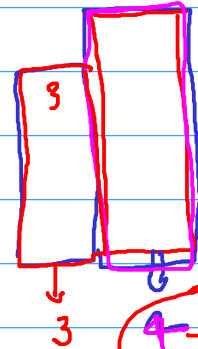
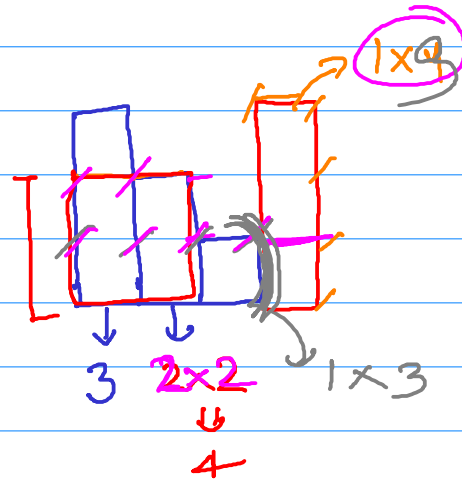
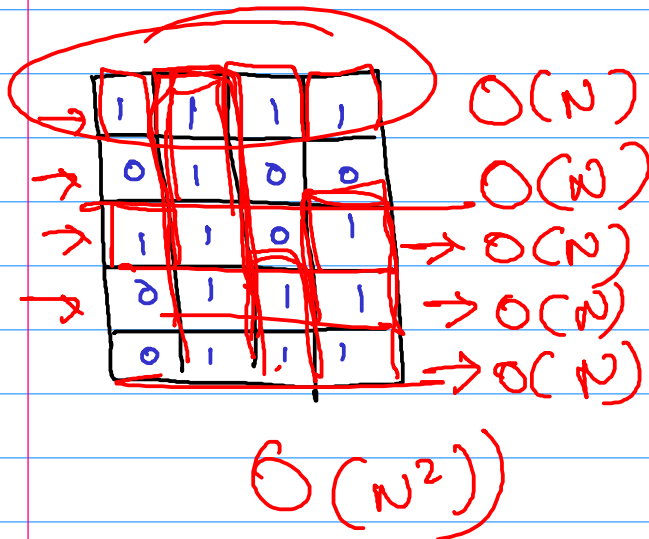


$1-0=1$

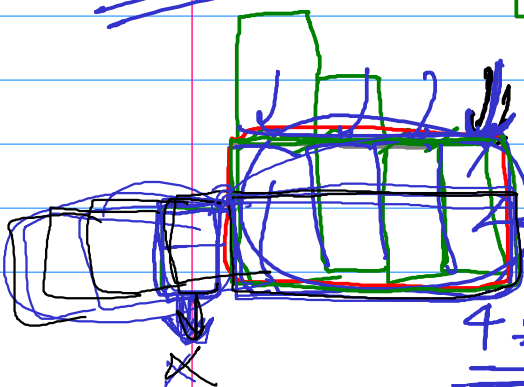
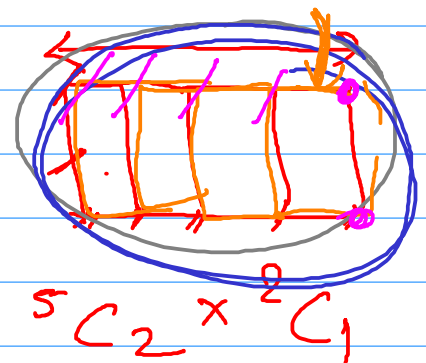
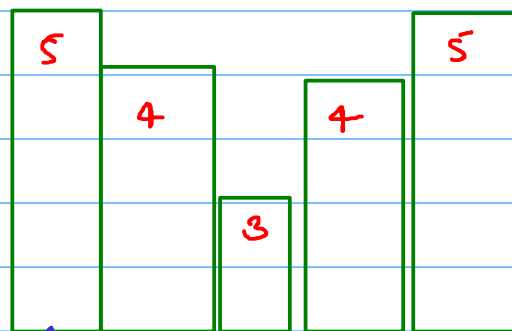
$4 \times 1 + 6 = 10$



$dp(i)$: i^{th} index end
No. of rectangles.



