Today's agenda

- Char and String
- ASCII
- Problems

- Sunday → optional class (Problem solving) → 8 PM
  - Aloosv Kumar → 7* on Codechef
    masters on Codeforces

String: Sequence (Combination) of Characters
↳ uppercase, lowercase, special etc.

String st = "Algobep";

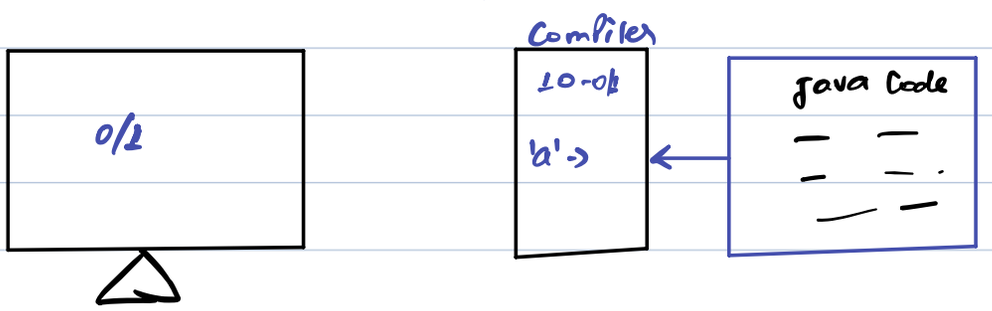
Characters: a) Alphabet
↳ a-z (lowercase)
↳ A-Z (uppercase)

b) Special characters
↳ @ # ? etc.

c) Numbers
↳ '0' - '9'

Syntax:

```
char ch = 'A';
```
(char: style, ch: name, 'A': character)




1. numbers → binary number
2. characters → numbers → binary numbers
(numbers ↔ characters: predefined)

ASCII → 256 Characters

| | | |
|---|---|---|
| 'A' : 65 | 'a' : 97 | '0' : 48 |
| 'B' : 66 | 'b' : 98 | '1' : 49 |
| 'C' : 67 | 'c' : 99 | |
| ⋮ | ⋮ | ⋮ |
| 'Z' : 90 | 'z' : 122 | '9' : 57 |
| | | '10' : Not a Character |

→ input

→ get

## * Char rules

1. Char to int : Implicit. eg: int x = 'A'; ↙

2. int to Char : Complicated. → (Typecast always)
   Explicity.

```
Char ch = (char)66;
```

Quiz 1 :

```
Char ch1 = 'B';
S.O.P(ch1);
```

Quiz 2 :

```
Char ch2 = 66;
S.O.P(ch2);
```

→ typecast automatic → 'B'

Quiz 3 :

```
Char ch3 = 'xyz';
S.O.P(ch3);
```

→ error

Quiz 4 :

```
int x = 'A';
x = x + 2;
System.out.println(x);
```

→ 65

x = 67

→ 67

→ mathematical operation in characters allowed (ASCII)

**Quiz5:**

```
char ch4 = 'A';
ch4 = (char) ch4 + 3;
S.O.p (ch4);
```

(A → 65, D → 68)

Explicit conversion.

**Quiz6:**

```
char ch5 = 'A';
if (ch5 >= 90) {
    S.O.p ("Greater");
}
else {
    S.O.p ("Smaller");
}
```

(65 >= 90 is false → Smaller)



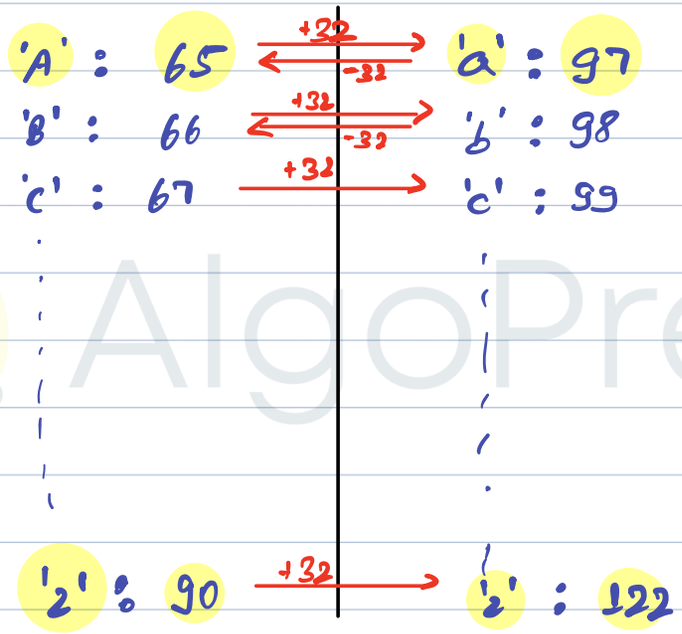
Break till 9:45 PM

## Q) Toggle Character

Given a `char[]` which contains only small and capital letters, toggle them.

