LEVEL 2

WEEK #3

SEASON 2024

HELWAN ICPC
COMMUNITY

Overview

- COMPARE FUNCTION
- BINARY SEARCH
- 2 POINTERS
- ADDITIONAL RESOURCES

Compare Function

Compare Function is used to sort a container in a specific manner based on certain criteria.

While building a compare function, we can think of it as doing a bubble sort, where each two pairs are checked to identify the smallest/greatest.

In the shown code, cmp function is passed as a third parameter in the sort function.

The cmp function will start comparing each pair by the passed function until sorting is done.

```
bool cmp(int i, int j){
    return i > j;
}
void solve () {
    vector <int> vec = {4, 6, -1, 100, 71};
    sort(all(vec), cmp);
    for(auto i: vec) cout << i << " ";
}
```

Compare Function

Example:

In the shown code, the sorting is done based on the following criteria :

- Odd numbers are considered smaller than Even numbers.
- If two numbers have the same parity then sort them ascendingly.

The expected output is :

```
1 71 4 6 100
```

```cpp
bool cmp(int i, int j){
    if(i%2 == j%2)
        return i < j;
    else if(i%2) return true;
    else return false;
}
void solve () {
    vector <int> vec = {4, 6, 1, 100, 71};
    sort(all(vec), cmp);
    for(auto i: vec) cout << i << " ";
}
```

RECAP

Binary Search

Binary Search is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half.

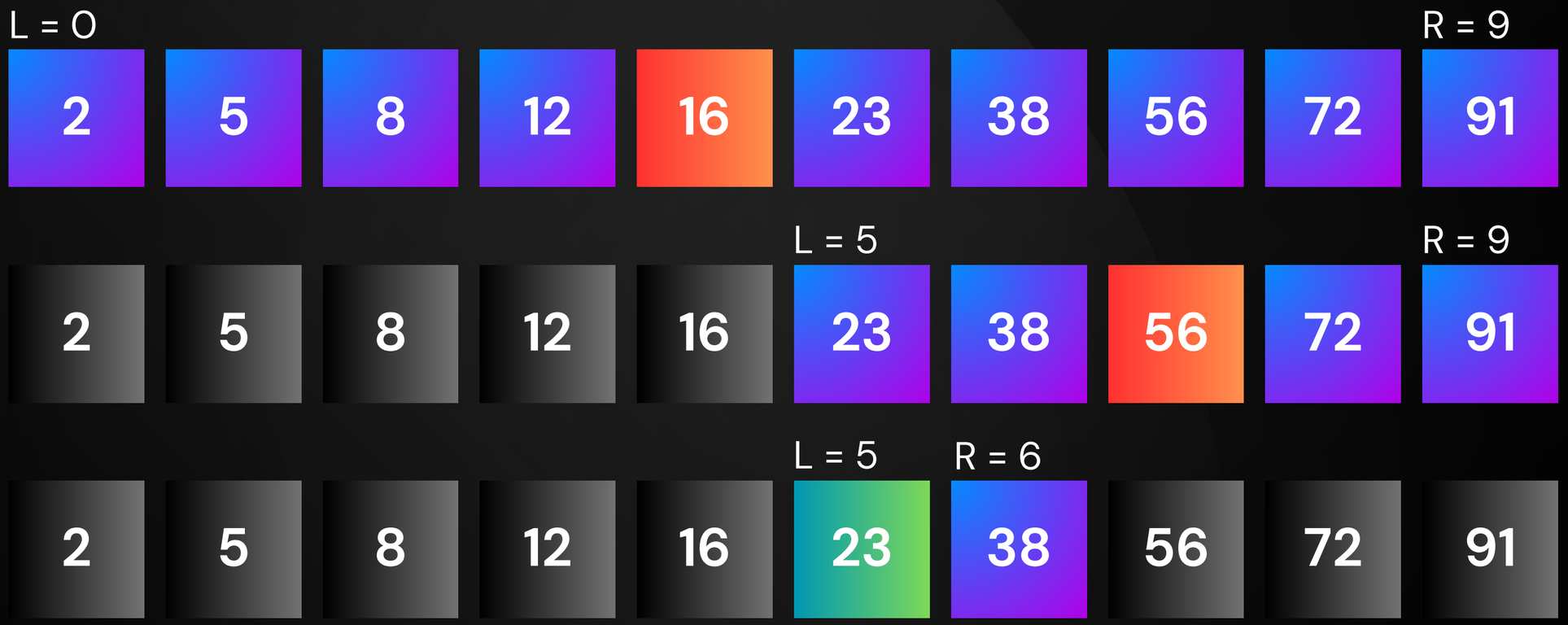
The idea of binary search is to use sorted array and reduce the time complexity to O(log N).

