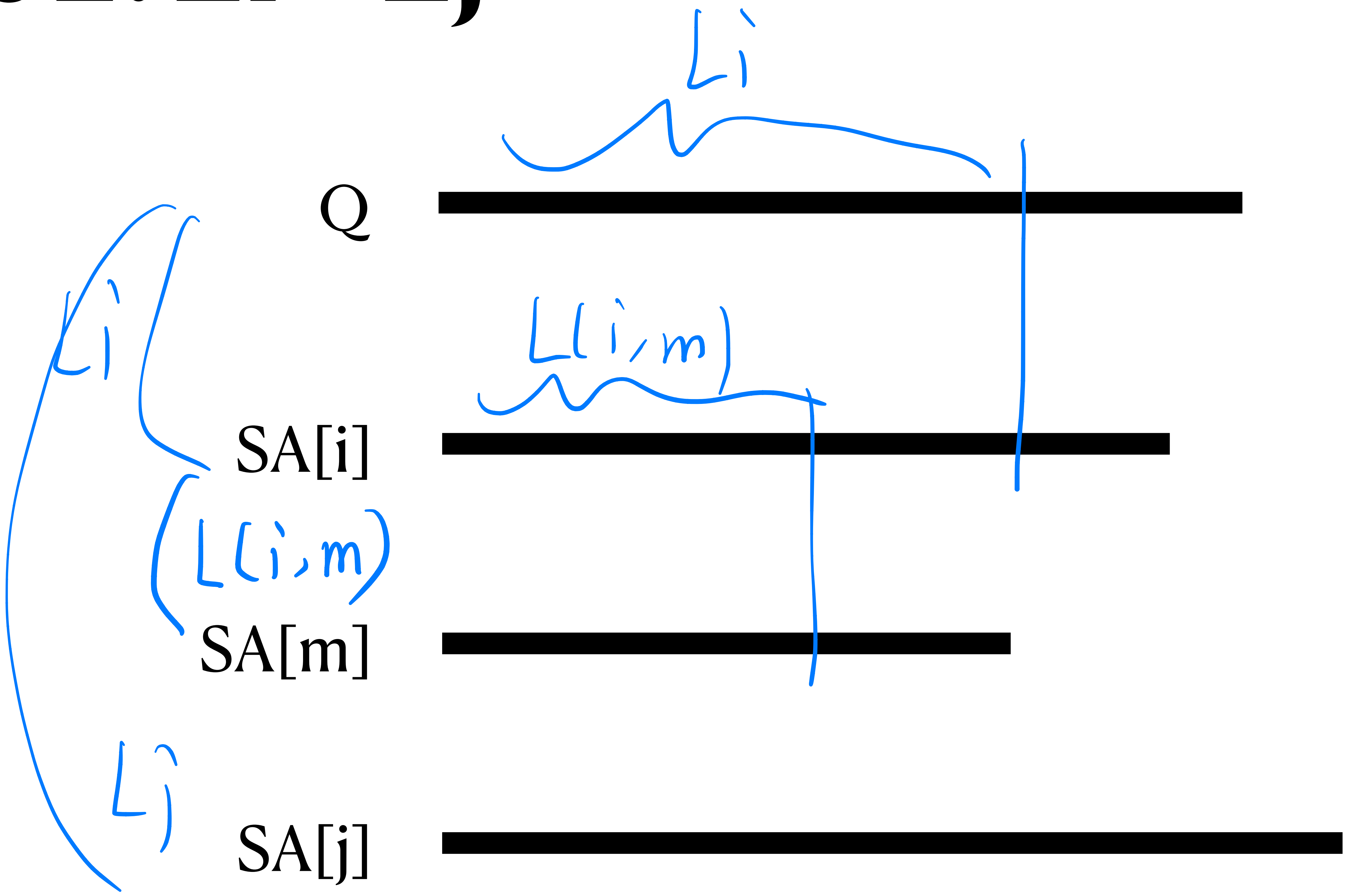
Case 1: Li = Lj

- **Fact**: Lm := LCP(SA[m], Q) >= Li = Lj
- PROCEDURE:
  - Compare Q and SA[m] starting from position Li + 1 => Lm
  - If Q gets exhausted: return m
  - If Q < SA[m]: BS(i, m, Li, Lm)
  - If Q > SA[m]: BS(m, j, Lm, Lj)
- END PROCEDURE



