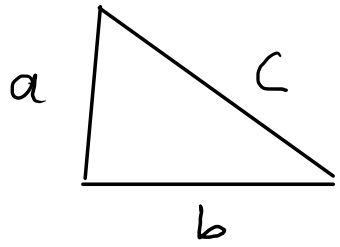
(1) Is it a triangle?



→ sum of length of two sides > length of third side.

a = 4, b = 5, c = 13

→ if (a+b > c &&
b+c > a &&
c+a > b)
{
print "yes"
}

→ else
{
print "no"
}

(2) Rotate the Array Returns back:-

n = 6, k = 1

A[] → | 4 | 5 | 1 | 6 | 8 | 7 |

k = 6

If k = n, then after rotation the array is going to be the same.

1 time → | 7 | 4 | 5 | 1 | 6 | 8 |

k = 1

2 times → | 8 | 7 | 4 | 5 | 1 | 6 |

k = 2

3 times → | 6 | 8 | 7 | 4 | 5 | 1 |

k = 3

k = 6 → no rotⁿ.

rotⁿ = k % n

7 % 6 = 1

6 % 6 = 0

If k = 7 . . . .

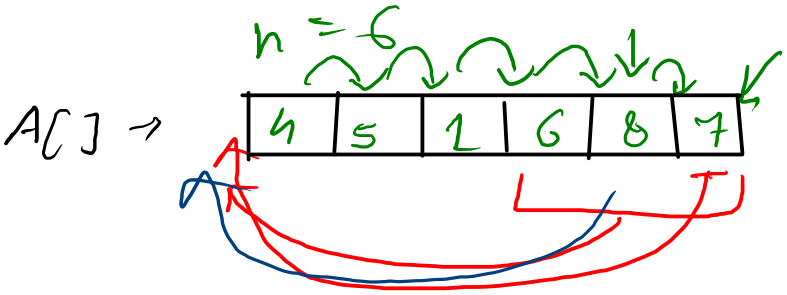
| 4 | 5 | 1 | 6 | 8 | 7 |

7 times rotated.

| 7 | 4 | 5 | 1 | 6 | 8 |

k = 8

rotⁿ = 8 % 6 = 2



$k = 3$

6	8	7	4	5	1

Brute force

$rot^n = k \% n$

$= 3 \% 6$

$= 3$

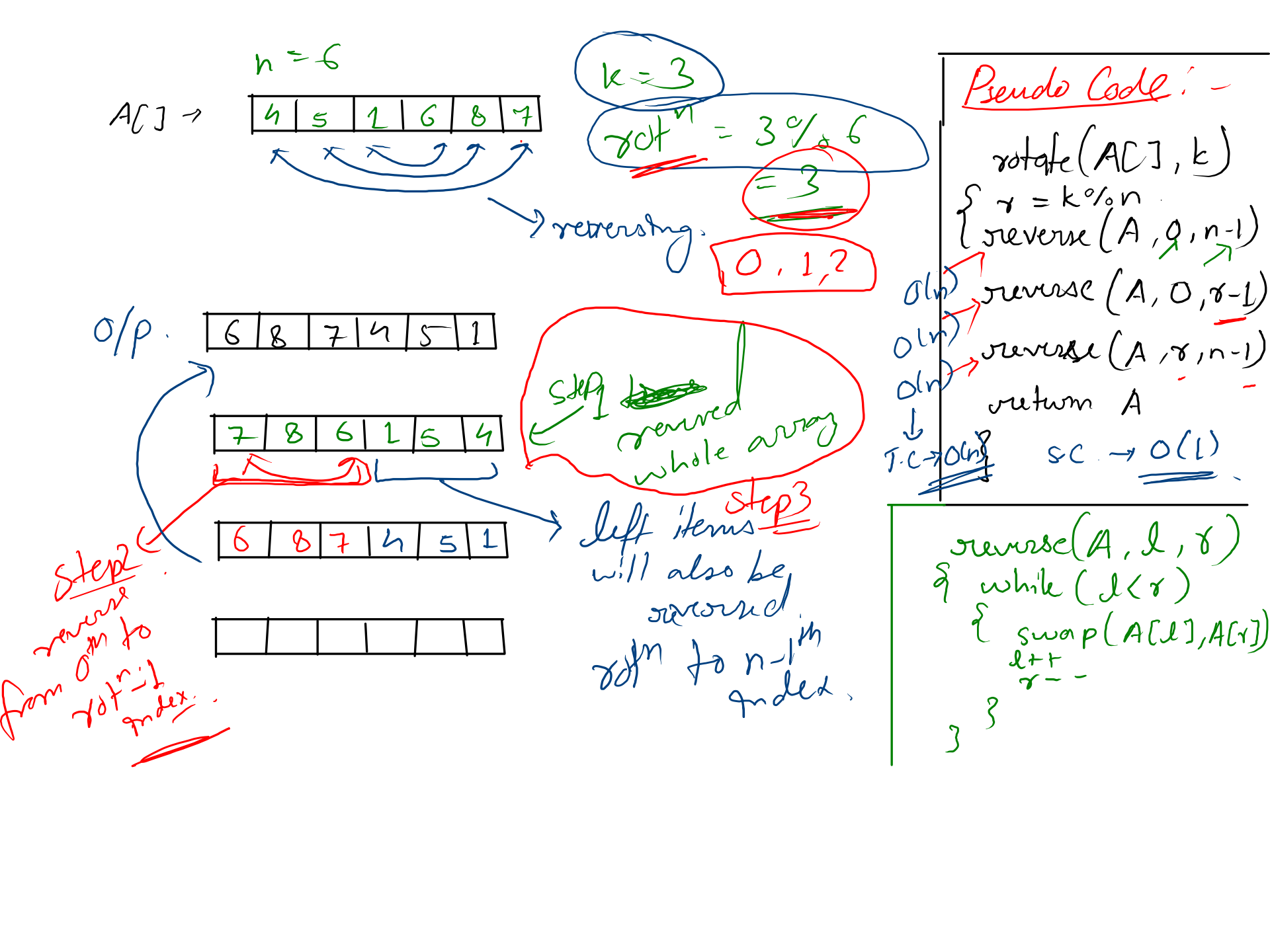
# of rot → `for(i=0; i<rot^n; i++)`

```
{ temp = A[n-1]   // 7
  for(j=n-2; j>=0; j--)
      A[j+1] = A[j];
      to shift items by 1 position
}
A[0] = temp
```