Prev small stack



$4 \times a(i) + x$

$4C_1$

$5C_2 \times 2C_1$

$\left(a(i) \times \frac{n(n+1)}{2} \right)$

STRING

KMP -

text: abababcababababab

patl: "abcd"

```
for (int i=0; i < text.size(); i++) {  
    bool flag = true;  
    for (int j=0; j < patl.size(); j++) {  
        if (text[i+j] != patl[j])  
            { flag = false;  
              break; }  
    }  
    if (flag) return true;  
}  
return false;
```

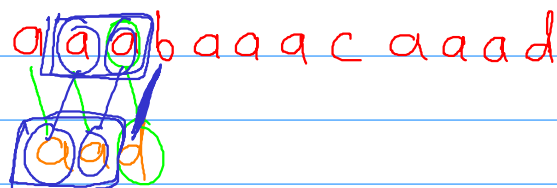
text: N
patl: M

$O(N * M)$;
Space $O(1)$.

KMP

↓
LPS Array / π -Array

aabaaacaaad



LPS[i]: Longest proper prefix which is also proper suffix ending at index i.

proper prefix

"ABCD"

A
AB
ABC

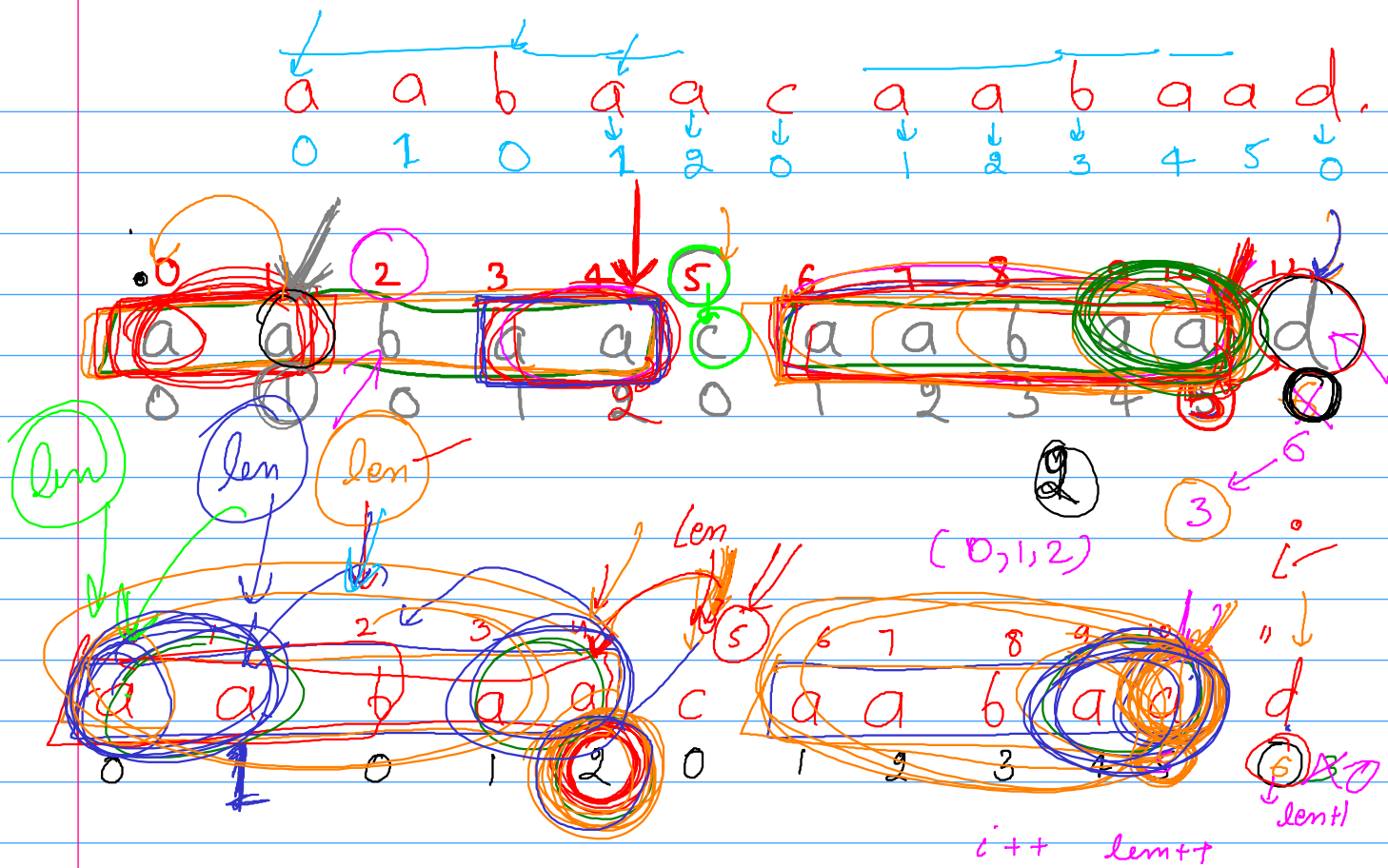
ABCD

proper suffix

"ABCD"

D
CD
BCD

ABCD



```

len = 0, i = 1
while (i < n)
{
    if (s[i] == s[len])
        lps[i] = len + 1;
        i++, len++;
    }
    else {
        if (len > 0) {
            len = lps[len - i];
        }
        else {
            lps[i] = 0;
        }
        i++;
    }
}

```

$O(n)$

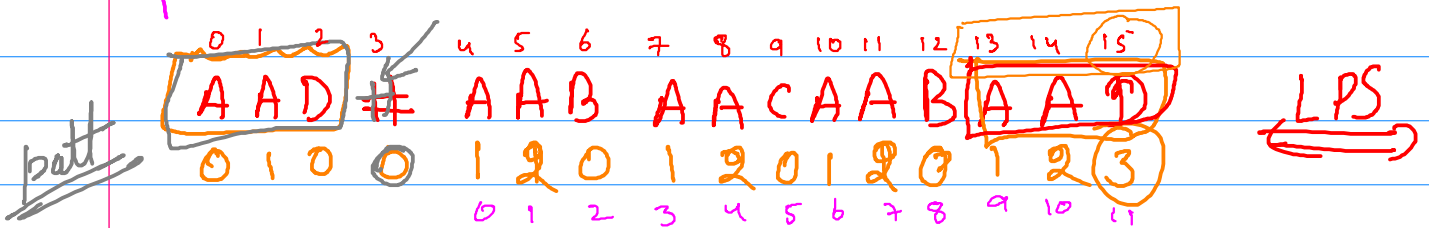
LPS

KMP

text: A A B A A C A A B A A D

pat: A A D

pat + "#" + text



pat AB AB

ABCD
AB#ABCD
001201200

15-2*7
15-6

Diagram illustrating the LPS array calculation for the concatenated string "ABAB#ABAB".