↳ lowercase → uppercase
uppercase → lowercase

Eg: AlgoPrep : aLgOpREP






uppercase to lowercase → +32
lowercase to uppercase → -32

//Pseudo code

```
P S  void main ( ) {
    Scanner scn = new Scanner (System.in);
     int  n = scn.nextInt();
    char[] ch = new char[n];
     String st = scn.nextLine();
      for(int i=0; i<n; i++) {

        ch[i] = st.charAt(i);
      }
     Toggle (ch);
}
```

```
P S  void Toggle (char[] ch) {

    for(int i=0; i<ch.length; i++) {
        if (ch[i]>=65 && ch[i]<=90) { //uppercase
          ch[i] = (char)(ch[i] + 32);

        } else { //lowercase
          ch[i] = (char)(ch[i]-32);
        }
    }
}
```

Q) Reverse the given String

↳ Given a String str, reverse and Print it.

ex: algoprep : perpogla

// Pseudo Code

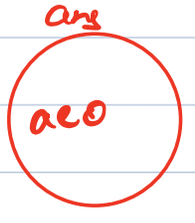
```
P S String reverseString (String str){
    int n = str.length();
    Char [] ch = str.toCharArray();

    int sp = 0;
    int ep = n-1;

    while (sp<ep){
        Char temp = ch[sp];
        ch[sp] = ch[ep];
        ch[ep] = temp;
        sp++;  ep--;
    }

    String ans = str.valueOf(ch);
    return ans;
}
```

```
String st = "Hello";
st = st + "e";
S.O.p (st); → Helloe
```

```
char [] ch = { 'a', 'e', 'o', 'm' };
String ans = "";
```

| i | ans |
|---|---|
| 0 | "a" |
| 1 | "ae" |
| 2 | "aeo" |
| 3 | "aeom" |

```
for (int i=0; i<ch.length; i++) {
    ans = ans + ch[i];
}
```






# Toggle Character

## Java Code:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);
        String st=scn.nextLine();

        char[] ch = st.toCharArray();
        toggle(ch);
        for(int i=0;i<ch.length;i++){
            System.out.print(ch[i]);
        }
            }

            private static void toggle(char[]ch) {

                    for(int i=0;i<ch.length;i++){
                            if(ch[i] >= 65 && ch[i]<= 90){
                ch[i] = (char)(ch[i] + 32);
            }else{
                ch[i] = (char)(ch[i] - 32);
            }
                    }


    }
}
```

## C++ Code:

```
#include <iostream>
#include <cctype>
using namespace std;

void toggle(char *ch, int length) {
    for (int i = 0; i < length; i++) {
        if (isupper(ch[i])) {
```

```
            ch[i] = tolower(ch[i]);
        } else {
            ch[i] = toupper(ch[i]);
        }
    }
}

int main() {
    string st;
    getline(cin, st);

    char ch[st.length() + 1];
    strcpy(ch, st.c_str());

    toggle(ch, st.length());

    for (int i = 0; i < st.length(); i++) {
        cout << ch[i];
    }

    return 0;
}
```

## Python Code:

```
def toggle(ch):
    toggled = []
    for char in ch:
        if char.isupper():
            toggled.append(char.lower())
        else:
            toggled.append(char.upper())
    return ''.join(toggled)

def main():
    st = input()
    toggled_string = toggle(st)
    print(toggled_string)

if __name__ == "__main__":
    main()
```

# Reverse String

## Java Code:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);
        String st=scn.nextLine();

        char[] ch = st.toCharArray();
        reverse(ch);
        for(int i=0;i<ch.length;i++){
            System.out.print(ch[i]);
        }
    }

    private static void reverse(char[]ch) {

        int sp =0;
        int ep = ch.length - 1;

        while(sp < ep){
            char temp = ch[sp];
            ch[sp] = ch[ep];
            ch[ep] = temp;
            sp++;
            ep--;
        }



    }
}
```

## C++ Code:

```
#include <iostream>
```

```
#include <cstring>
using namespace std;

void reverse(char *ch, int length) {
    int sp = 0;
    int ep = length - 1;

    while (sp < ep) {
        char temp = ch[sp];
        ch[sp] = ch[ep];
        ch[ep] = temp;
        sp++;
        ep--;
    }
}

int main() {
    string st;
    getline(cin, st);

    char ch[st.length() + 1];
    strcpy(ch, st.c_str());

    reverse(ch, st.length());

    for (int i = 0; i < st.length(); i++) {
        cout << ch[i];
    }

    return 0;
}
```

