Linear Search

Linear Search

index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

Linear Search

index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

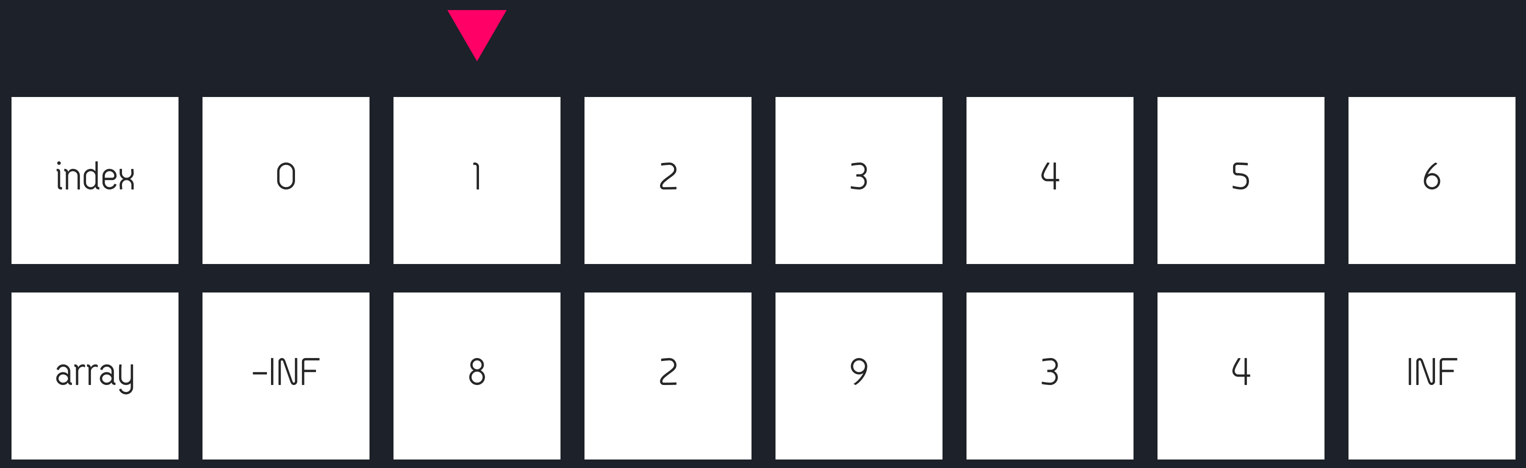
Where is the index of 3 ?

Linear Search

index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

Where is the index of 3 ?