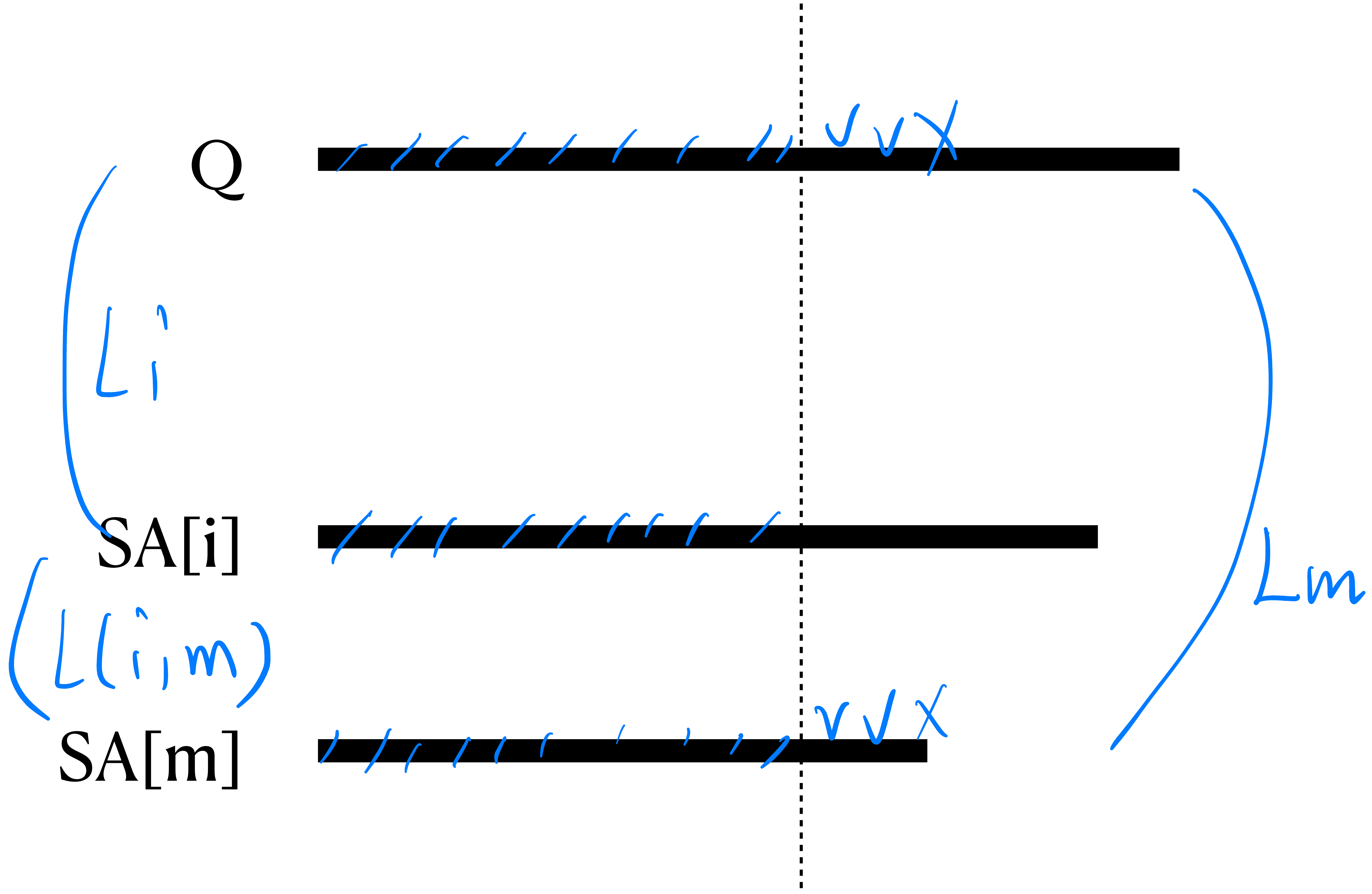
Case 2: Li > Lj

- Query L(i,m) := LCP(SA[i], SA[m])
  - Can be done in constant time!
- Case 2a: L(i, m) = Li
- Case 2b: L(i, m) < Li
- Case 2c: L(i, m) > Li