## Python Code:

```
def reverse(ch):
    sp = 0
    ep = len(ch) - 1

    while sp < ep:
        ch[sp], ch[ep] = ch[ep], ch[sp]
        sp += 1
        ep -= 1

def main():
    st = input()
```

```
    ch = list(st)
    reverse(ch)
    reversed_string = ''.join(ch)
    print(reversed_string)

if __name__ == "__main__":
    main()
```

# Insert Difference_HW

## Solution Vid:

https://youtu.be/6u99_gdqtOU

## Java Code:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);
                String st=scn.nextLine();
                System.out.println(insert(st));
        }

        private static String insert(String st) {
                String ans = "";

                for(int i=0;i<st.length()-1;i++){
                        char ch1 = st.charAt(i);
                        ans = ans + ch1;
                        int temp = st.charAt(i+1)-st.charAt(i);
                        ans = ans + temp;
                }

        ans = ans + st.charAt(st.length()-1);
        return ans;

    }
}
```

## C++ Code:

```
#include <iostream>
#include <string>
using namespace std;

string insert(string st) {
    string ans = "";

    for (int i = 0; i < st.length() - 1; i++) {
        char ch1 = st[i];
        ans += ch1;
        int temp = st[i + 1] - st[i];
        ans += to_string(temp);
    }

    ans += st[st.length() - 1];
    return ans;
}

int main() {
    string st;
    getline(cin, st);

    string result = insert(st);
    cout << result << endl;

    return 0;
}
```

## Python Code:

```
def insert(st):
    ans = ""

    for i in range(len(st) - 1):
        ch1 = st[i]
        ans += ch1
        temp = ord(st[i + 1]) - ord(st[i])
        ans += str(temp)

    ans += st[-1]
    return ans

def main():
```

```
    st = input()
    result = insert(st)
    print(result)

if __name__ == "__main__":
    main()
```

# Is Palindrome_HW

## Solution Vid:

https://youtu.be/QBsaSTN48kA

## Java Code:

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);
        String st=scn.nextLine();
        int n = st.length();

        char[] arr = new char[n];
        for(int i=0;i<n;i++){
            if(st.charAt(i)>='A' && st.charAt(i)<='Z'){
                arr[i] = (char)(st.charAt(i) + 32);
            }else{
                arr[i] = st.charAt(i);
            }
        }

        int sp = 0;
        int ep = n-1;

        while(sp<ep){
            if(arr[sp]!= arr[ep]){
                System.out.println("false");
                return;
            }
```

```
            sp++;
            ep--;
        }

        System.out.println("true");
    }
}
```

## C++ Code:

```
#include <iostream>
#include <cctype>
using namespace std;

int main() {
    string st;
    getline(cin, st);
    int n = st.length();

    char arr[n];
    for (int i = 0; i < n; i++) {
        if (isupper(st[i])) {
            arr[i] = tolower(st[i]);
        } else {
            arr[i] = st[i];
        }
    }

    int sp = 0;
    int ep = n - 1;

    while (sp < ep) {
        if (arr[sp] != arr[ep]) {
            cout << "false" << endl;
            return 0;
        }

        sp++;
        ep--;
    }

    cout << "true" << endl;

    return 0;
}
```

## Python Code:

```
def main():
    st = input()
    n = len(st)

    arr = [0] * n
    for i in range(n):
        if st[i].isupper():
            arr[i] = st[i].lower()
        else:
            arr[i] = st[i]

    sp = 0
    ep = n - 1

    while sp < ep:
        if arr[sp] != arr[ep]:
            print("false")
            return

        sp += 1
        ep -= 1

    print("true")

if __name__ == "__main__":
    main()
```

Today's agenda

- Prefix and Suffix String
- LPS of a given String
- LPS array of a given String
- Problems

Q) Given a String of size N.

Prefix Strings : All the subStrings starting from 0th idx.
Suffix Strings : " " " ending at last idx.

ex: a b d b

| Prefix Strings | Suffix Strings |
| --- | --- |
| a | b |
| a b | d b |
| a b d | b d b |
| a b d b | a b d b |

Q) Given $S_n$, LPS of the String.

