# Video URL: <https://youtu.be/vou9obStcRg>

<https://youtu.be/_XswXyAa3xw>

Remember strings are nothing but character array.

Directly try to use indexes.

Time Complexity with String problems is very tricky.

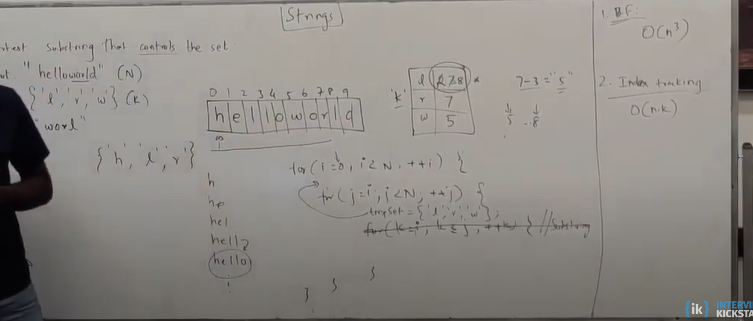
For(int i = 0; I < N; i++) { //O(N)

S1 = s1 + ‘a’; //O(N) – Creating a new string because strings are immutable in java.

}

Time Complexity: O(N^2)

# Shortest substring that controls set



# Approach 1: Brute Force Approach

1. Find all substrings
2. Maintain controlling set for each starting index of substring
3. Remove character from controlling set when found.
4. If the set is empty, we found the substring which contains all control set characters.
5. Calculate the length and update min.

Time Complexity: O(n^2) by maintaining set and removing from set.

It could be O(n^3) if the set is maintained for each substring.

# Approach 2: Using Index tracking

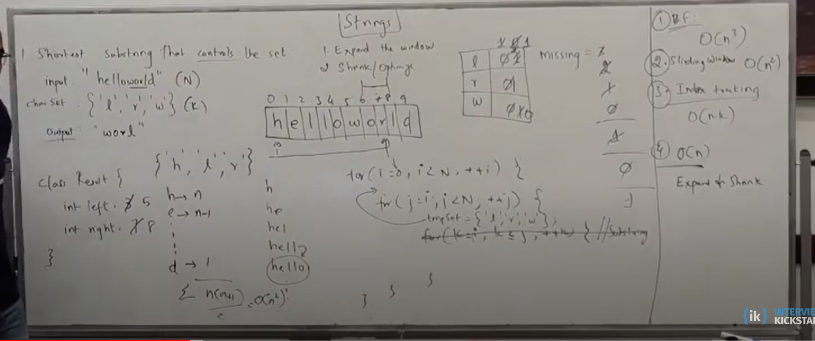
Maintain table of index and character set.

Whenever all three indexes are filled, find the min Index and max index, this will give us length of substring. Update min length.

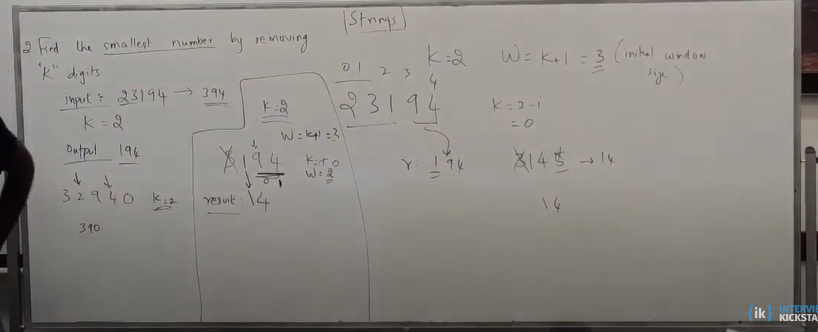
Continue, and again do update if all three index are filled.

Time Complexity: O(nk)

# Approach 3: Optimal Solution O(n)



# Problem: Find smallest number by removing k digits.



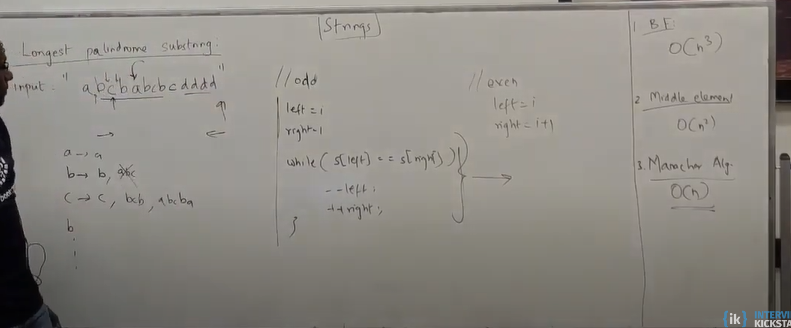
# Problem: Longest Palindrome substring

Whenever there is a subsequence, it becomes a recursive and hence DP problem.

