Todays Content

1. First missing positive Integer

Q: Given ar[N], find first missing positive number / Natural Numbers.

Note: First the number not in arr[]

ar[ ] = {3 -2 1 2 7} : ans = 4

ar[ ] = {-9 2 6 4 -8 1 3} ans = 5

ar[ ] = {1 2 5 6 4 3} ans = 7

ar[ ] = {4 2 1 3} ans = 5

ar[ ] = {-4 8 3 -1 0} ans = 1

ar[ ] = {-8 -3 -1 -5} ans = 1

Idea: $ar[N] = \{a_0\ a_1\ a_2 \ldots a_{N-1}\}$

	If Not present	Present	
Search 1 :	* return 1	✓	1
Search 2 :	* return 2	✓	2
Search 3 :	* return 3	✓	3
⋮	;		i
Search N :	* return N	✓	N

Note: If all elements from 1..N are present 1st missing number is N+1

```
#Idea: for(int i=1; i<=N; i++){
           #Iterate on arr[] & search i
           if i is not present: return i;
       }   else goto next element
       return N+1;
```

TC: $O(N*N) = O(N^2)$ SC: $O(1)$

Idea2: Sort arr[]

```
for(int i=1; i<=N; i++){
    # Apply Binary Search in arr[] for i
    if i is not present: return i;
    else goto next element
}
return N+1;
```

TC: O(NlogN + N*logN) = O(NlogN)   SC: O(1)

↳ BS, N Times

↳ Sort

Idea3: Using sum of +ve elements*

arr[] = {1 2 3}   # Both have same sums

arr[] = {2 2 2}

Idea4: Insert all arr[] elements in hashset hs

```
for(int i=1; i<=N; i++){
    #Search i in hashset hs.
    if i is not present: return i;
    else goto next element
}
return N+1;
```

TC: O(N + N*1) = O(N)   SC: O(N) → Because hashset

Idea4:

Hint1: We are only searching 1..N, other numbers are irrelevant.

Hint2:

```
           0  1  2  3          0  1  2  3
arr[4] = { 4  3  1  2 }   →  { 1  2  3  4 }
    ind i  =  i+1 ele
    ele x  =  x-1 ind
```

ar[8] = {

0	1	2	3	4	5	6	7
~~4~~	2	~~7~~	~~6~~	9	~~1~~	~~8~~	~~3~~
~~6~~		~~8~~	4		6	7	8
1		3					

}

ind i = i+1 ele
ele x = x-1 ind

i	Correct data	Take data to correct position
0	ar[0] = 4	ar[0] = 4 ⇄ ar[3] #swap ar[0] & ar[3]
	ar[0] = 6	ar[0] = 6 ⇄ ar[5] #swap ar[0] & ar[5]
	ar[0] = 1	#goto next
1	ar[1] = 2	#goto next
2	ar[2] = 7	ar[2] = 7 ⇄ ar[6] #swap ar[ ] & ar[ ]
	ar[2] = 8	ar[2] = 8 ⇄ ar[7] #swap ar[ ] & ar[ ]
	ar[2] = 3	#goto next
3	ar[3] = 4	#goto next Out of bounds / Irrelevant data / goto next
4	ar[4] = 9	ar[4] = 9 ⇄ ar[8]
5	ar[5] = 6	#goto next
6	ar[6] = 7	#goto next
7	ar[7] = 8	#goto next
8	#Stop.	