↳ Length of longest Prefix which is also a suffix.

note: other than entire string

(longest Proper Prefix which is also Proper suffix)

Ex: S = b d a b d

| Prefix | Suffix |
|---|---|
| b | d |
| b d | b d → 2 |
| b d a | a b d |
| b d a b | d a b d |
| b d a b d | b d a b d |

$S = \overset{0}{S_0}\ \overset{1}{S_1}\ \overset{2}{S_2}\ \overset{3}{S_3}\ \overset{4}{S_4}$

| | | iter: |
|---|---|---|
| $S_0$ | $S_4$ | 1 |
| $S_0 S_1$ | $S_3 S_4$ | 2 |
| $S_0 S_1 S_2$ | $S_2 S_3 S_4$ | 3 |
| $S_0 S_1 S_2 S_3$ | $S_1 S_2 S_3 S_4$ | 4 |
| | | ⋮ |
| | | n-1 |

T.C: $\frac{N*(N-1)}{2}$

$\frac{(N-1)*N}{2} \simeq O(N^2)$

Q) Given $S_N$, return LPS[N]

Ex: S[7]: a a b a a b a

LPS[7]: 0 1 0 1 2 3 4

LPS[i] = LPS value of substring {0, ..., i}

i:3 → LPs of aaba

Proper Prefix: a, aa, aab

Proper Suffix: a, ba, aba

LPS[3] = 1

i:5 → LPs of aabaab

Proper Prefix: a, aa, aab, aaba, aabaa

Proper Suffix: b, ab, aab, baab, abaab

LPS[5] = 3

T.C: $N * N^2 \simeq O(N^3)$

expected T.C: $O(N)$ → we will cover this in 2nd class of String.

```
int[] LPS(String s){
    // already written in O(N)
}
```

Q) Search for a given Pattern P in Text T.

↳ KMP

ex: $T_N$ : b C a c d a → True

$P_K$ : a c d a

## Idea 1

$T_N$ : b C a c d a

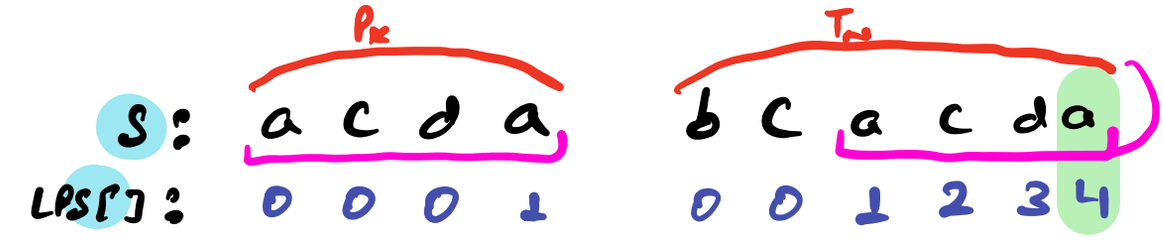
$P_K$ : a c d a

T.C: $O(N * K)$ = $K = N/2$ → $O(N^2)$

## Idea 2

$T_N$ : b C a c d a

$P_K$ : a c d a



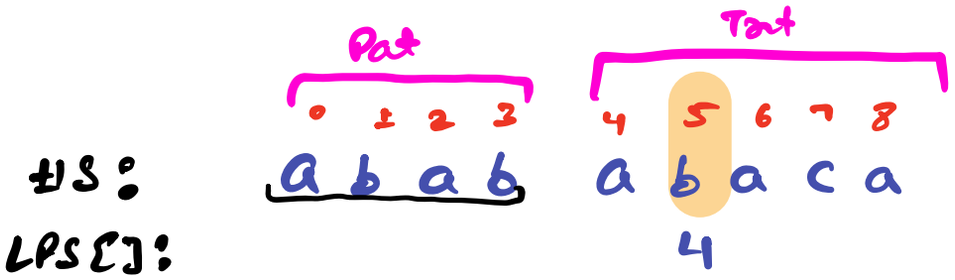

↳ if LPS[i] == P.length, K → return true.

