class 3: Arrays

pow(x,n) x^n

$(2,3) = 2^3 = 8$

Brute force 1 ... n

$x * x * \dots$ n times $= O(n)$

If n is -ve $= x \to \frac{1}{x} \times \dots$ n times

$2^{-3} = \left(\frac{1}{2}\right)^3$

2nd approach: $x^n \equiv (x^2)^{n/2}$

$2^4 \equiv (2^2)^{4/2}$

$2^3 \equiv 2 \cdot (2^2)^{3/2} \equiv 2 \cdot (2^2)^1 \equiv 2^3$

If even : $(x^2)^{n/2}$

Odd : $x * (x^2)^{n/2}$

$3^7 \equiv 3 \cdot (3^2)^{7/2} \equiv 3 \cdot (3^2)^3$

$(-3)^5 = (-3) * ((-3)^2)^{5/2}$

$= (-3) * (-3)^{2 \times 2}$

$= (-3)^5$

```
Pow (x,n)
{
   if (n==0)
      return 1;
   if (n<0)
   {
      n = -n
      x = 1/x
   }
   return (n%2==0) ? pow(x*x, n/2) : x * pow(x*x, n/2);
}
```

TC: $n \quad \frac{n}{2} \quad \frac{n}{4} \dots \dots 1$

$\Rightarrow \frac{n}{2^k} = 1$

$n = 2^k$

$k = \log n$

$O(\log n)$

SC: $O(\log n)$

Q2 Next permutation

① 1,2,3 ≡ 1,3,2

② 7,5,8,3 ≡ 7,8,3,5

7,5,3,8 ~~7853~~

7 8 5 3
7 8 3 5

③ 1,3,8,3,5,7 ≡ 1,3,8,3,7,5

1 3 8 3 7 5

④ {3,2,1} ≡ {1,2,3}

$i^{th} < (i+1)^{th}$ position

1,5,8,4,5,6,3,7,1 ≡ 1,5,8,4,5,6,3,7,4,1

1 5 8 4 5 6 7 3 1 ≡ 1,5,8,4,5,6,4,1,3,1

1 5 8 4 5 6 7 1 3 1,5,8,4,5,6,4,1,3,7

[1,2,3] = 132

Algorithm:

1) we need to find first pair of 2 successive numbers a[i] & a[i-1] such that a[i] > a[i-1]

2) from the end try to find the number which is just greater than $(i-1)^{th}$ number let it be j

swap those numbers

3) reverse the array from position i

TC: O(n)

SC: O(1)

Rotation 1

1,2,3 → 3,1,2 ← Rotation 2

2,3,1

Q3: Rotated sorted array

0	1	2	3	4	5
11	15	6	8	9	10

sum = 16

2 pointers l, r

$15+6 = 21$

$11+6 = 17$

$6+10 = 16$ True

$l \equiv (l+1) \bmod N$

$r \equiv (r-1+N) \bmod N$

$(0-1+6) \bmod 6$ % → remainder

$= 5 \bmod 6 \equiv 5$

Algorithm:

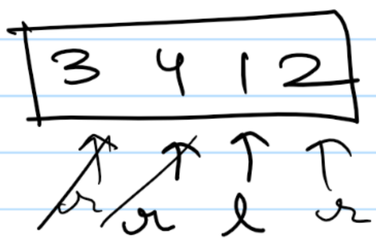
1) we will run a loop from 0 to N-1 to find pivot points
2) left pointer will point to the smallest element & right pointer will point to largest ".
3) while (l != r), we'll keep checking if arr[l] + arr[r] = sum ?
   a) if arr[l] + arr[r] > sum, update r = (N+1-r)% N
   b) if < sum l = (l+1) % N
   c) if = sum return True

TC: O(N)

SC: O(1)