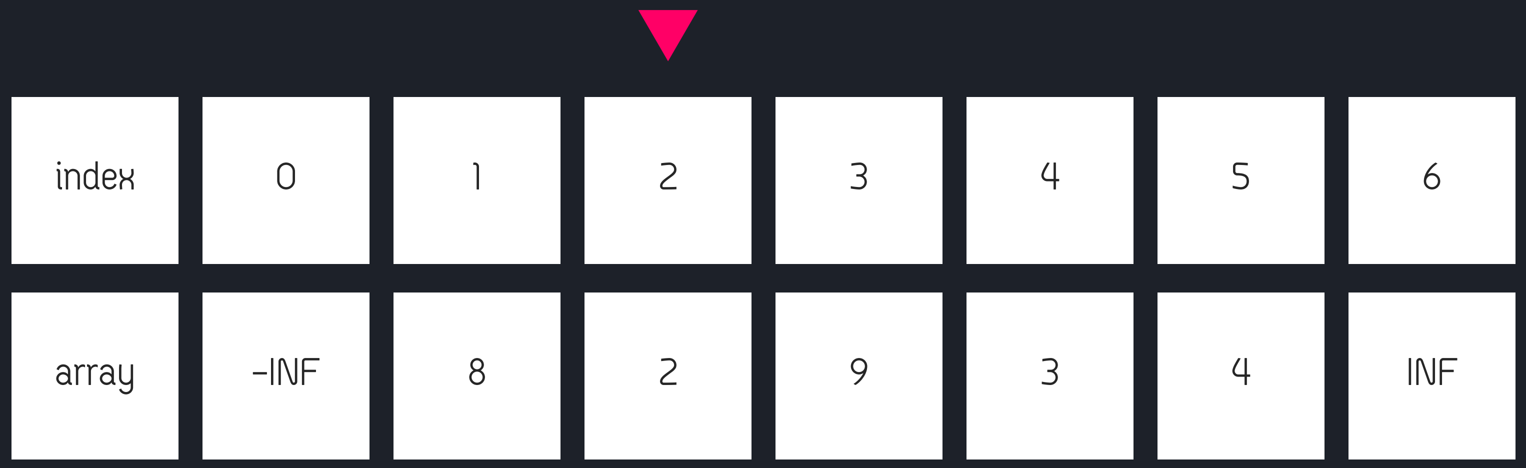
Linear Search



index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

Where is the index of 3 ?

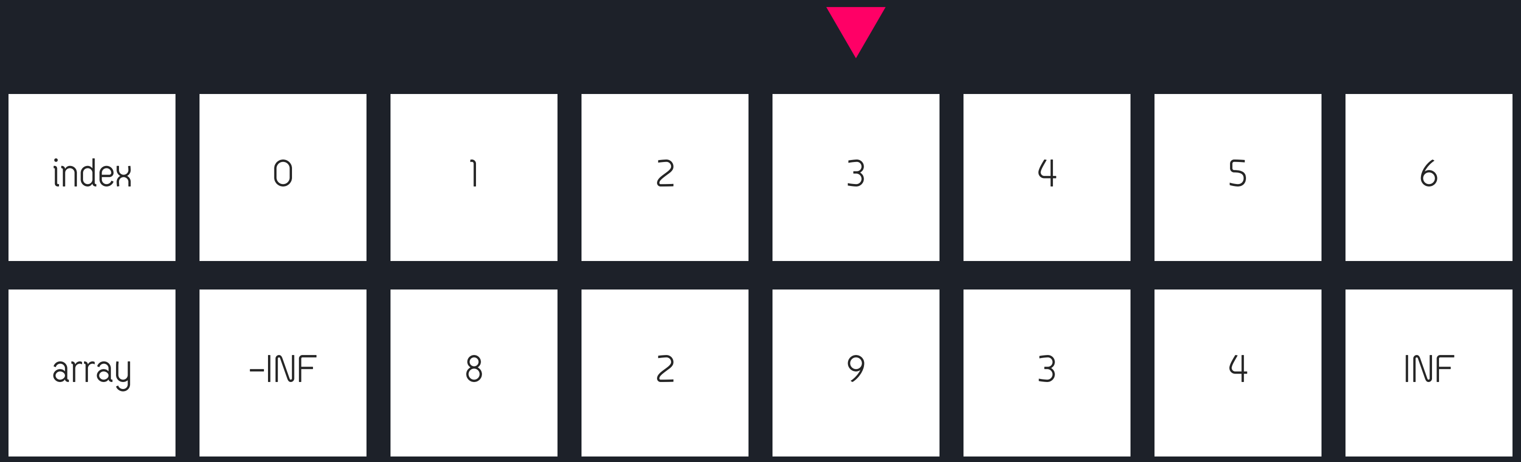
Linear Search



index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

Where is the index of 3 ?

Linear Search



index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

Where is the index of 3 ?

Linear Search

						▼		
index	0	1	2	3	4	5	6	
array	-INF	8	2	9	3	4	INF	

Where is the index of 3 ?