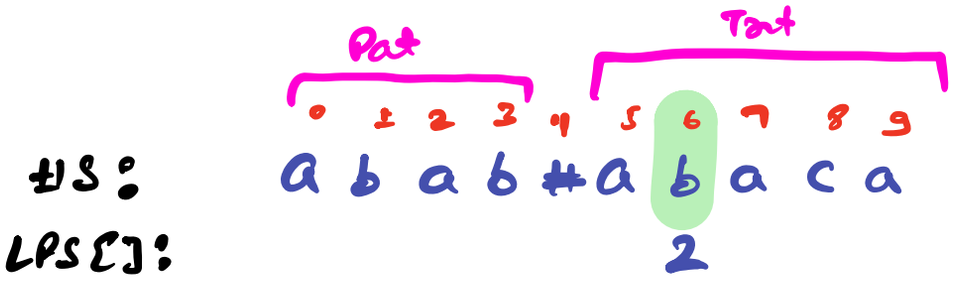
T.C: $O(N+K)$

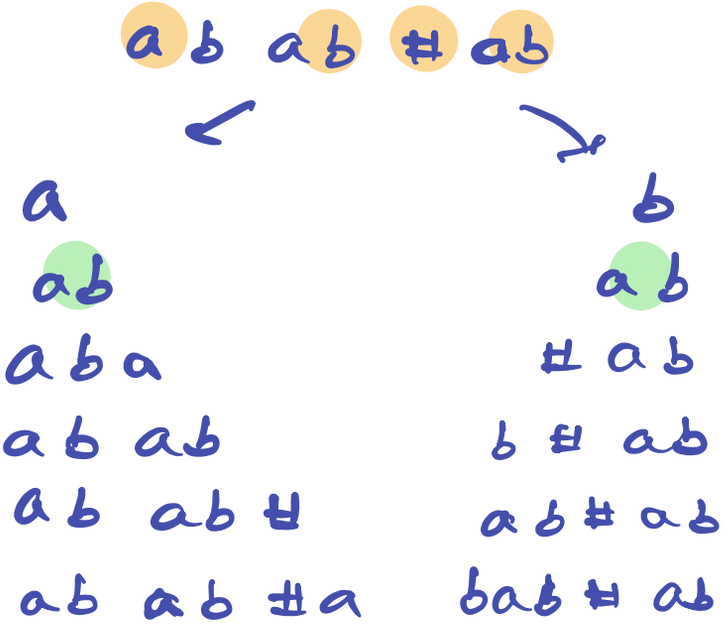
edge case

$T_{n}$ : a b a ca → false

$P_{n}$ : a b a b



Steps:

1. S = Pat + "#" + text → O(N+K)

2. Calculate LPS[] of S[] → O(N+K)

3. if (LPS[i] == Pat.length){ return true; }

⋮

if none of the idx equals Pat.length
return false.

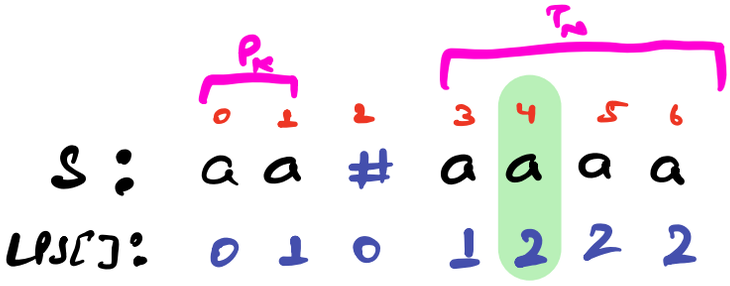
T.C: O(N+K)

Q) No. of Substrings of given Pattern P in Text T.

Ex:

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| $T_N$: | a | a | a | a | → ans: 3 |
| $P_K$: | a | a | | | |

//idea 1




↳ ans = 0 + 1 + 1 + 1 = 3

→ count of P.length in your LPS array of S.

T.C: O(N+K)

Q) Given a String S, min characters to be added at the start of String to make entire String Palindrome.

Ex: d c a d a c d → ans = 2

Ex: d c a a b b a a c d → ans = 3

Ex: a b a d a a d a b a → ans = 4

//idea1

S = a b c b a d e f (0 1 2 3 4 5 6 7)

isPalindrome → O(N)

Str[0-7] NO

Str[0-6] NO

Str[0-5] NO

Str[0-4] (i) yes → ans : N-1-i

T.C: N * O(N) = O(N²)

## Idea 2

Palind. String → reversed

a b c b a → a b c b a

a b c b a d → d a b c b a

a b c b a d e → e d a b c b a

→ s = a b c b a d e f (indices 0 1 2 3 4 5 6 7)

s1: a b c b a d e f # f e d a b c b a (i)

LPS of s1 = 5

→ LPS of whole string = length of longest possible Palindrome starting from 0th idx.

↳ ans = length − LPS of whole string

8 − 5 = 3

T.C: O(2N) ≈ O(N)

→ String s

s = "Hello"

s = s + "e" → O(N)

→ If add N strings one by one.

↳ O(N²)

ex: String s = " "

```
for (int i=0; i<n; i++){
    s = s + "1";
}
```

→ O(N²)

→ StringBuilder

↳ StringBuilder sb = new StringBuilder();

```
sb = "Hello";
sb.append("e");
```

→ O(1)

↳ String s = "Hello";

↳ StringBuilder sb = new StringBuilder(s);

```
for (int i=0; i<N; i++){
    sb.append("1");
}
String s1 = sb.toString();
```

O(N) O(N)

# Pattern Searching

## Java Code:

```
class Solution {
    int search(String text, String pat) {
        String s1 = pat+"#"+text;
        int[] lps = LPS(s1);
        for(int i = 0; i<s1.length(); i++){
            if(lps[i] == pat.length()) return 1;
        }
        return 0;
    }
    public int[] LPS(String s){
        int n = s.length();
        int[] lps = new int[n];
        for(int i = 1; i<n; i++){
            int len = lps[i-1];
            if(s.charAt(i) == s.charAt(len)){
                lps[i] = len +1;
            } else {

                while(s.charAt(i) != s.charAt(len)){
                    if(len == 0){
                        len = -1;
                        break;
                    }
                    len = lps[len - 1];
                }
                lps[i] = len+1;
            }


        }
        return lps;
    }
};
```