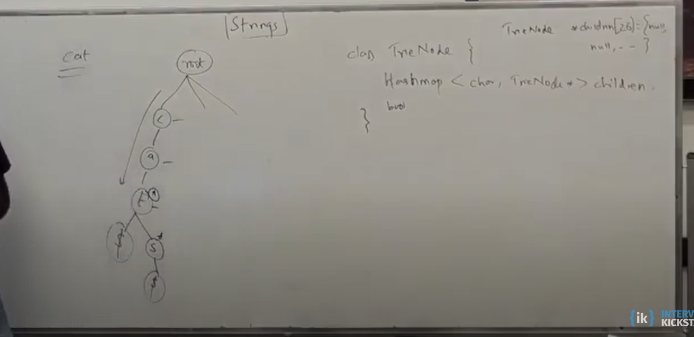
For example – “abcd”

You choose a then you can either choose b or c or d

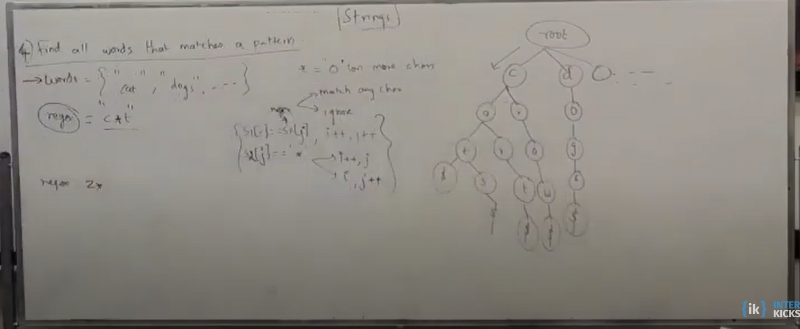
Longest Common Subsequence problem

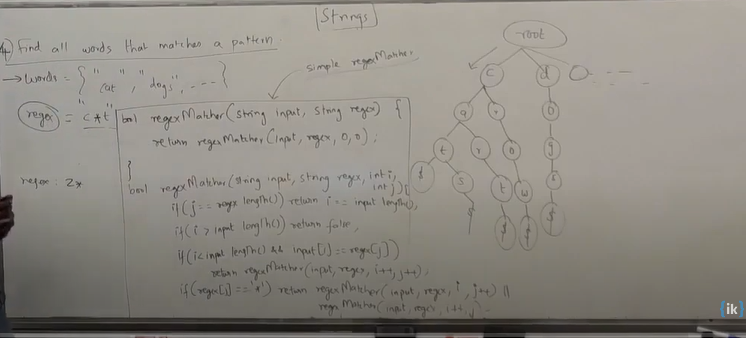


# Trie Data Structure

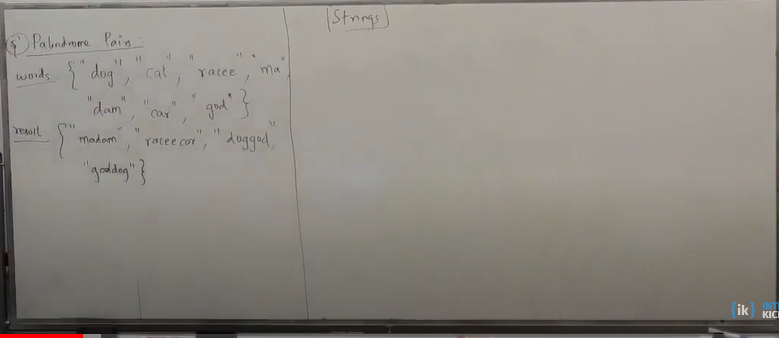


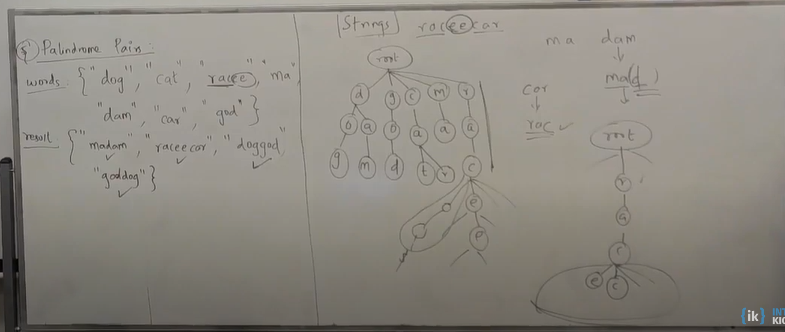
# Pattern Matching





# Palindromic Pairs





Time Complexity:

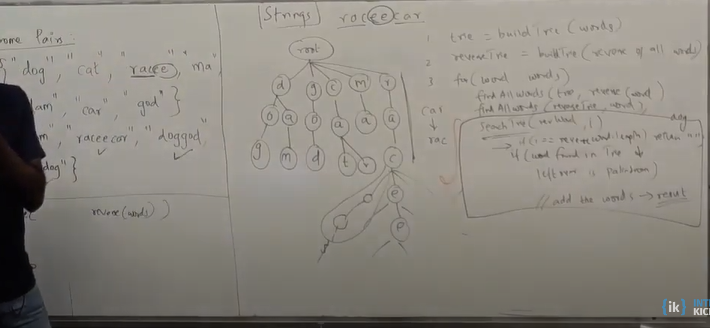
Regular Trie – O(nk)

Reverse Trie – O(nk)

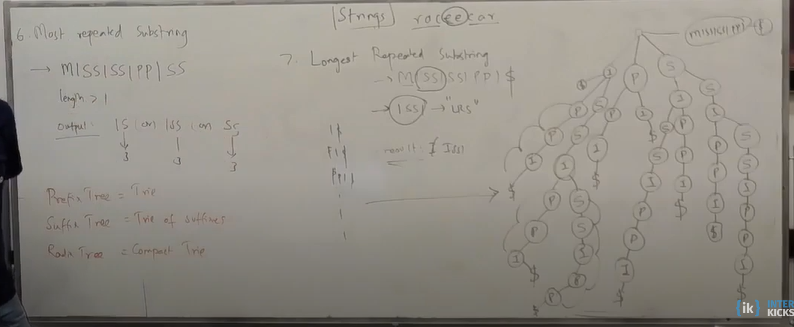
Regular Trie Search – O(nk)

Reverse Trie Search – O(nk)

Space Complexity: O(nk)



# Longest Repeated Substring



# Most Repeated Substring

