Longest Consecutive Subsequence

BRUTE FORCE:

We pick up one element and search for an element just greater than 1 ex: if 1 is curr elem then 2 is just greater so we linearly search for 2 if it is found then we increment count and curr becomes 2 we keep doing this until a next element is found and if the consecutive element is not found we break.

We do this for every element and store the max count in maxlen variable and return it.

```
#include <bits/stdc++.h>
bool search(vector<int> &arr,int x)
{
    for(int i=0;i<arr.size();i++)
    {
        if(arr[i]==x)
            return true;
    }
    return false;
}
int lengthOfLongestConsecutiveSequence(vector<int> &arr, int n) {
    // Write your code here.
    int maxlen=INT_MIN;
    for(int i=0;i<n;i++)
    {
        int curr=arr[i];
        int cnt=1;
        while(search(arr,curr+1)==true)
        {
            curr=curr+1;
            cnt++;
        }
        maxlen=max(maxlen,cnt);

    }
    return maxlen;
}
```

- Time Complexity : O(N^3)
- Space Complexity : O(1).

Better Approach :

We sort the array and keep counting the consecutives ignoring duplicates. Storing maxlen for maximum count. We keep a track of lastSmaller element encountered. If a new non-consecutive element is encountered we set the cnt =1 and lastSmaller to current element. when lastSmaller is not equal to curr element we know that it has not entered the previous if statement and that the consecutiveness has ended so we reset the lastSmaller elment and keep couting the consecutives for new elem.

```
#include <bits/stdc++.h>

int lengthOfLongestConsecutiveSequence(vector<int> &arr, int n) {
    // Write your code here.
    sort(arr.begin(),arr.end());
    int x,cnt=1,lastSmaller=INT_MAX,maxlen=0;
    for(int i=0;i<n;i++)
    {
      if (arr[i] - 1 == lastSmaller) {
        lastSmaller = arr[i];
        cnt++;
      }
      if(arr[i]!=lastSmaller)
      {
          cnt=1;
          lastSmaller=arr[i];
      }
      maxlen=max(maxlen,cnt);
    }
    return maxlen;
}
```

- Time Complexity : O(NlogN)+ O(N)
- Space Complexity : O(1)

Optimal Approach :

We push all elements to set because we do not care about duplicates next we try to find the starting element of consecutive sequence. If no previous element exist that is a num 1 less than current element then we know it can be starting of consecutive element so we loop until an element greater than current element can be found in hashset and keep on increasing cnt. We store the max of cnt in maxlen.

```
#include <bits/stdc++.h>

int lengthOfLongestConsecutiveSequence(vector<int> &arr, int n) {
    // Write your code here.
    unordered_set<int> s;
    int x,cnt=1,maxlen=0;
    for(int i=0;i<n;i++)
    {
        s.insert(arr[i]);
    }
    for(auto it:s)
    {
        if(s.find(it-1)==s.end())
        {
            cnt=1;
            x=it+1;
            while(s.find(x)!=s.end())
            {
                cnt++;
                x=x+1;
            }
            maxlen=max(maxlen,cnt);
        }
    }
    return maxlen;
}
```

- Time Complexity : O(2N)

Inner loop traverses N times and outer loop N times so totally eleemnts are traversed 2N.

for test case : 102 4 101 1 2 100 3

for 102 1 comparison is made 101 is found so inner loop does not run. it runs only for 1 2 3 4 and 100 101 102 which gives 7 iterations and iterating each element in hashset once gives 7 iteration which equals 14 iteration = 2N so complexity is O(2N).

- Space Complexity : O(N).