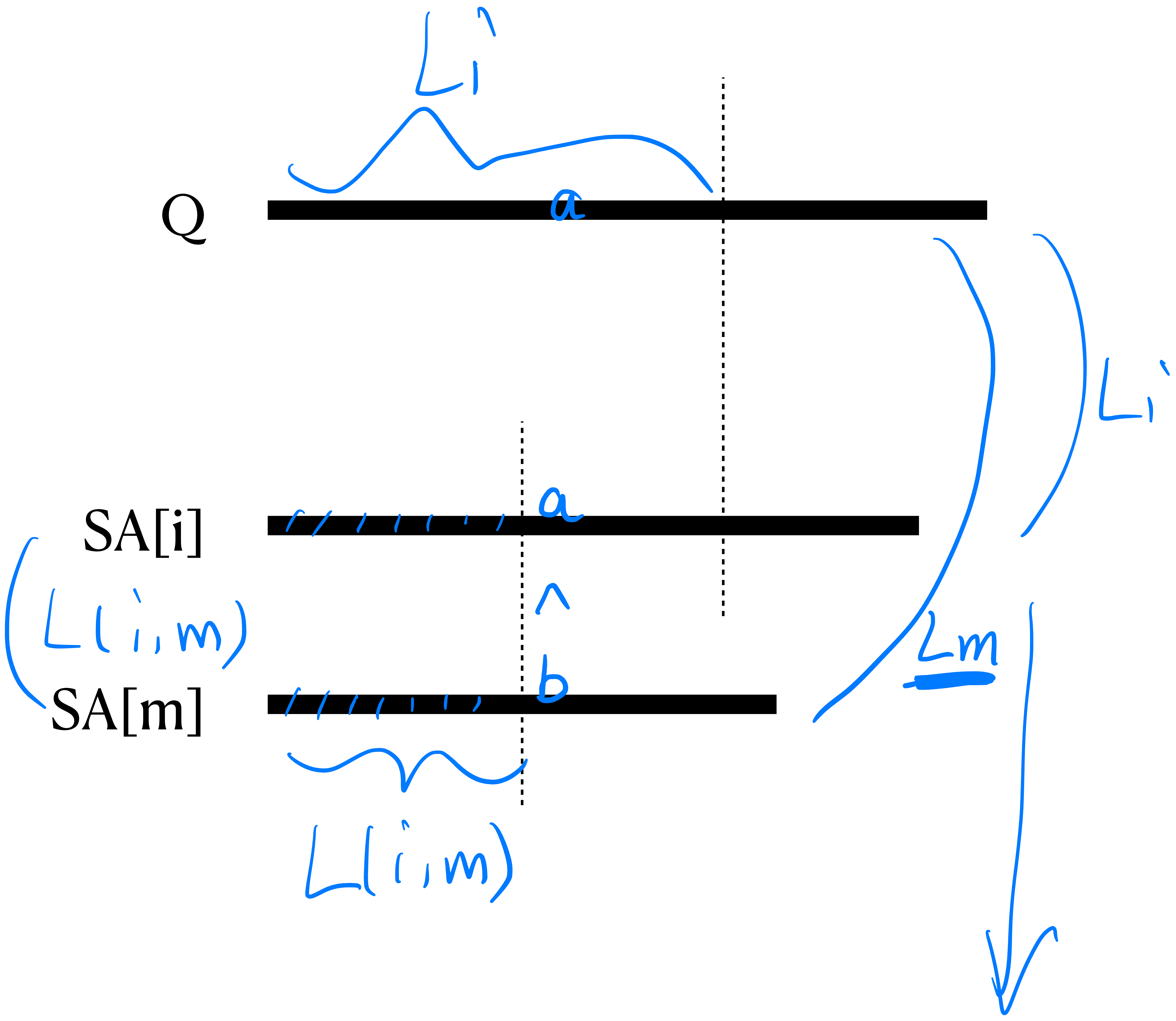
Case 2a: Li > Lj and L(i, m) = Li

- **Fact**: Lm >= Li
- PROCEDURE:
  - Compare Q and SA[m] starting from position Li + 1; this calculates Lm
  - If Q gets exhausted: return m
  - If Q < SA[m]: BS(i, m, Li, Lm)
  - If Q > SA[m]: BS(m, j, Lm, Lj)
- END PROCEDURE