Linear Search

index	0	1	2	3	4	5	6
array	-INF	8	2	9	3	4	INF

The index of 3 is 4

Linear Search

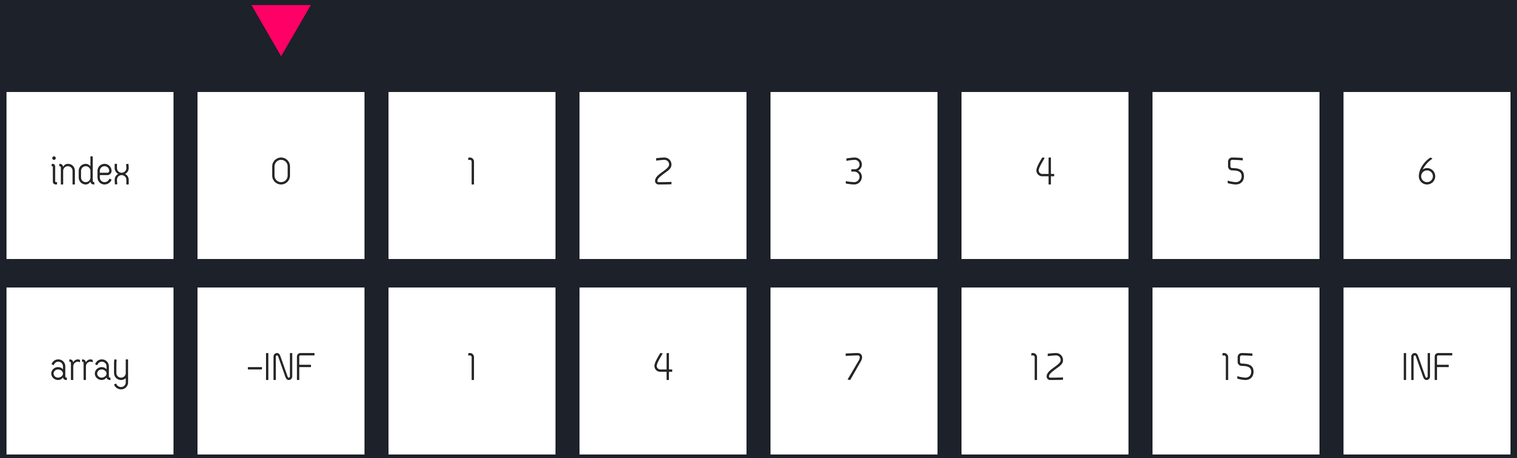
$O(n)$

Linear Search

index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

Where is the index of 12 ?

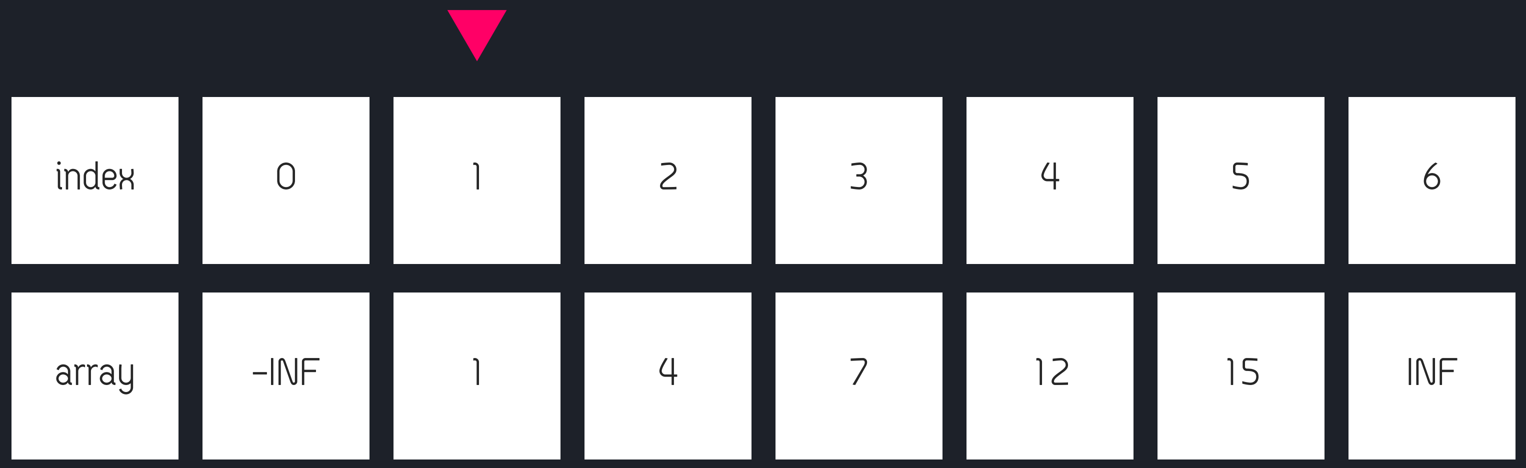
Linear Search



index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

Where is the index of 12 ?