$O(k * n)$

$k \simeq n$

$\simeq O(n^2)$ → $k \simeq n-1$


n = 6
A[] →
4 5 1 6 8 7
k = 3
rot^n = 3 % 6
= 3
reversing.
0, 1, 2
O/p.
6 8 7 4 5 1
7 8 6 1 5 4
6 8 7 4 5 1
step1 reversed whole array
step2 reverse from 0th to rot^n-1 index.
step3 left items will also be reversed rot^n to n-1th index.
Pseudo Code:-
rotate(A[], k)
{ r = k%n
reverse(A, 0, n-1)
reverse(A, 0, r-1)
reverse(A, r, n-1)
return A
O(n)
O(n)
O(n)
T.C → O(n)
SC → O(1)
reverse(A, l, r)
{ while(l<r)
{ swap(A[l], A[r])
l++
r--
}
}

Break ;
back at
10 : 26 am

(3) Max Length Substring Distinct :-

① calculate # of distinct characters in the i/p string

Step 1 → Generate Substrings.

↳ step2.

↳ find out max # of distinct chars.

↳ if maximum then remember it by its length.

step3:

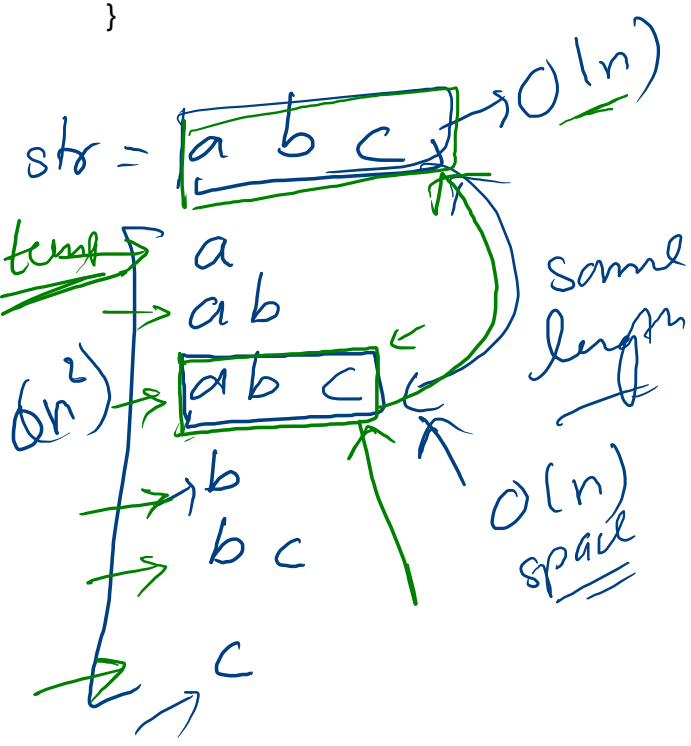
if true then

if length < max then

update max.

