# Video URL: <https://youtu.be/N5v_g6qqi9k>

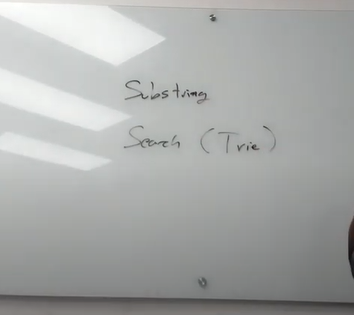
# Problem 1: Reverse string without reversing word

Input: my dog ran

Output ran dog my

# Two types of problems in string

1. Substring
2. Search (which includes Tries and Suffix Tries)



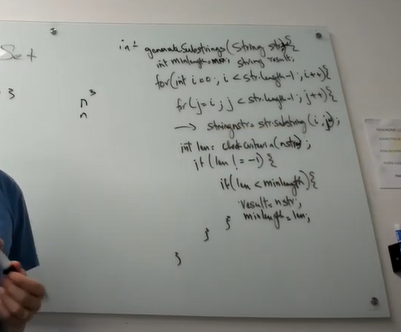
# Controlling Set or Find the smallest substring which contains all the characters from set or Minimum Substring window

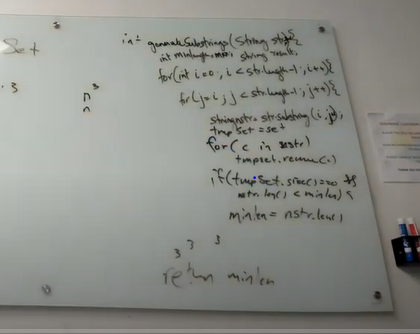
# Brute Force

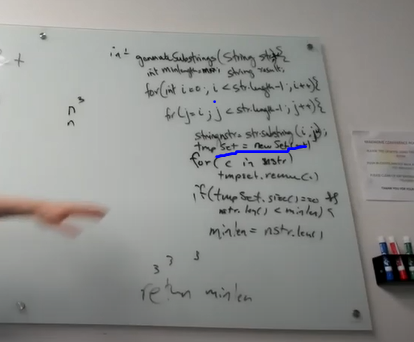
1. Find all substrings O(n^2)
2. Check if all the characters are present in the substrings with the same count O(n)
3. Keep the shortest substring.

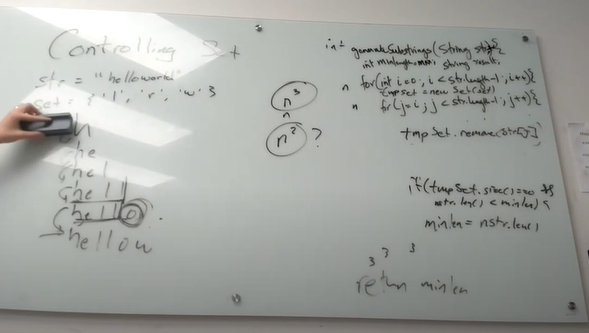
Time Complexity: O(s^3)

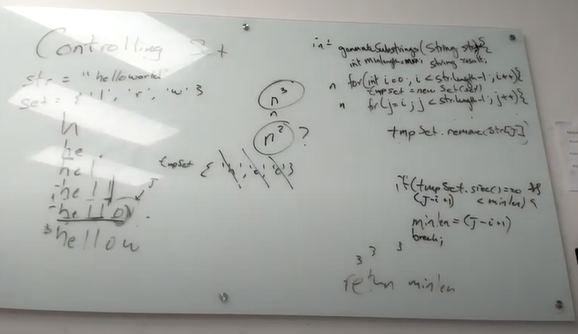
Space Complexity: O(s + t)