## C++ Code:

```
#include <string>
#include <vector>
```

```
class Solution {
public:
    int search(std::string text, std::string pat) {
        std::string s1 = pat + "#" + text;
        std::vector<int> lps = LPS(s1);
        for (int i = 0; i < s1.length(); i++) {
            if (lps[i] == pat.length()) return 1;
        }
        return 0;
    }

    std::vector<int> LPS(std::string s) {
        int n = s.length();
        std::vector<int> lps(n);
        for (int i = 1; i < n; i++) {
            int len = lps[i - 1];
            if (s[i] == s[len]) {
                lps[i] = len + 1;
            } else {
                while (s[i] != s[len]) {
                    if (len == 0) {
                        len = -1;
                        break;
                    }
                    len = lps[len - 1];
                }
                lps[i] = len + 1;
            }
        }
        return lps;
    }
};
```

## Python Code:

```
class Solution:
    def search(self, text: str, pat: str) -> int:
        s1 = pat + "#" + text
        lps = self.LPS(s1)
        for i in range(len(s1)):
            if lps[i] == len(pat):
                return 1
        return 0

    def LPS(self, s: str) -> list:
        n = len(s)
```

```
    lps = [0] * n
    for i in range(1, n):
        len_ = lps[i - 1]
        if s[i] == s[len_]:
            lps[i] = len_ + 1
        else:
            while s[i] != s[len_]:
                if len_ == 0:
                    len_ = -1
                    break
                len_ = lps[len_ - 1]
            lps[i] = len_ + 1
    return lps
```

# Minimum characters to be added at front to make string palindrome

## Java Code:

```
class Solution {
    public static int minChar(String str) {
        StringBuilder sb = new StringBuilder(str);
        sb.append("#");
        for(int i = str.length()-1; i>=0; i--){
            sb.append(str.charAt(i));
        }
        String s1 = sb.toString();
        int lps = LPS(s1);

        return str.length()-lps;
    }
     public static int LPS(String s){
        int n = s.length();
        int[] lps = new int[n];
        for(int i = 1; i<n; i++){
            int len = lps[i-1];
            if(s.charAt(i) == s.charAt(len)){
                lps[i] = len +1;
            } else {

                while(s.charAt(i) != s.charAt(len)){
                    if(len == 0){
```

```
                    len = -1;
                    break;
                }
                len = lps[len - 1];
            }
            lps[i] = len+1;
        }


    }
    return lps[n-1];
  }
}
```

## C++ Code:

```
#include <string>
#include <vector>

class Solution {
public:
    static int minChar(std::string str) {
        std::string s1 = str + "#";
        for (int i = str.length() - 1; i >= 0; i--) {
            s1 += str[i];
        }
        int lps = LPS(s1);
        return str.length() - lps;
    }

    static int LPS(std::string s) {
        int n = s.length();
        std::vector<int> lps(n);
        for (int i = 1; i < n; i++) {
            int len = lps[i - 1];
            if (s[i] == s[len]) {
                lps[i] = len + 1;
            } else {
                while (s[i] != s[len]) {
                    if (len == 0) {
                        len = -1;
                        break;
                    }
                    len = lps[len - 1];
                }
                lps[i] = len + 1;
            }
```

```
        }
        return lps[n - 1];
    }
};
```

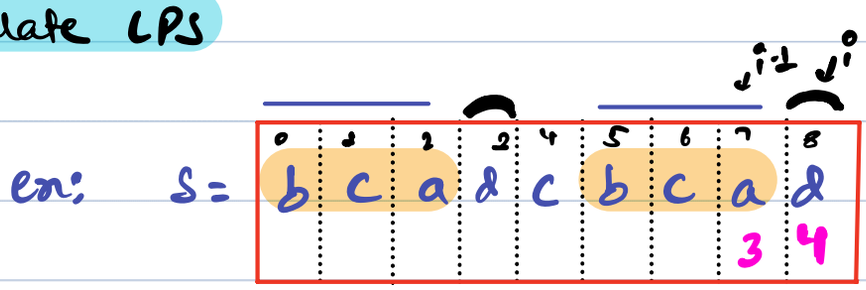
## Python Code:

```
class Solution:
    @staticmethod
    def minChar(str_: str) -> int:
        s1 = str_ + "#"
        for i in range(len(str_) - 1, -1, -1):
            s1 += str_[i]
        lps = Solution.LPS(s1)
        return len(str_) - lps

    @staticmethod
    def LPS(s: str) -> int:
        n = len(s)
        lps = [0] * n
        for i in range(1, n):
            len_ = lps[i - 1]
            if s[i] == s[len_]:
                lps[i] = len_ + 1
            else:
                while s[i] != s[len_]:
                    if len_ == 0:
                        len_ = -1
                        break
                    len_ = lps[len_ - 1]
                lps[i] = len_ + 1
        return lps[-1]
```