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
String	A	B	A	B	#	A	B	A	B	A	B	A	B	A	B	A
LPS	0	0	1	2	0	0	1	2	0	0	1	2	0	0	1	2

Arrows indicate the calculation of LPS values for indices 13, 14, and 15, showing a jump to index 12.

text i
A B A C

```
1 class Solution {
2 public:
3     int strStr(string S, string P) {
4         int m = P.size();
5         cout << m << endl;
6         P = P + "#" + S;
7         int n = P.size();
8         int lps[n];
9         lps[0] = 0;
10        int len = 0, i = 1;
11        while(i < n){
12            if(P[i] == P[len]){
13                lps[i] = ++len;
14                if(lps[i] == m) return i - 2*m;
15                i++;
16            }
17            else{
18                if(len > 0) len = lps[len - 1];
19                else lps[i++] = 0;
20            }
21        }
22        // cout << P << endl;
23        // for(int i = 0; i < n ; i++){
24        //     if(lps[i] == m) return i - 2*m;
25        // }
26        return -1;
27    }
28 }
```

Longest Palindrome in a String

Medium Accuracy: 49.2% Submissions: 57044 Points: 4

Given a string S , find the longest palindromic substring in S . **Substring of string S :** $S[i \dots j]$ where $0 \leq i \leq j < \text{len}(S)$. **Palindrome string:** A string which reads the same backwards. More formally, S is palindrome if $\text{reverse}(S) = S$. **Incase of conflict**, return the substring which occurs first (with the least starting index).

Example 1:

Input:
 $S = \text{"aaaabbaa"}$
Output: aabbaa
Explanation: The longest Palindromic substring is "aabbaa".

```
1 // } Driver Code Ends
2 class Solution {
3 public:
4     int idx, len = -1;
5     void help(int c1, int c2, string& s){
6         while(c1 >= 0 and c2 < s.size() and s[c1] == s[c2]){
7             int l = c2 - c1 + 1;
8             if(l > len){
9                 len = l;
10                idx = c1;
11            }
12            c1--, c2++;
13        }
14    }
15    string longestPalin (string s) {
16        for(int i = 0; i < s.size(); i++){
17            help(i, i, s);
18            help(i, i + 1, s);
19        }
20        return s.substr(idx, len);
21    }
22 };
23 // } Driver Code Ends
```

$O(N)$
Time: $O(N \cdot 2n)$
 $\approx O(N^2)$
Space: $O(1)$.

6
a a a a b b a a

Q

$S =$

$\text{rev}(S)$

Longest common sub sequence.

a b c d c b a .

680. Valid Palindrome II

Easy 5595 301 Add to List Share

Given a string s , return `true` if the s can be palindrome after deleting **at most one** character from it.

Example 1:

Input: $s = \text{"aba"}$
Output: true

Example 2:

Input: $s = \text{"abca"}$
Output: true
Explanation: You could delete the character 'c'.

$C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7$
 $O(2N)$

1177. Can Make Palindrome from Substring

Medium 531 225 Add to List Share

You are given a string `s` and array `queries` where `queries[i] = [lefti, righti, ki]`. We may rearrange the substring `s[lefti...righti]` for each query and then choose up to `ki` of them to replace with any lowercase English letter.

If the substring is possible to be a palindrome string after the operations above, the result of the query is `true`. Otherwise, the result is `false`.

Return a boolean array `answer` where `answer[i]` is the result of the `ith` query `queries[i]`.

Note that each letter is counted individually for replacement, so if, for example `s[lefti...righti] = "aaa"`, and `ki = 2`, we can only replace two of the letters. Also, note that no query modifies the initial string `s`.

Example :

Input: `s = "abcda"`, `queries = [[3,3,0],[1,2,0],[0,3,1],[0,3,2],[0,4,1]]`

Output: `[true,false,false,true,true]`

Explanation:

`queries[0]`: substring = "d", is palidrome.

`queries[1]`: substring = "bc", is not palidrome.

`queries[2]`: substring = "abcd", is not palidrome after replacing only 1 character.

`queries[3]`: substring = "abcd", could be changed to "abba" which is palidrome. Also this can be changed to "baab" first rearrange it "bacd" then replace "cd" with "ab".

`queries[4]`: substring = "abcda", could be changed to "abcba" which is palidrome.