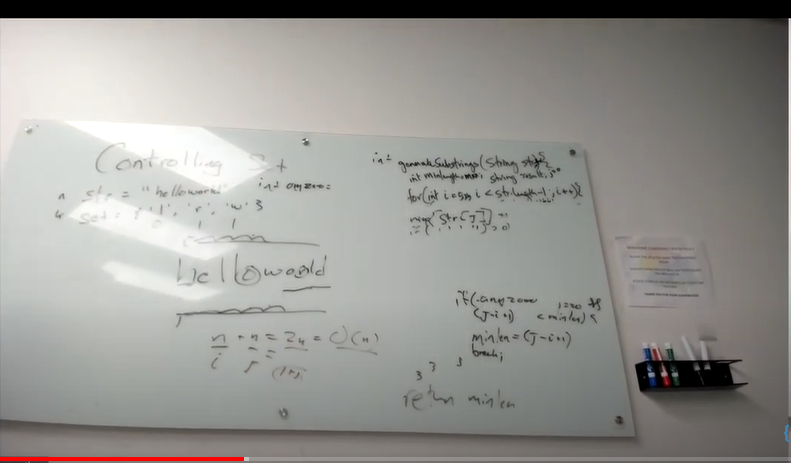




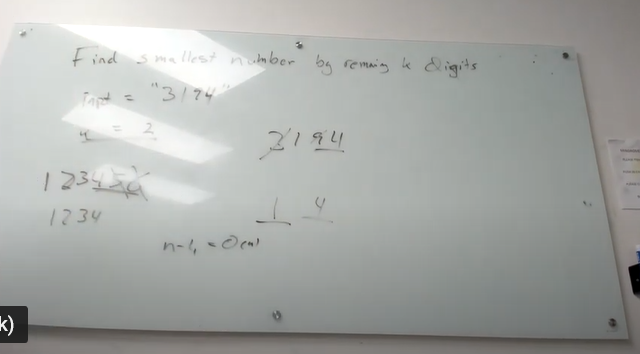








# Problem 2: Build lowest number by removing k digits

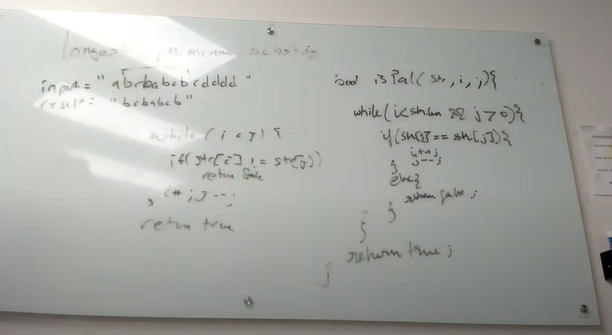


Take window of K + 1 and pick minimum

Then slide the window next to minimum picked.

# Problem 2: Longest Palindrome Substring

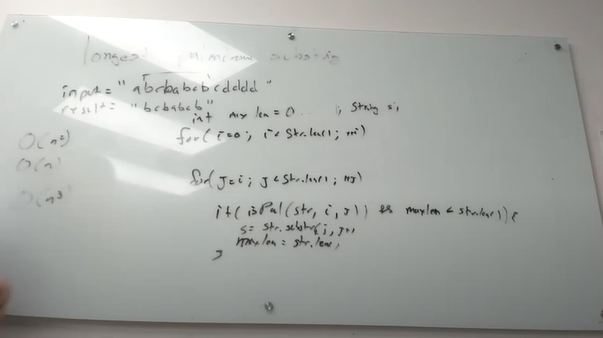
isPalindrome function



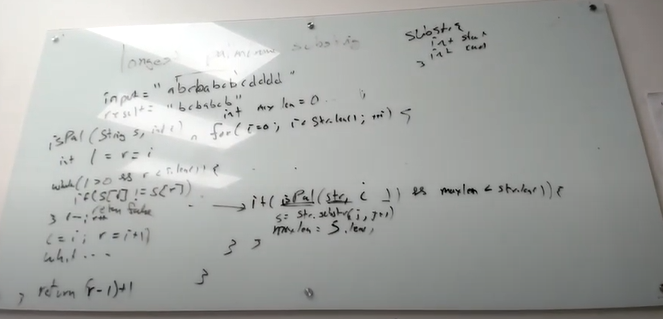
# Brute Force Solution

1. Find all substrings
2. For every substrings, check whether it’s palindrome or not

Time Complexity: O(n^3)



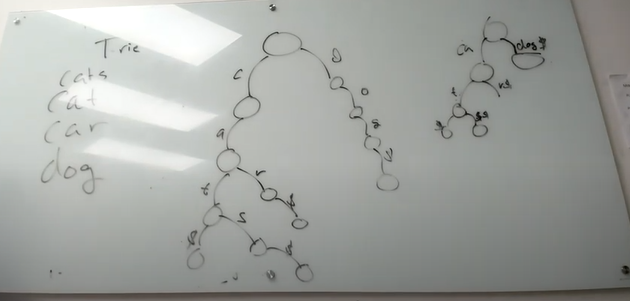
Expand from middle for odd and even length palindrome



# Trie

n-ary tree. The size of n is the size of alphabets and special characters.

# Trie and Radix Tree



# Use Case for Trie

Auto complete box on Google website.

# Another approach for prefix search

Another approach for prefix search is reverse index.