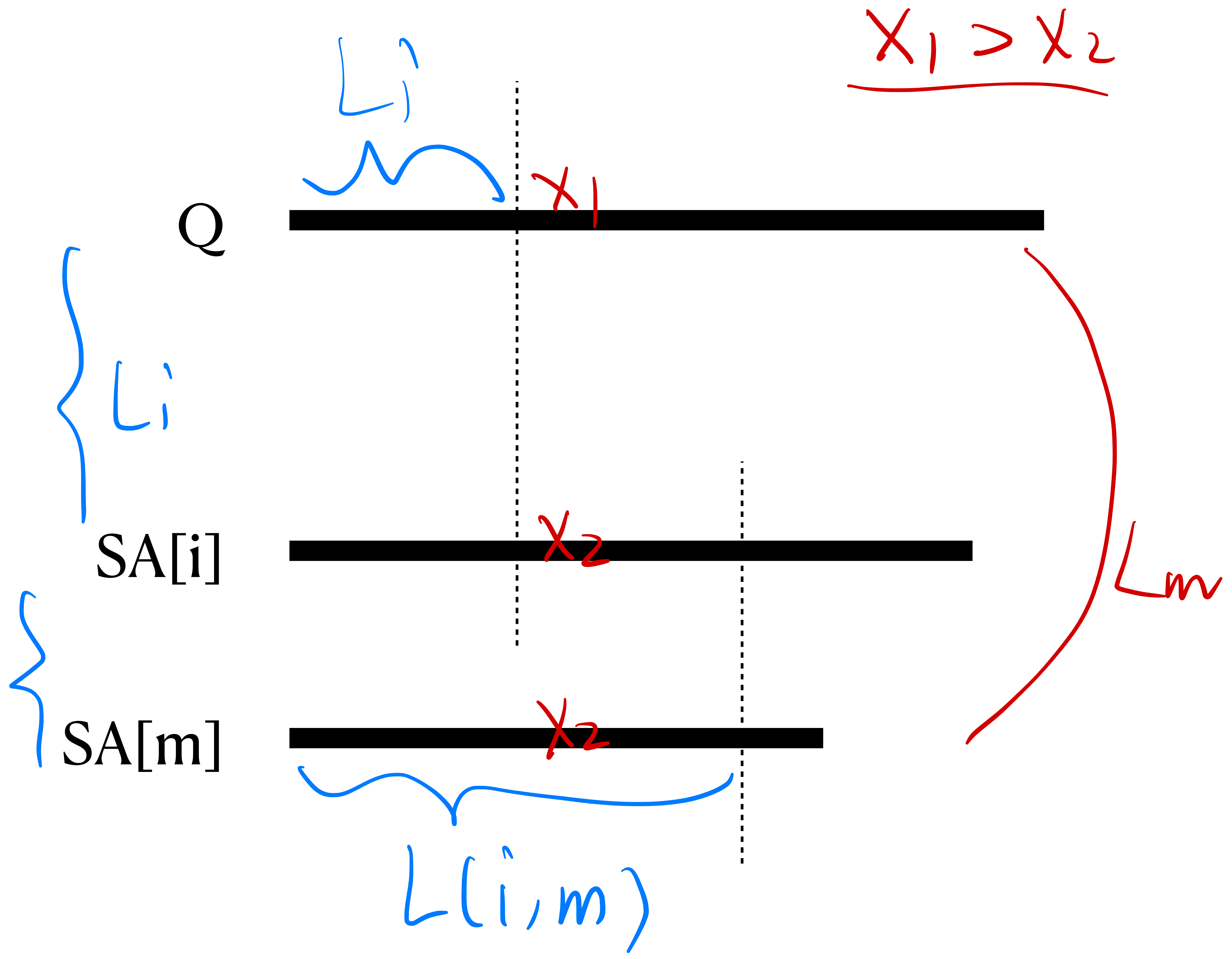
Case 2b: $Li > Lj$ and $L(i, m) < Li$

- **Fact**: $Q < SA[m]$
- **Fact**: $Lm = L(i,m)$
- PROCEDURE:
  - BS(i, m, Li, Lm)
- END PROCEDURE