$k=2$ odd=5
~~a~~ b c d e
 e d
 (5) <= k

left_i

right_i

precon [i] = { a b c d e ... z }

(d, z) "a b c d ... z"

(d-1) - - - - -

(2k+1)

odd = X

k=2

k

$\frac{x}{2} \leq k$

$\frac{5}{2} \leq 2$

5 =>

(2k+1) = oddcount

Now, you are given a non-empty string **S**, containing only lowercase English letters. The given string may or may not be palindrome. Your task is to make it a palindrome. But you are only allowed to add characters at the right side of the string. And of course, you can add any character you want, but the resulting string has to be a palindrome, and the length of the palindrome should be as small as possible.

For example, the string is "bababa", you can make many palindromes including:

- "bababababab"
- "babababab"
- "bababab"

Since we want a palindrome with minimum length, the solution would be **"bababab"** cause its length is minimum.

Input

Input starts with an integer T (≤ 10), denoting the number of test cases.

Each case starts with a line containing a string **S**. You can assume that $1 \leq \text{length}(\mathbf{S}) \leq 10^6$.

Output

Sample

Input	Output
4 bababababa pqrs	Case 1: 11 Case 2: 7 Case 3: 11

$$\left\{ \begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix} \right\}$$



$\begin{matrix} \cancel{1} & \cancel{0} & \cancel{1} & 0 & \cancel{1} & 0 & \cancel{1} & \cancel{1} \\ \cancel{1} & \cancel{0} & \cancel{1} & 1 & \cancel{1} & 1 & \cancel{1} & \cancel{1} \end{matrix} \Rightarrow \begin{Bmatrix} 0 & 0 \\ 1 & 1 \end{Bmatrix}$

$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$

$(x, y) \in$
 $10 \leq x$
 $0.1 = y$

✓ $\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 2 & 0 \end{pmatrix}$

$$0.1 = y$$
$$\left(\frac{x}{2} + \frac{y}{2} + \frac{z}{2} \right) = 1$$


 $\Rightarrow 10 \Rightarrow$


A hand-drawn diagram of a cell. It features a large, irregular outer boundary representing the cell membrane. Inside, there is a smaller, roughly circular nucleus with a central dot representing the nucleolus. A large, clear, oval-shaped vacuole is positioned on the right side of the cell. The drawing is done in black ink on a white background.

A handwritten number 0 is shown on lined paper. The number is formed by a single continuous stroke, starting from the top line, curving around to the left and bottom, and then curving back up to the top line. A vertical stroke is drawn from the bottom line to the middle line, intersecting the curve of the 0.

even

```

graph TD
    A((2m)) --> B((m))
    A --> C((m))
  
```

Diagram illustrating two sets, X and Y , each containing a subset. Set X contains a subset with one element. Set Y contains a subset with two elements.



(13)

↓

S + solo

S

↓

G + Isolo

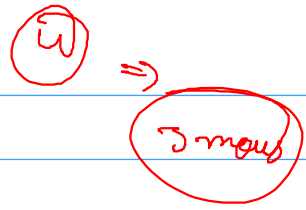
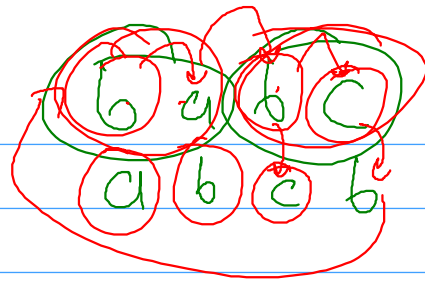
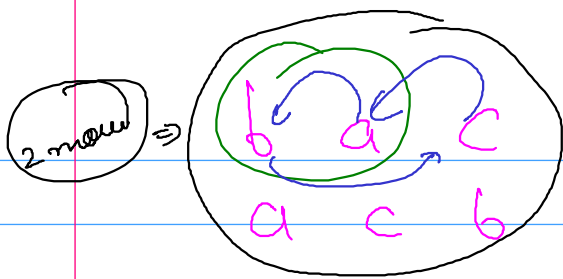
A hand-drawn diagram of a set S . It consists of a large, irregular oval shape. Inside this oval, there are two smaller circles, one above the other. Each of these smaller circles contains the number '0'. To the left of each '0' is the number '1'. This represents the set $S = \{1, 0\}$.