# **Difference between Inverted Index and Forward Index**

[**Inverted Index**](https://www.geeksforgeeks.org/inverted-index/)

1. It is a data structure that stores mapping from words to documents or set of documents i.e. directs you from word to document.
2. Steps to build Inverted index are:
   * Fetch the document and gather all the words.
   * Check for each word, if it is present then add reference of document to index else create new entry in index for that word.
   * Repeat above steps for all documents and sort the words.
3. Indexing is slow as it first checks that word is present or not.
4. Searching is very fast.
5. Example of Inverted index:

Word Documents

hello doc1

sky doc1, doc3

coffee doc2

hi doc2

greetings doc3

It does not store duplicate keywords in index.

1. Real life examples of Inverted index:
   * Index at the back of the book.
   * Reverse lookup

**Forward Index:**

1. It is a data structure that stores mapping from documents to words i.e. directs you from document to word.
2. Steps to build Forward index are:
   * Fetch the document and gather all the keywords.
   * Append all the keywords in the index entry for this document.
   * Repeat above steps for all documents
3. Indexing is quite fast as it only append keywords as it move forwards.
4. Searching is quite difficult as it has to look at every contents of index just to retrieve all pages related to word.
5. Example of forward index:

Document Keywords

doc1 hello, sky, morning

doc2 tea, coffee, hi

doc3 greetings, sky

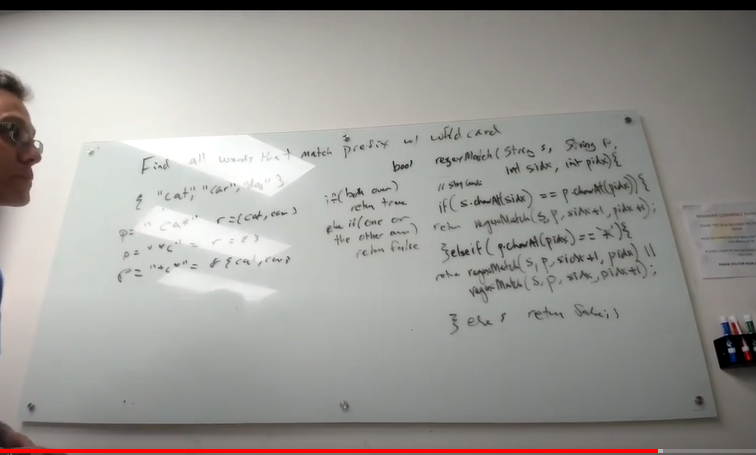
It stores duplicate keywords in index. Eg: word “sky” is stored multiple times.

1. Real life examples of Forward index:
   * Table of contents in book.
   * DNS lookup

**Similarity between Forward index and Inverted Index:**

* Both are used to search text in document or set of documents.

# Find all the words that match prefix with a wild card



Here \* represent 0 or more characters.

P = “ca\*”

Output: cat, car

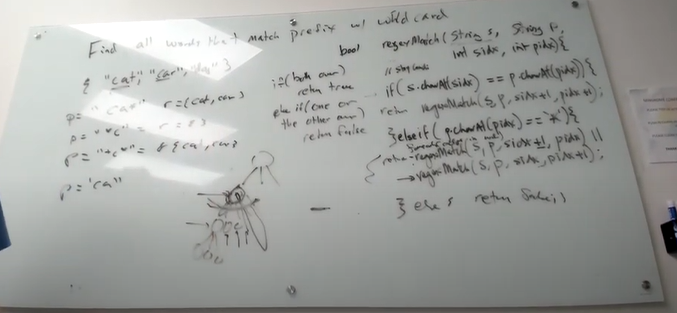
P = “\*c”

Output: {}

P = “\*c\*”

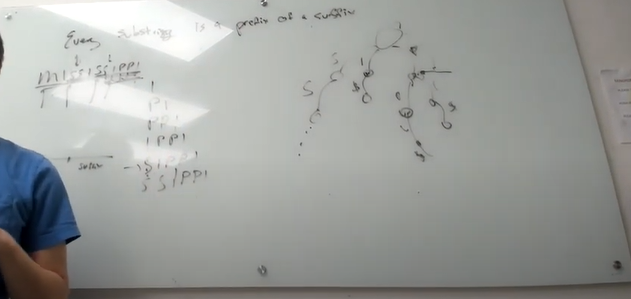
Output: car, cat

# Using Trie implement prefix match with wild card



# Longest Repeated Substring

* + 1. Create Suffix trie
    2. If dollar count > than 1 then I am repeating so append my self.



All substring requires suffix trie.