Case 2c: Li > Lj and L(i, m) > Li

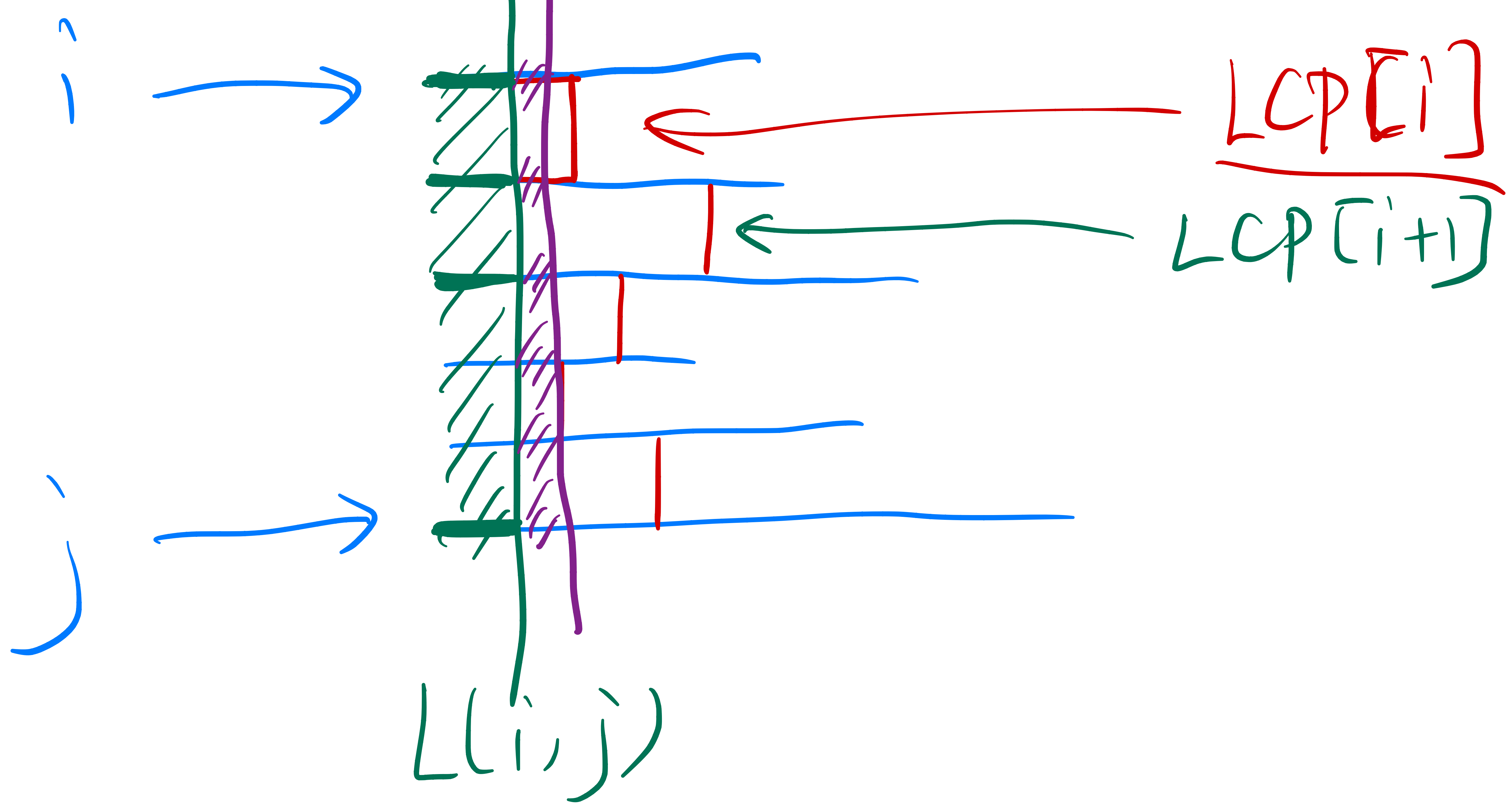
- **Fact**: Q > SA[m]
- **Fact**: Lm = Li
- PROCEDURE:
  - BS(m, j, Lm, Lj)
- END PROCEDURE



Fast Querying LCP

LCP(SA[i], SA[j])

- **Question:** given LCP array, query L(i, j) in constant time, for arbitrary i and j. $i < j$
- **Fact**: L(i, j) = min_{i <= k < j} LCP[k].
- Proof: "≤" "≥"