sum = 3

1234 ≡ 4123 ≡ 3412



4+1 = 5

3+1 = 4

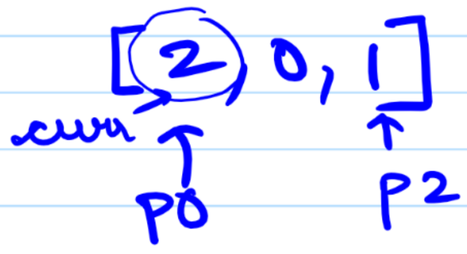
1+2 = 3

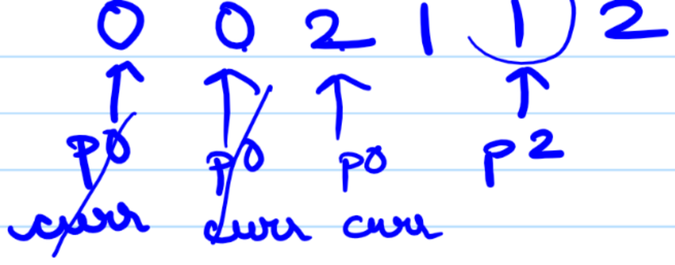
Q4: Dutch National Flag Problem

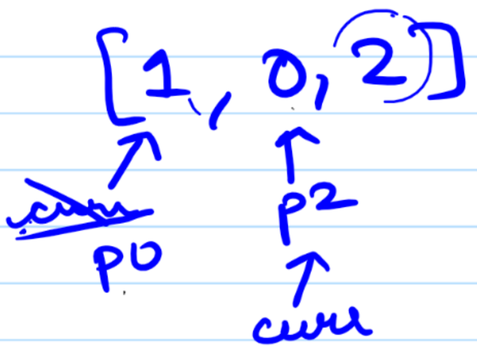
[2, 0, 2, 1, 1, 0]

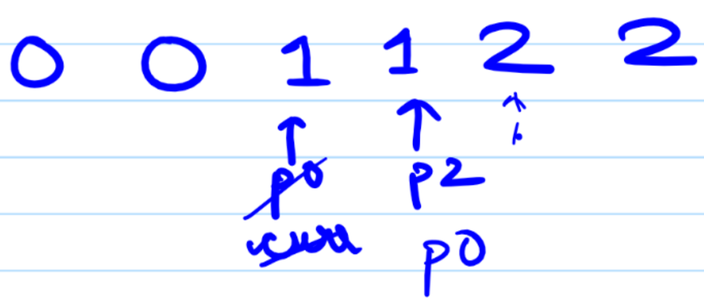
p0 p2

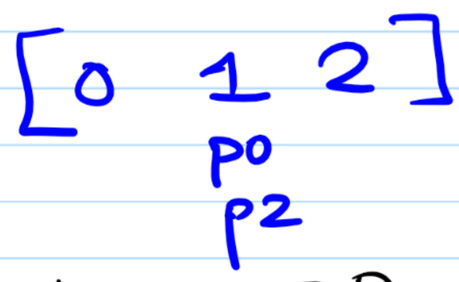
curr











Algorithm:

1) Initialise boundary of 0's by $p0 = 0$
2) Initialise " " 2's by $p2 = n-1$.
3) curr = 0
4) while curr <= p2
   a) if nums[curr] = 0, swap curr & p0 elements & move both pointers
   b) if nums[curr] = 2, swap curr & p2 element, move p2 to left
      $p2 = p2 - 1$
   c) if nums[curr] = 1, curr = curr + 1

TC: O(N) SC: O(1)

Q5: Rotate the array

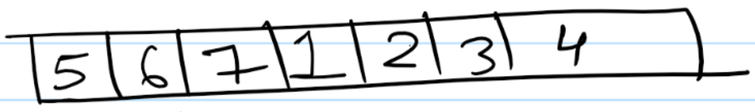
1,2,3,4,5,6,7 K=3

Brute force: 7 1 2 3 4 5 6 → O(n*k)

6 7 1 2 3 4 5

5 6 7 1 2 3 4

Using extra array



TC = O(n)

SC: O(n)

Without using extra space

1,2,3,4,5,6,7 $\xrightarrow{K=3}$ 5,6,7,1,2,3,4

Hint: Reverse?

1) Reverse all the numbers 7 6 5 4 3 2 1

2) Reverse 1st K numbers 5674321

3) Reverse numbers from K, n-1 5671234

TC: O(n)

SC: O(1)

4 = 4%3
= 1

3 times
123 = 123

1 = 312

6 times
= 123

4 = 312

9 9

Q6 Max consecutive 1's in an array

1, 0, 1, 1, 1, 0, 1, 1 = 3

count=0 1 0 1 2 3 0 1 2

Max=0 2 3

	1	0	1	1	0	1
co=0	1	0	1	2	0	1
Max=0	1			2		

Max = 2

Algorithm:

1) Maintain a counter for the number of 1's

2) Increment counter by 1, whenever we see 1

3) whenever we encounter 0,
update max
Reset counter = 0

4) Return max

TC: O(n)

SC: O(1)