$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow ?$

Hand-drawn diagram illustrating the addition of two binary numbers:

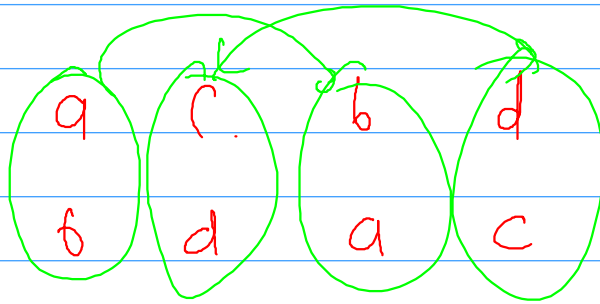
- Top row: 1100 and 1011
- Bottom row: 10111

The numbers are written in circles, and the result is written in a red circle.

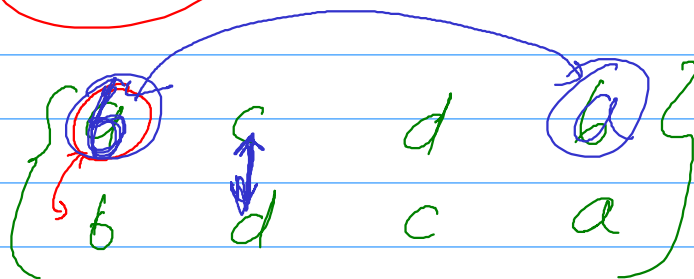
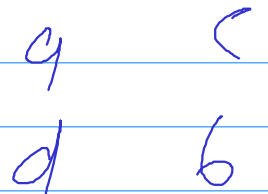
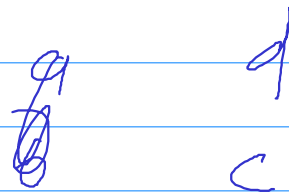
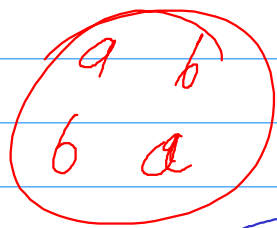
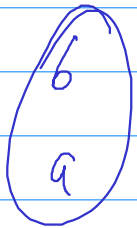


a b c a 5 4 3 2 1

b c d d 1, 2, 3, 4, 5, 6, 7, 8 - - 20 21

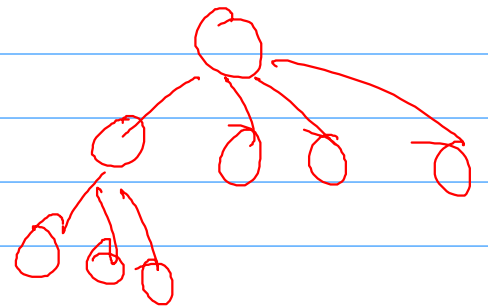


a b c d e
d c a ~~e~~ b



b

f(i, s, t)
if(i == s.size()) return 0;
for(it(s[i] == f(i))
f(i+1, s, t);



for(int j = i+1; j < n; j++)

if(s[j] == t(i))

swap(s[i], s[j]);

j = i+1; ans = min(ans, 1 + f(i+1, s, t));

return

9.

1 2 3 4 5 6 7 8 9
 (15, 14, 11, 12, 13, 16, 17, 19, 18)
 5 4 1 2 3 6 7 9 8

(15, 0) (14, 1) (11, 2) (12, 3) (13, 4)

+ 1 swap (16, 5) (17, 6)

+ 0 swap + 1 swap (19, 7) (18, 8)