Conditions for when to apply Binary Search

- The data structure must be sorted.
- Access to any element of the data structure takes constant time O(1).

Binary Search

RECAP



Searching for 23

Binary Search

RECAP

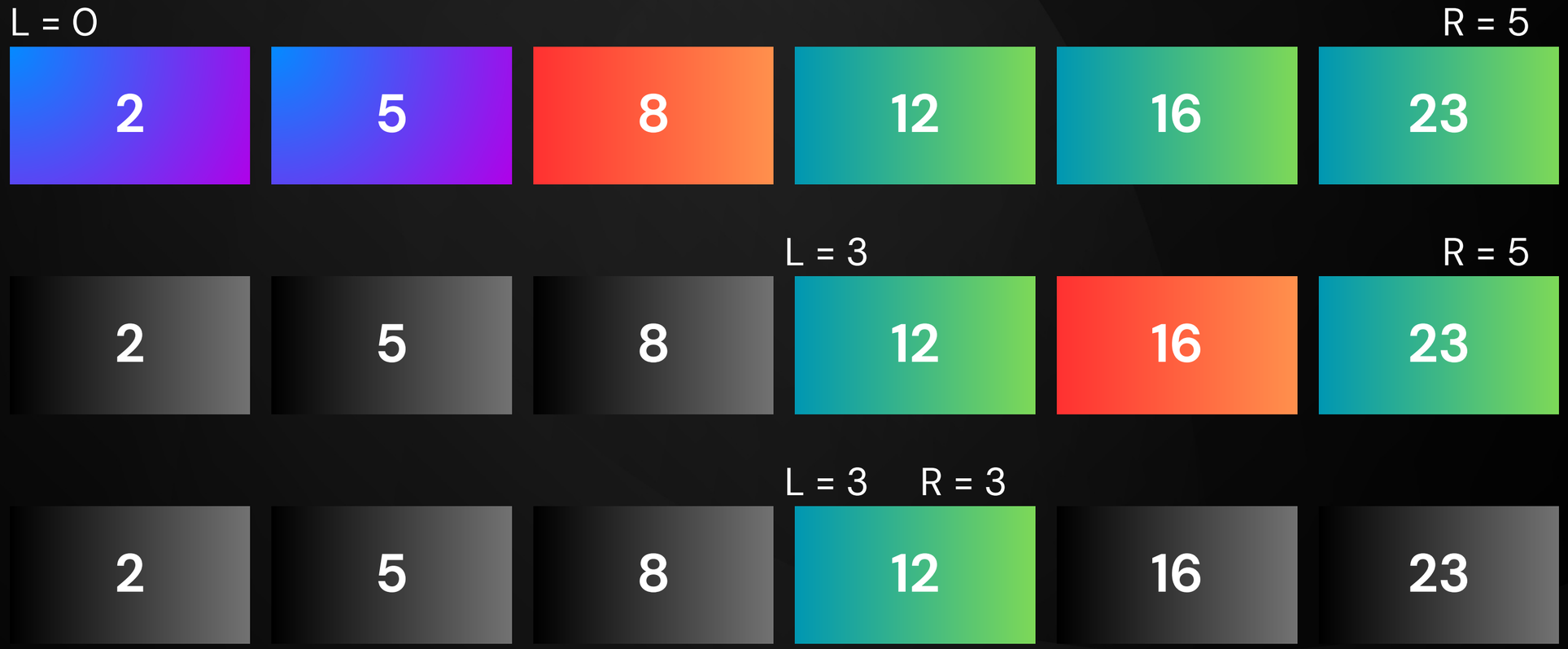
- **Divide the search interval** into two halves by finding the middle index “m”.
- **Compare the middle element** of the search interval with the key.
- If the **key is found** at middle element, the process is **terminated**.
- If the key is not found at middle element, choose which half will be used as the next search space.
- If the **key is smaller than the middle** element, then the **left side** is used for next search.
- If the **key is larger than the middle** element, then the **right side** is used for next search.
- This process is continued until the key is found or the total search interval is exhausted.

```
int arr[] = { 2, 3, 4, 10, 40 };
int x = 10, n = 5, ans = -1;
int l = 0, r = n-1;
while (l ⩽ r) {
    int m = (r + l) / 2;

    // Check if x is present at mid
    if (arr[m] == x) {
        ans = m;
        break;
    }

    // If x greater, ignore left half
    if (arr[m] < x)
        l = m + 1;

    // If x is smaller, ignore right half
    else
        r = m - 1;
}
cout << ans;
```