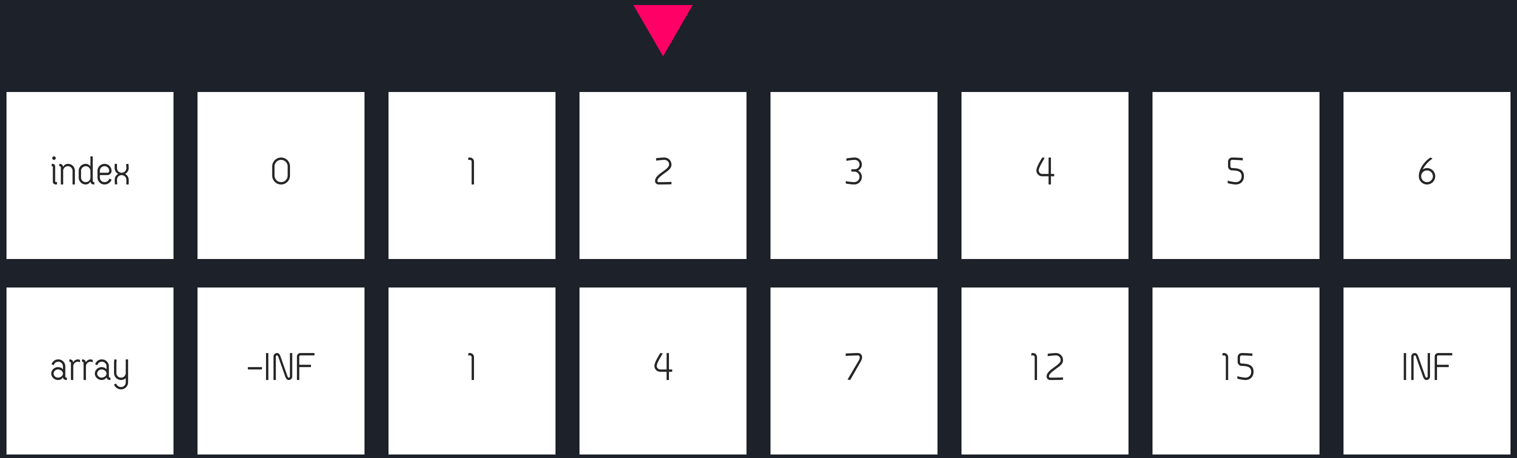
Linear Search



index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

Where is the index of 12 ?

Linear Search



index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

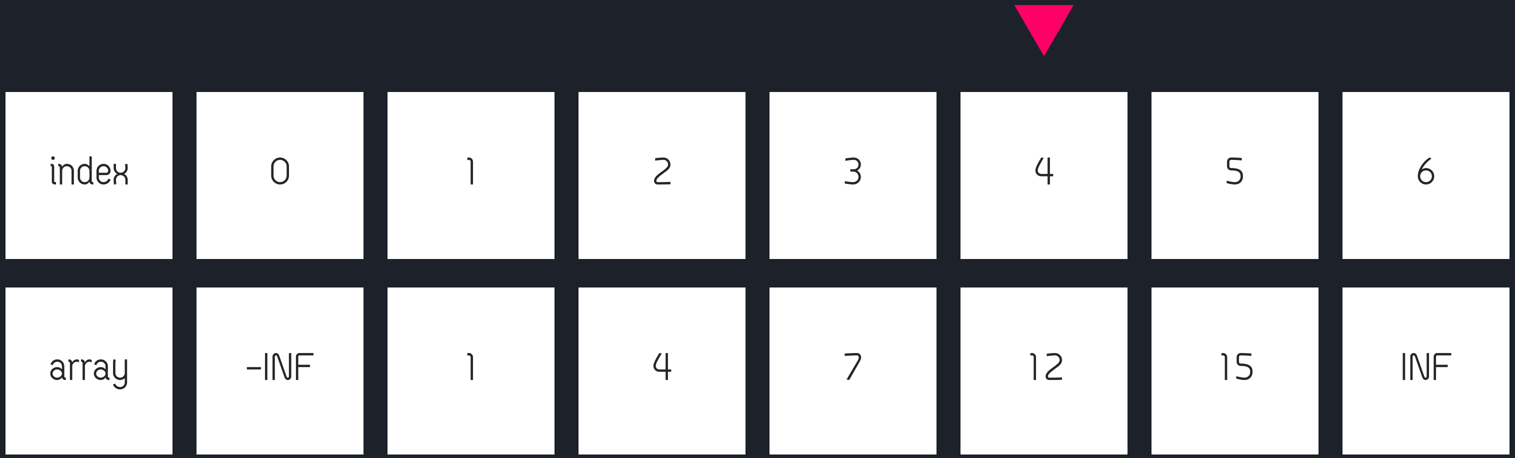
Where is the index of 12 ?