Today's agenda

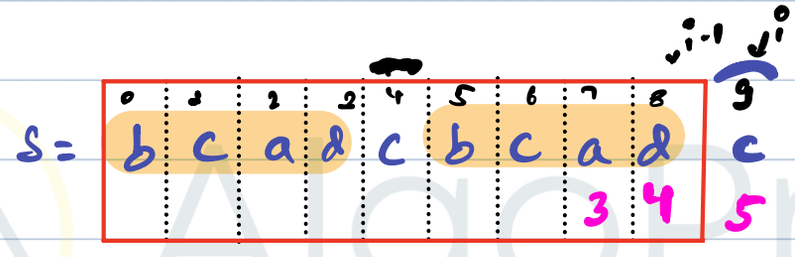
- LPS[] construction
- multiply 2 strings

↳ Calculate LPS



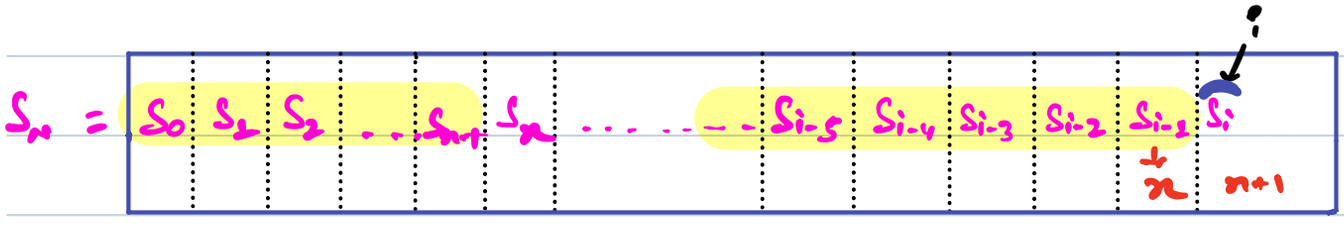

Obs1: The LPS value (i+1)th index can never be more than 1 compared to ith index.

↳ The minimum LPS value can be 0.






// General




```
if (Si == Sn){
    LPS[i] = LPS[i-1] + 1;
}
```

(Si: i; Sn: n; LPS[i-1]: n)

4

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S = | c | a | c | y | c | a | c | a | b | c | a | c | y | c | a | c | y |
| | 0 | 0 | 1 | 0 | 1 | 2 | 3 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

i = 16 len = 7

| ans | S[i] == S[len] |
|---|---|
| 8 | S[16] == S[7] → not same → LPS[i] ≠ 8 ↳ len = 3 |
| 4 | S[16] == S[3] ↳ matches → LPS[i] = 4 |

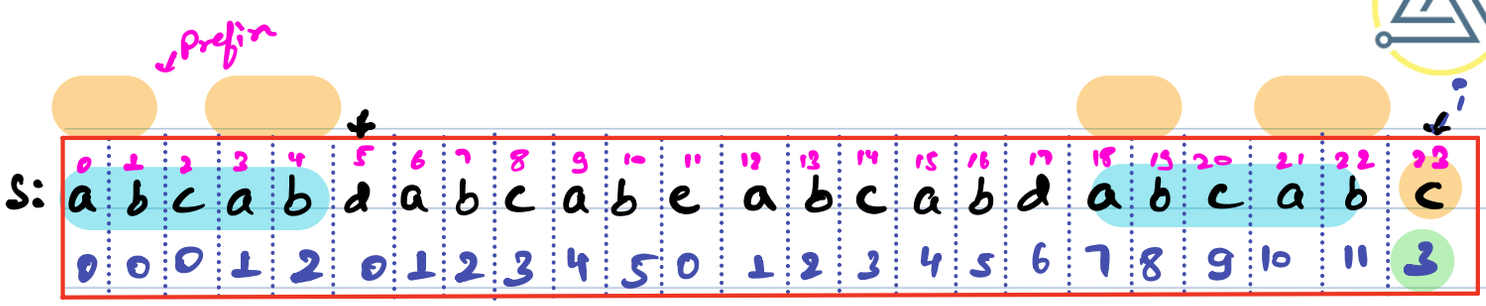
8 length x x

7 length

6 length

1 length

0 length




i = 23 len = 11

| len | ans | S[i] == S[len] |
|---|---|---|
| 11 | 12 | S[23] == S[11] → Not matching |
| 5 | 6 | S[23] == S[5] → Not matching |
| 2 | 3 | S[23] == S[2] → matching ⇝ LPS[i] = 3 |

