**Anagram** $\rightarrow$ Two strings are called Anagrams of each other if they are permutation of each other.

LISTEN , SILENT

$\updownarrow$

SILENT

ABC $\longleftrightarrow$ BAC

**Q ⇒** Given two strings, write a program that detects whether they are anagram of each other or not ??

**Ex** RACE, CARE $\longrightarrow$ **True**

SANKET, SAMYAK $\longrightarrow$ **FALSE**

**Brute force** ⇒ If two string are permutation of each other they're called anagram, so, we can generate all the permutations of the first string and check whether any one of them matches the second string.

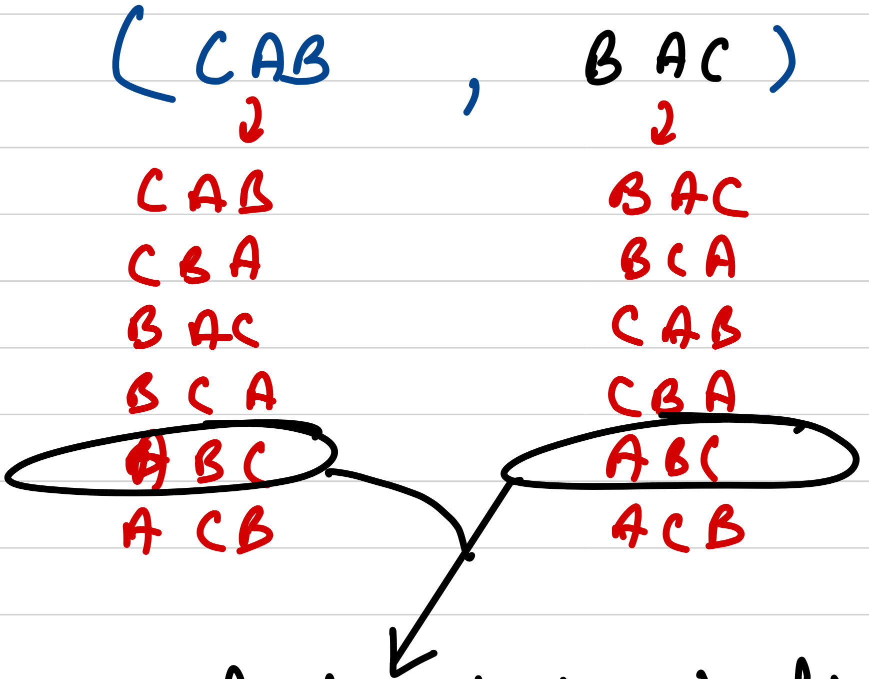
(ABC , BAC)

Yes

$$\left.\begin{array}{l}ABC\\ACB\\BAC\\BCA\\CAB\\CBA\end{array}\right\} 3!$$

O(n!)

## How can we optimise ??



The permutation in which the string is sorted.

Q1

$$\begin{tikzcd}
\text{listen} \arrow[d] & \text{silent} \arrow[d] \\
\boxed{\text{eilnst}} & \boxed{\text{eilnst}}
\end{tikzcd}$$

We know that if two strings are anagrams, they will be have common permutation. And one of the permutation is going to be the sorted arrangement of characters

$$\begin{tikzcd}
\text{care} \arrow[d] & \text{race} \arrow[d] \\
\boxed{\text{acer}} & \boxed{\text{acer}}
\end{tikzcd}$$

$\rightarrow$ time $\rightarrow O(n \log n)$

$\rightarrow$ Let's sort both the strings by chars, and check whether the sorted strings are same or not.

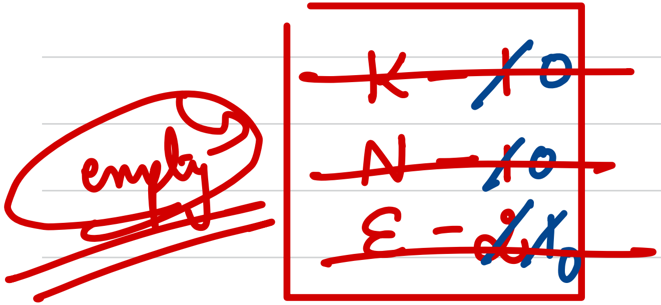
Can we optimize more on time ??

→ If two strings are anagram of each other then →

1) they will have same length
2) they will have exactly same set of characters. i.e. if one string has 2 d's then other should also have 2 d's.

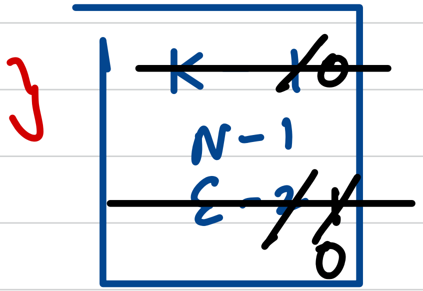
Can we try to detect the fact that characters are same in both strings ??

$S_1 = \underbrace{KNEE}$ , $S_2 = KEEN$



{key-value

$S_1$ = KNEE $S_2$ = KEEL



char ⟷ frequency

→ we will prepare a frequency mapping of $s_1$

→ then for each character in S2, check if we have that char in the mapping of $s_1$. if not return false, else reduce the frequen as we got the char.

KNEE

mp = {

N : 1,

}

$\rightarrow$ [illegible]

KEEt

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ K & N & E & A & E & D \end{array}$$

$i = \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{3}\ 4$ ✓

```
{
  K : 1
  N : 1
  E : 1 2
  A : 1
}
```