Linear Search

index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

Where is the index of 12 ?

Linear Search



index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

Where is the index of 12 ?

Linear Search

index	0	1	2	3	4	5	6
array	-INF	1	4	7	12	15	INF

The index of 12 is 4

Linear Search

$$O(n)$$

Binary Search

Binary Search

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

Binary Search

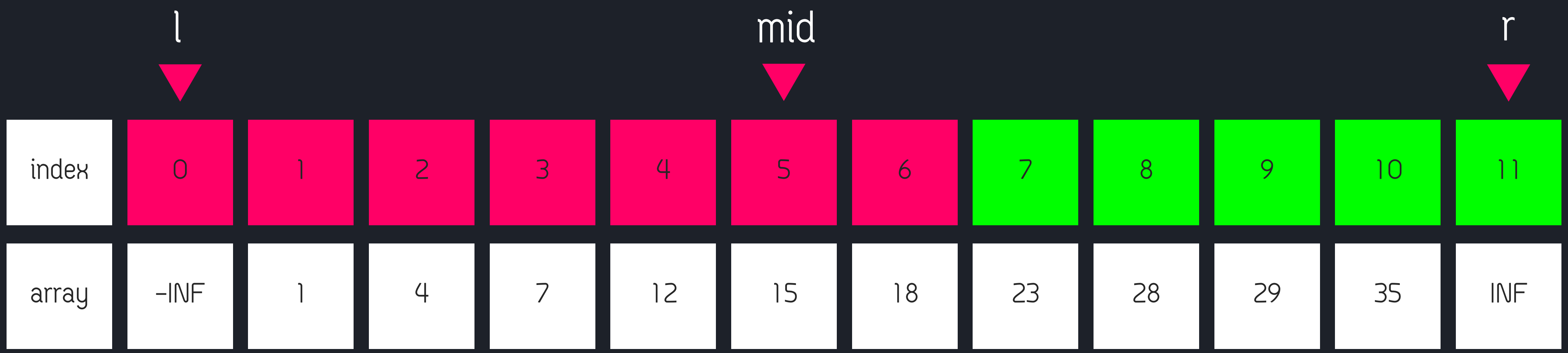
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