Note: After Modifications

	0	1	2	3	4*	5	6	7	
ar[8] = {	1	2	3	4	9	6	7	8	}

```
Iterate in arr[];
    if (ar[i] != i+1) {return i+1}
}
```

return N+1; # if all elements from 1-N are present ar[ ] = {

0	1	2	3	4	5	6	7	8	9
~~5~~	~~-14~~	~~6~~	~~7~~	~~9~~	~~-10~~	~~2~~	~~3~~	~~1~~	12 }
~~9~~	2	~~-10~~	~~2~~	5	6	7	-10	9	
1		3	-14						

ind i = i+1 ele

ele x = x-1 ind

i	Correct data	Take data to correct position
0	ar[0]=5	ar[0]=5 ⇄ ar[4] #swap ar[0] & ar[4]
	ar[0]=9	ar[0]=9 ⇄ ar[8] #swap ar[ ] & ar[ ]
	ar[0]=1	#goto next Out of bounds / Irrelevant data / goto next
1	ar[1]=-14	ar[1]=-14 ⇄ ar[-15]
2	ar[2]=6	ar[2]=6 ⇄ ar[5] #swap ar[2] & ar[5]
	ar[2]=-10	ar[2]=-10 ⇄ ar[-11];
		Out of bounds / Irrelevant data / goto next
3	ar[3]=7	ar[3]=7 ⇄ ar[6] #swap ar[3] & ar[6]
	ar[3]=2	ar[3]=2 ⇄ ar[1] #swap ar[3] & ar[1]
	ar[3]=-14	ar[3]=-14 ⇄ ar[-15]
		Out of bounds / Irrelevant data / goto next
4	ar[4]=5	#goto next
5	ar[5]=6	#goto next
6	ar[6]=7	#goto next
7	ar[7]=3	ar[7]=3 ⇄ ar[2] #swap ar[7] & ar[2]
	ar[7]=-10	ar[7]=-10 ⇄ ar[-11];
8	ar[8]=9	#goto next Out of bounds / Irrelevant data / goto next
9	ar[9]=12	ar[9]=12 ⇄ ar[11]
10	# stop	

After Modifications:

ans = 4.

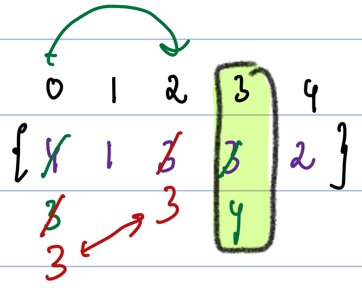
	0	1	2	3	4	5	6	7	8	9
ar[ ] = {	1	2	3	-14	5	6	7	-10	9	12 }

```
int firstMissing(vector<int> &arr){
    int N=arr.size();
    for(int i=0; i<N; i++){
        # Bring correct data to index i.
        while(arr[i] != i+1){
            int ele= arr[i];
            int ind= ele-1;
            if(ind<0 || i>=N){ break; }  ⟶ #irrelevent data
            if(arr[i]==arr[ind]){ break; }  ⟶ #duplicates
            swap arr[i] & arr[ind];
        }
    }

    for(int i=0; i<N; i++){
        if( arr[i] != i+1){
            return i+1;
        }
    }
    return N+1;
}
```

Edge Case:

ar[5] =



i	Correct Data	Take data to correct position.
0	ar[0] = 4	ar[0] = 4 ⇒ ar[3] swap ar[0] & ar[3]
	ar[0] = 3	ar[0] = 3 ⇒ ar[2] swap ar[0] & ar[2]
	ar[0] = 3	ar[0] = 3 ⇒ ar[2] swap ar[0] & ar[2]

Note: if elements we swap are same we break.

```
int firstMissing(vector<int> &ar) {
    int N = ar.size();
    for(int i=0; i<N; i++){
        # Bring correct data to Index i.
        while(ar[i] != i+1 && ar[i]>=1 && ar[i]<=N && ar[i] != ar[ar[i]-1])
            swap ar[i] & ar[ar[i]-1]   # Index
        }
    }

    for(int i=0; i<N; i++){
        if(ar[i] != i+1){
            return i+1;
        }
    }
    return N+1;
}
```

Total Iterations = N+N = 2N

Total Outerloop = {0. N-1} = N

+

Total Innerloop = N

a. 1 Innerloop itera = 1 swap

b. 1 swap will bring atleast 1 element to it's correct position

c. At most we have N swaps

Total Inner loop Iterations = N swaps

#Note: If there is break in 2nd loop calculate iterations & estimate BigO