RECAP

Binary Search

Using binary search, how to find the number of elements greater than or equal to x? Assume x = 10.



RECAP

Binary Search

Unlike searching for specific value, searching for a range is guaranteed to yield a solution.

Here is an example showing the code for the previous figure on how to get the number of elements in a specific range.

```
int arr[] = { 2, 3, 4, 10, 40 };
int x = 4, n = 5;
int l = 0, r = n-1;
while (l ⩽ r) {
    int m = (r + l) / 2;
    if (arr[m] < x)
        l = m + 1;
    else
        r = m - 1;
}
cout << n - l;
```

RECAP

Lower and Upper Bound

Both Set and Map contain a function that perform operation similar to the binary search, lower_bound() and upper_bound().

```
set <int> st{1, 3, 5, 7, 9, 14, 15};
auto it = lower_bound(all(st), 5); //iterator on 5
auto ii = upper_bound(all(st), 5); //iterator on 7

map <int, int> mp{{2, 5}, {5, 10}};
auto it = lower_bound(all(mp), 2); //iterator on 2
auto ii = upper_bound(all(mp), 2); //iterator on 3
```

Given a specific value x, lower_bound(x) returns an iterator on the first element with value bigger than or equal to x.

While upper_bound(x) returns an iterator on the first element with value bigger than x.

Binary Search by Answer

As discussed so far, Binary Search is usually done on a container where searching for a specific element is done.

The binary search by answer approach is not employed directly on a container; rather, it is used to verify a particular algorithm's ability to identify the minimum or maximum value that satisfies the given problem.

Binary Search on answer is usually used when the target is to find the minimum or maximum value that satisfies the given problem.

The approach is applied by searching the whole answer range, validating the chosen value using an algorithm then updating the range based on the result of that algorithm.

HOW TO BS BY ANSWER?

LET'S DO SOME CODE

Two Pointers Method

Briefly, Two Pointers Method is iterating on a container using more than one variable or moving in more that one direction.

The easiest way to explain it is with the example of the task "Merging two sorted arrays".
We are given two arrays, sorted in ascending order, a and b.
We want to combine the elements of these arrays into one big array c, also sorted in ascending order.

A simple way to solve such a problem, is to insert all the elements of both arrays in a new large array and apply sorting.

Such technique is applied in O(n * log(n)) time complexity.

By using two pointers, such problem can be solved in linear time O(n).