calculate the no. of distinct characters.

```
str = "aaabbbcdcd"
obj = []

for(i = 0; i < str.length; i++) {
        if(obj[str[i]] == undefined) {
                obj[str[i]] = 1
        }
}
```



obj

a	→ 1
b	→ 1
c	→ 1
d	→ 1

# of distincts = obj.length !

In Java use
↳ HashMap
↳ HashSet
↑ don't put redundant items.

```
str = "aaabbbcdcd"

obj = []  //number of distinct characters in the i/p string

for(i = 0; i < str.length; i++) {
        if(obj[str[i]] == undefined) {
                obj[str[i]] = 1
        }
}

//start generating substrings
min = INT_MAX
for(i = 0; i < str.length; i++) {
        temp = ""
        for(j = i; j < str.length; j++) {
                temp += str[i]

                //calculate the number of distinct characters in the substring

                obj2 = []

                for(k = 0; k < temp.length; k++) {
                        if(obj2[temp[i]] == undefined) {
                                obj2[temp[i]] = 1
                        }
                }

                //check if the substring have max # of distinct char
                if(obj2.length == obj.length) {
                        if(min > temp.length) {
                                min = temp.length
                        }
                }
        }
}

print(min)
```

aaabbbcdcd

T.C → O(n³)

S.C → O(n)

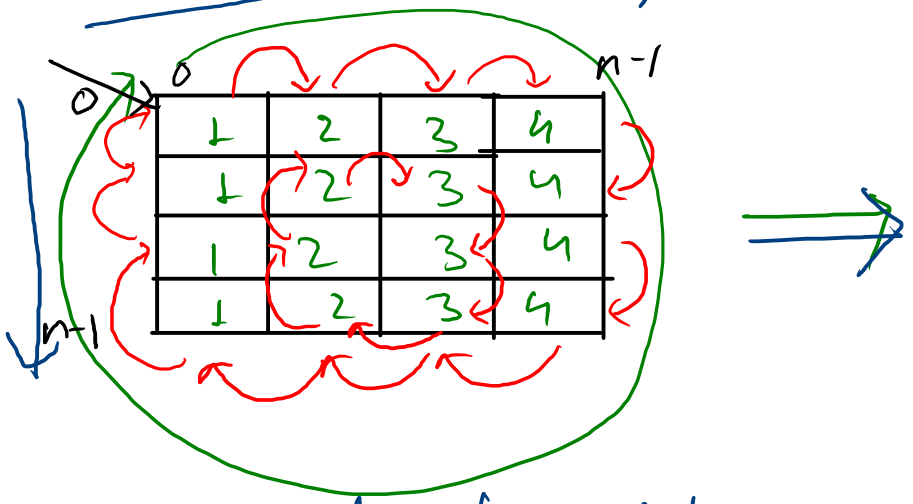
H.W

reduce it by not generating useless substrings.

→ O(n²)

n² * n = n³

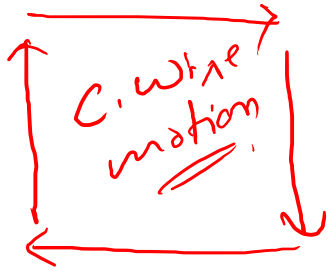
(4) Rotate Elements :-



1	1	2	3
1	2	2	4
1	3	3	4
2	3	4	4

in order to rotate as a cycle.
4 motions are happening.

1. left to right
2. top to bottom
3. right to left
4. bottom to top



matrix size = n x n

left = 0

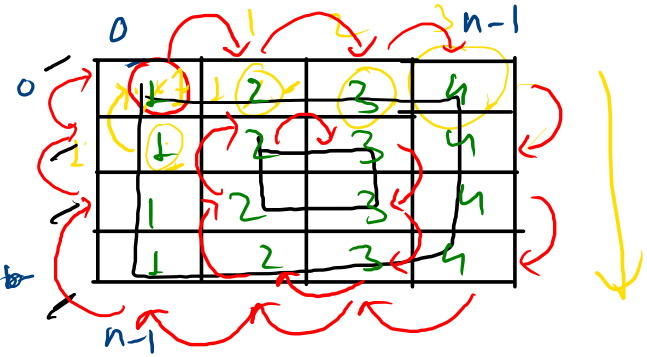
right = n-1

top = 0

bottom = n-1

1. left to right
2. top to bottom
3. right to left
4. bottom to top

```
for(i = left; i<= right; i++){
        temp = A[top][i]
        A[top][i] = prev
        prev = temp
}
top++
for(j = top; j <= bottom; j++){

}

for(k = right; k >= left; k--){

}

for(m = bottom; m >= top; m--){

}
```