SA(s) = (7, 6, 4, 2, 1, 5, 3)

LCP(s) = (0, 1, 3, 0, 0, 2)

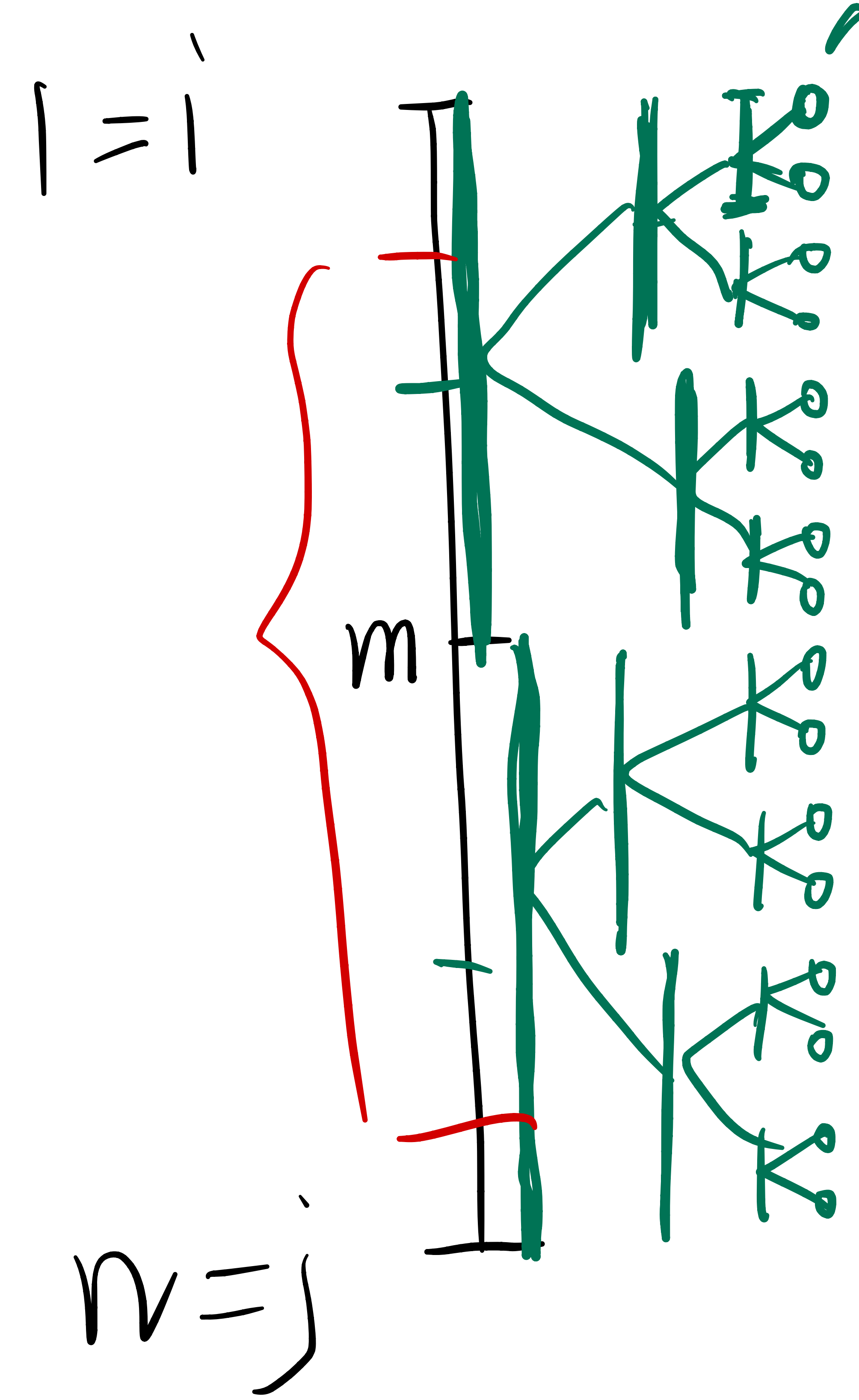
7 $
0
6 a$ ← i
1
4 ana$
3
2 anana$ ← j
0
1 banana$
0
5 na$
2
3 nana$ ← j

