Next Permutation

Brute Force

generating all permutation and then searching the given permutation and finding out the next in the order then returning it. If it is last permutation then returning 1st permutation.

Time Complexity : O(N*N!)

Space Complexity :O(N)

Approach

Treating the array as real number for example [1,2,3] read as 123, the next permutation or the next greater number that can be formed using these digits is 132 [1,3,2] is thus the next permutation.One thing to observe is that when we move from right to left as soon as we encounter a smaller number we think of swapping it with its just greater number because we need a number just greater than the given number so using this observation we can loop in reverse order and find a number that is less than the current digit .On the contrary if we get a number greater than the current number then using the digits we cannot form a greater number so we are in search of a smaller number.As soon as it is found we break out from the loop. We then find a just greater number in the right half(already traversed part of the array just now) and swap the dip element with it. As we know if we swap a number greater than the dip element we need the other numbers to be in sorted order in th right half so that the number formed is just greater as the array was already sorted in descending order we just reverse it to get the increasing order.

Complexity

- Time complexity:

```
O(N)
```

- Space complexity:

```
O(1)
```

Code

```
class Solution {
public:
    void nextPermutation(vector<int>& nums) {
       int i,j,n=nums.size();
       for(i=n-2;i>=0;i--)
       {
           if(nums[i]<nums[i+1])
                break;
       }
       if(i<0)
       {
           reverse(nums.begin(),nums.end());
       }
       else
       {
            for(j=n-1;j>=i;j--)
            {
                if(nums[j]>nums[i])
                    break;
            }
           swap(nums[i],nums[j]);
           reverse(nums.begin()+i+1,nums.end());
       }
    }
};
```

can also use next_permutation(nums.begin,nums.end()); //inbuilt function