Two Pointers Method

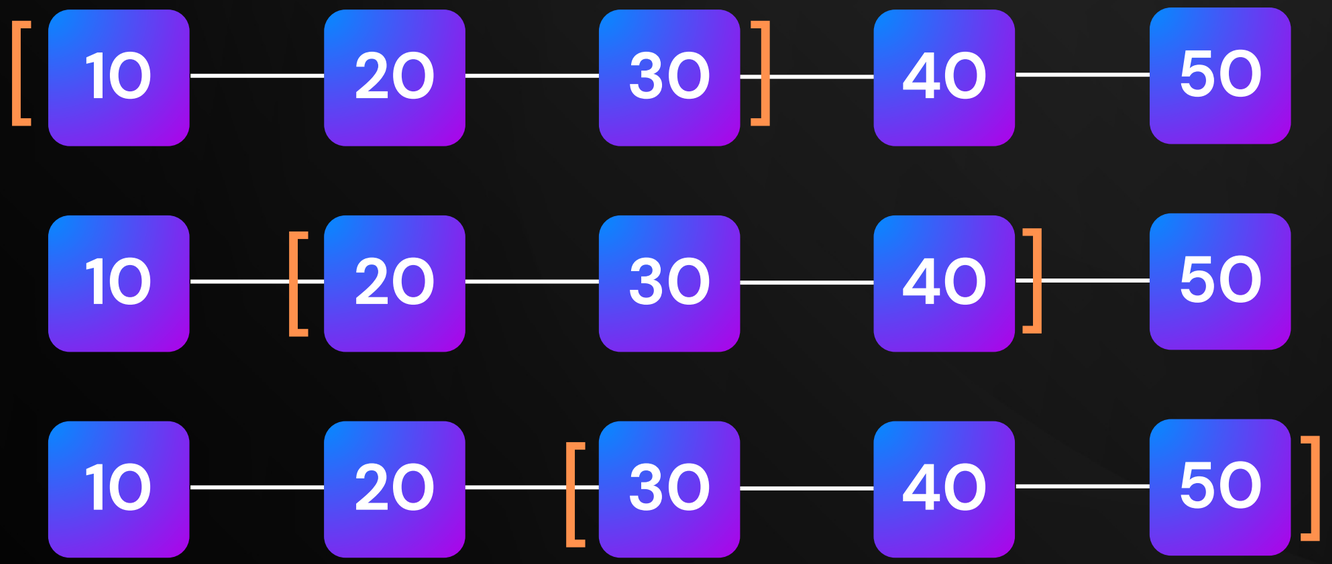
We can iterate both arrays in the same time using a pointer on each of them, and start comparing each element with its corresponding in the other array.

We start by comparing the first two elements 1 and 3, 1 is smaller so it’s inserting in the new array, then the pointer of the first array is moved into 2.
Then comparing 2 with 3 is applied, and so on.

Two Pointers Method

Fixed Window:

Fixed Window technique is used to iterate the container with fixed size interval.



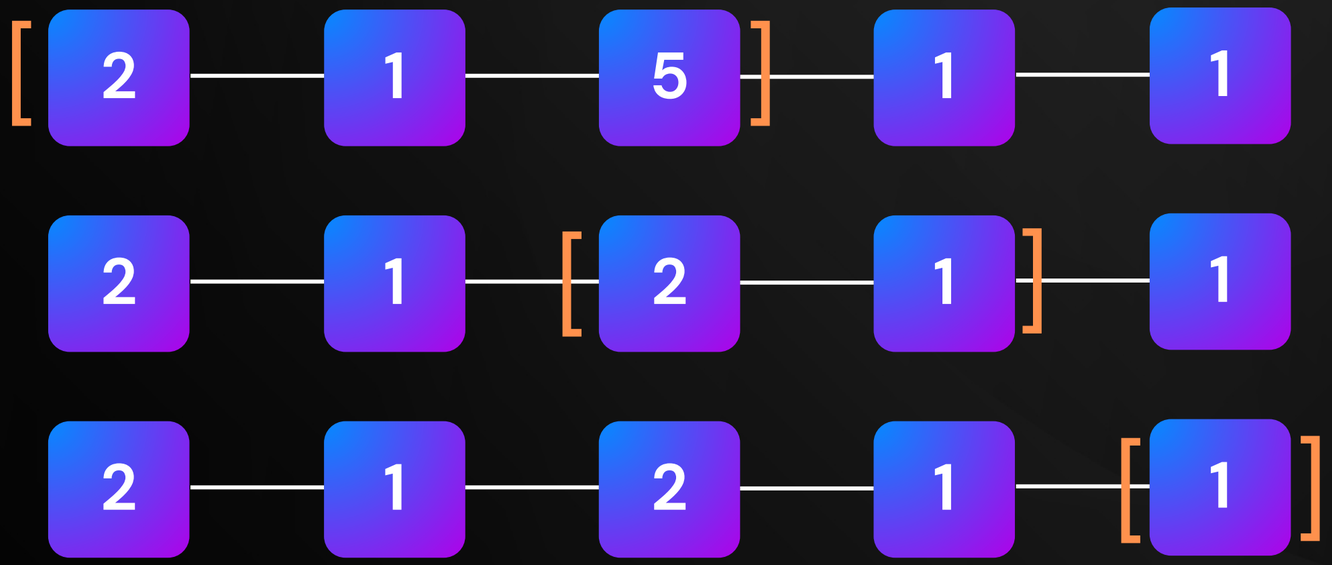

```
vector <int> vec = {10, 20, 30, 40, 50};
int windowSize = 3, currSize;
int sum = 0;
for (int i = 0; i < 5; ++i) {
    sum += vec[i];
    currSize ++;
    if(currSize > windowSize){
        currSize --;
        sum -= vec[i - windowSize];
    }
    if(currSize == windowSize)
        cout << sum << " ";
}
```