Algorithms

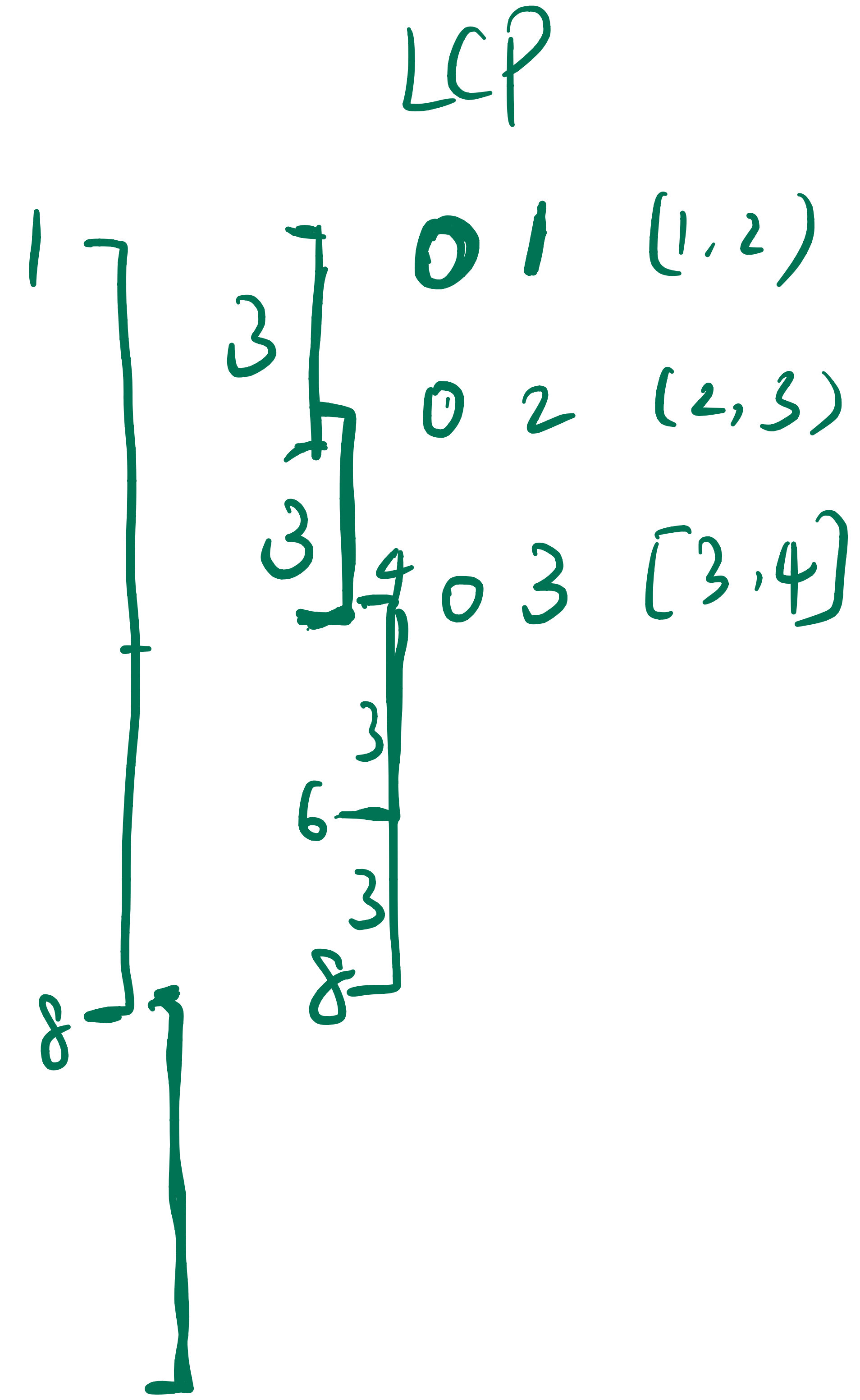
[i, j] for each query

- General algorithm for range-min-query-problem. O(n) space, O(1) time
- **Key Observation:** not every pair of query L(i, j) will be used!