## //Pseudo Code

```
int[] LPS (String s(N)){
    int LPS[N];

    LPS[0] = 0;

    for(int i=1; i<N; i++){
        int len = LPS[i-1];
        if (s[i] == s[len]){
            LPS[i] = len+1;
        }
        else{
            while (s[i] != s[len]){
                if(len == 0){ len = -1; break;}
                len = LPS[len-1];
            }
            LPS[i] = len+1;
        }
    }

    return LPS;
}
```

T.C: O(N)

S.C: O(1)/O(N)

```
for (int i=0; i<N; i++) {
    for (int j=0; j<i; j++) {
        -> O(N^2)
    }
}
```

| i | j | count |
|---|---|---|
| 0 | -- | 0 |
| 1 | [0,0] | 1 |
| 2 | [0,1] | 2 |
| 3 | [0,2] | 3 |
| | | . |
| N-1 | [0, N-2] | N-1 |

$\frac{N*(N-1)}{2} = O(N^2)$

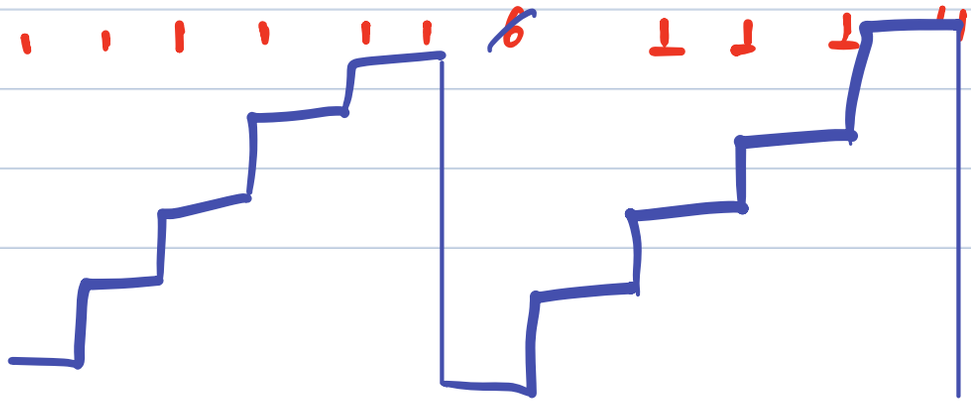
---

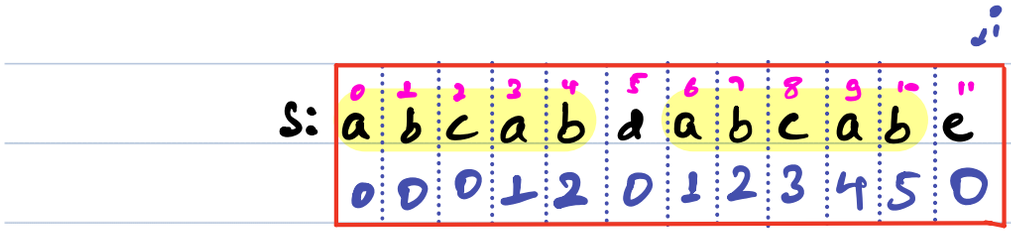
```
for(int i=1; i<N; i++) {
    int len = LPS[i-1];
    if (s[i] == s[len]) {
        LPS[i] = len+1;
    }
    else {
        while (s[i] != s[len]) {
            if(len == 0) { len = -1; break; }
            len = LPS[len-1];
        }
        LPS[i] = len+1;
    }
}
```

| i | j | count |
|---|---|---|
| 0 | | 0 |
| 1 | | 1 |
| 2 | | 1/2 |
| 3 | | 1/2/3 |
| | | . |
| | | . |
| | | . |
| | | . |

LPS → 0 1 2 3 4 5 [0,0] 1 2 3 [0,4] ⇒ O(2N)

count → 1 1 1 1 1 1 6 1 1 1 1






```
for(int i=1; i<N; i++) {
    int len = LPS[i-1];
    if (s[i] == s[len]) {
        LPS[i] = len+1;
    }
    else {
        while (s[i] != s[len]) {
            if(len == 0) { len = -1; break; }
            len = LPS[len-1];
        }
        LPS[i] = len+1;
    }
}
```

| i | len |
|---|---|
| 1 | 0 → -1 |
| 2 | 0 → -1 |
| 3 | 0 |
| 4 | 1 |
| 5 | 2 → 0 → -1 |
| 6 | 0 |
| 7 | 1 |
| 8 | 2 |
| 9 | 3 |
| 10 | 4 |
| 11 | 5 → 2 → 0 → -1 |