How Fixed Window is applied to get the sum of all intervals of size 3 in a container

Two Pointers Method

Sliding Window:

Sliding Window technique is same as fixed window, except for its size is dynamic and can be changed based on the problem.



You are given an array of integers. The array has a total of n numbers.

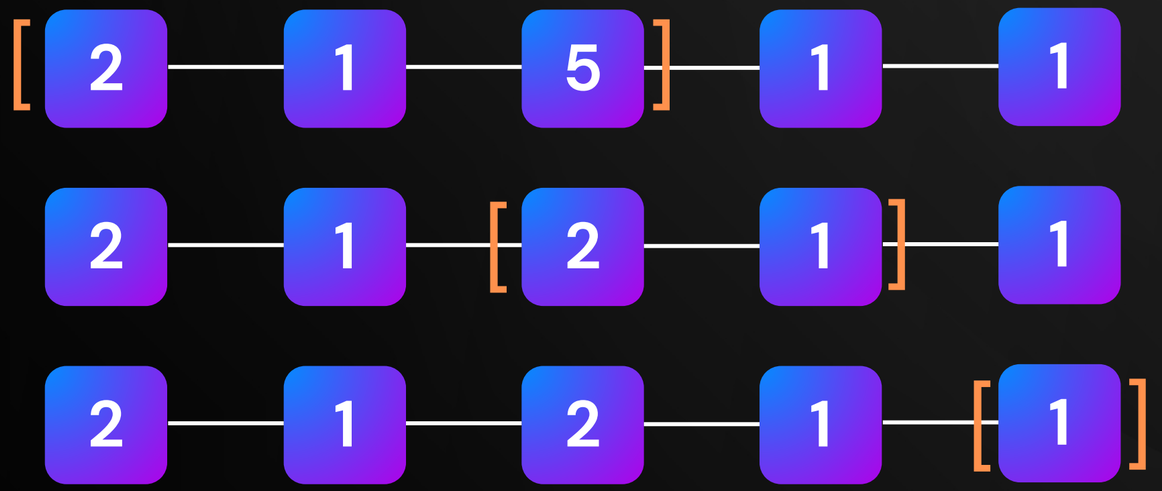
What is the longest sequence of successive elements where each element is unique?

Such a problem can be solved using sliding window technique, as shown in the opposite figure.

Iterating the container with a pointer on the start of the sequence, and once an element is duplicated, shrink the window.

Two Pointers Method

Sliding Window:




```
int n;
cin>>n;
vector <int> a(n);
map <int, int> mp;
for(int i = 0 ; i < n ; i ++)
    cin >> a[i];
int winStart = 0, winSize = 0, res = 0;
for(int i = 0 ; i < n ; i ++){
    if(mp.count(a[i])){
        for(int j = winStart ; j < mp[a[i]] ; j ++){
            mp.erase(a[j]);
            winSize --;
        }
        winStart = mp[a[i]] + 1;
        mp[a[i]] = i;
    }
    else{
        winSize ++;
        mp.insert({a[i], i});
        res = max(res, winSize);
    }
}
cout << res;
```

LET'S DO SOME CODE

HELWAN ICPC
COMMUNITY

THANK YOU

For attending

For more about Binary Search, you may watch the first 2 videos from here:

https://www.youtube.com/playlist?list=PLLtYWar82kdViawfHIMzIiPsXUJeIoNyk

For more about Two Pointers, you may watch this:

https://www.youtube.com/watch?v=ePTCI2xqZvo&list=PLLtYWar82kdViawfHIMzIiPsXUJeIoNyk&index=7

For both Binary Search and Two Pointers, study and solve codeforces edu.