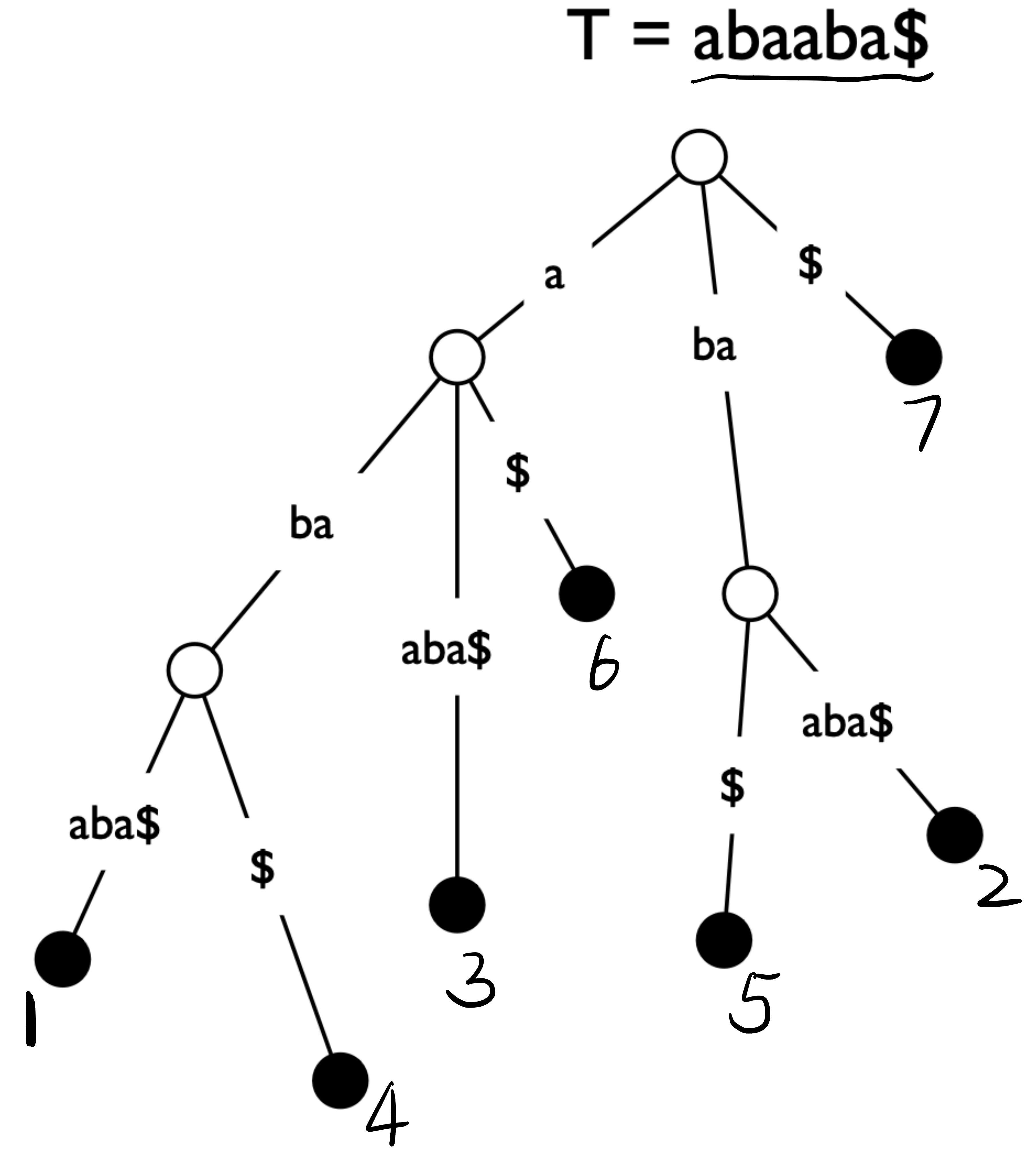
LCP array



#nodes = O(n)



Constructing Suffix Array using Suffix Tree



$O(|T|)$:

traversal the edges out of a vertex following the alphabetical order.

Linear-Time Algorithm

$n = |S|$

D&C $\frac{2}{3}$ $\frac{1}{3}$

Example:

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

S = a b a b b a b a b b a $

1 2 3 1 2 3 1 2 3 1 2 3

Step 1: G1

$\boxed{G2}$

$\boxed{G3}$

token

$|T| = 2n$

$|X| = \frac{2n}{3}$

Step 2: T = b a b | b a b | a b b | a $ $ | a b b | a b a | b b a | $ $ $

X: E