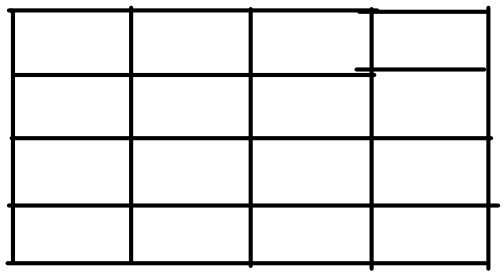
1	1	2	3
1	2	2	4
1	3	3	4
2	3	4	4

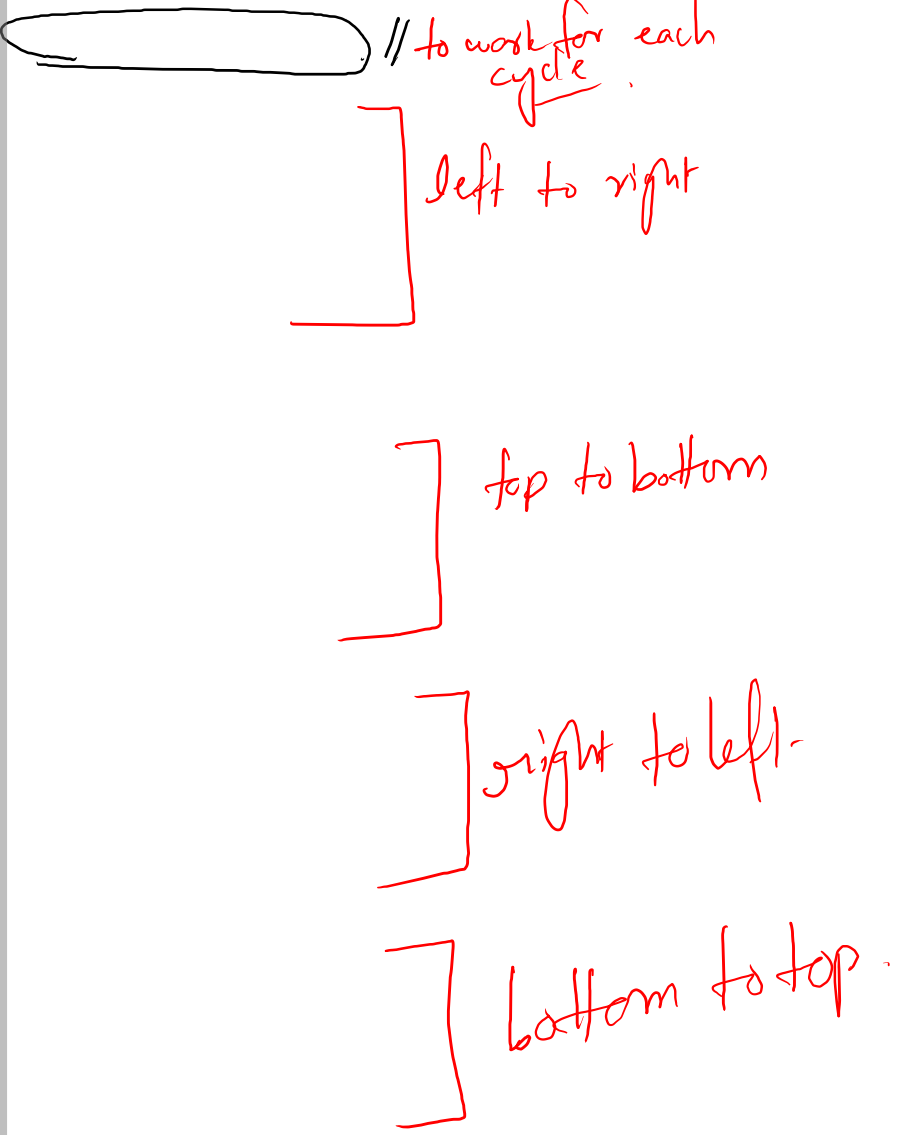
p = 1
i = 0
t = 1
p = 1

p = 1
i = 1
t = 2
p = 2

p = 2
i = 2
t = 3
p = 3

p = 3
i = 3
t = 4
p = 4




// to work for each cycle.
left to right
top to bottom
right to left.
bottom to top.