🟥 Value < 23

🟩 Value >= 23

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

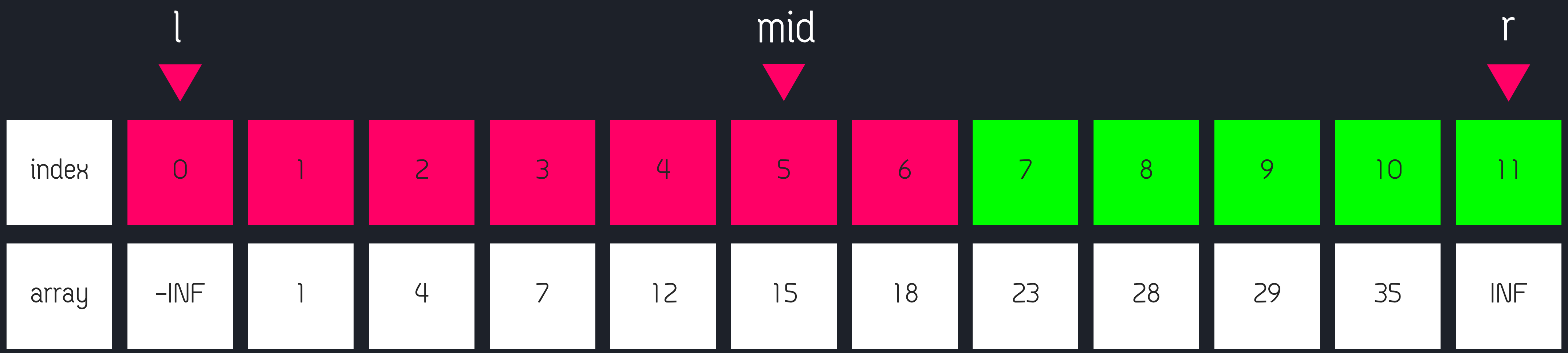
1.let l = 0 , r = 11

2.let mid = (l + r) / 2

3.If array[mid] >= 23 then r = mid
else array[mid] < then l = mid + 1

4.Repeat 2-3 until l = r

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

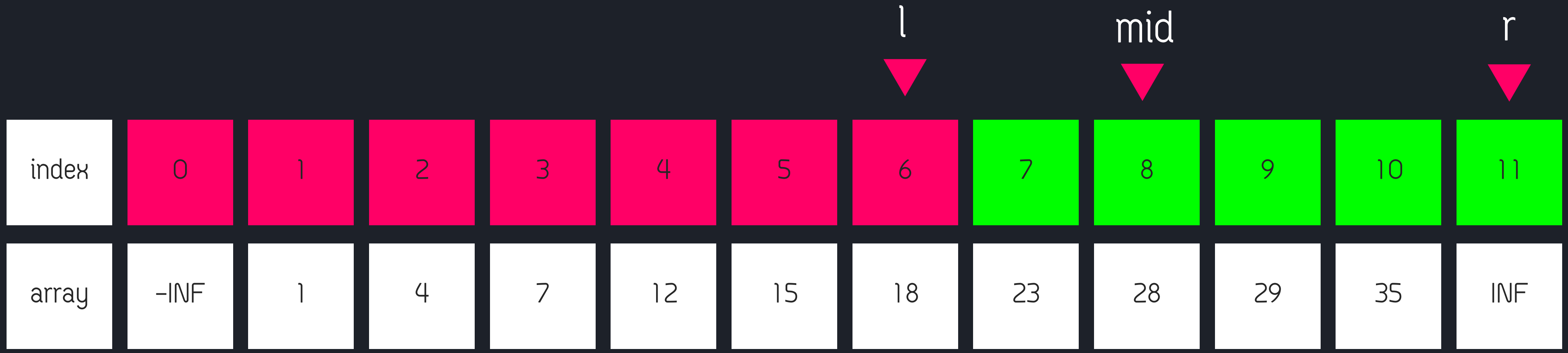
```
1.let l = 0 , r = 11

2.let mid = (l + r) / 2

3.If array(mid) >= 23 then r = mid
  else array(mid) < then l = mid + 1

4.Repeat 2-3 until l = r
```

Binary Search

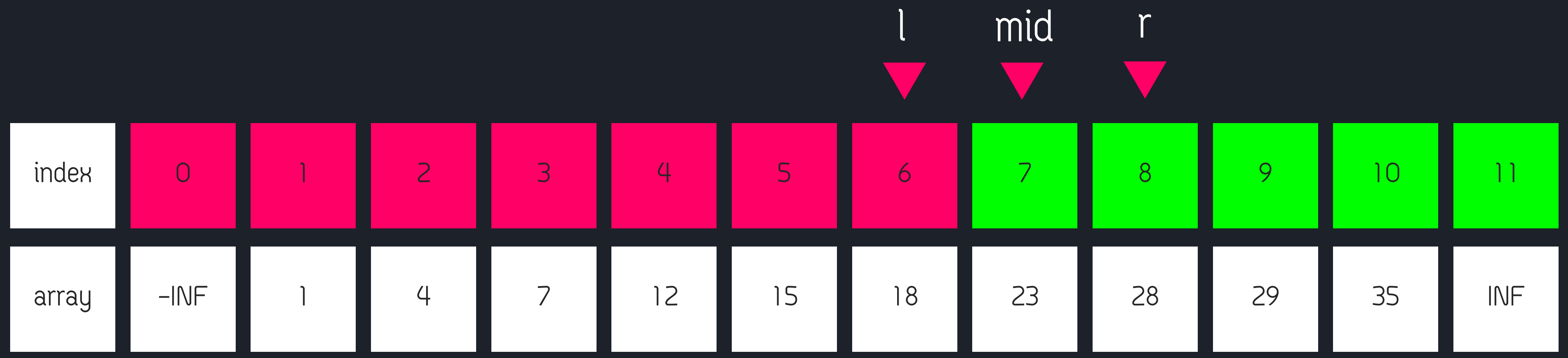


							l		mid			r
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search

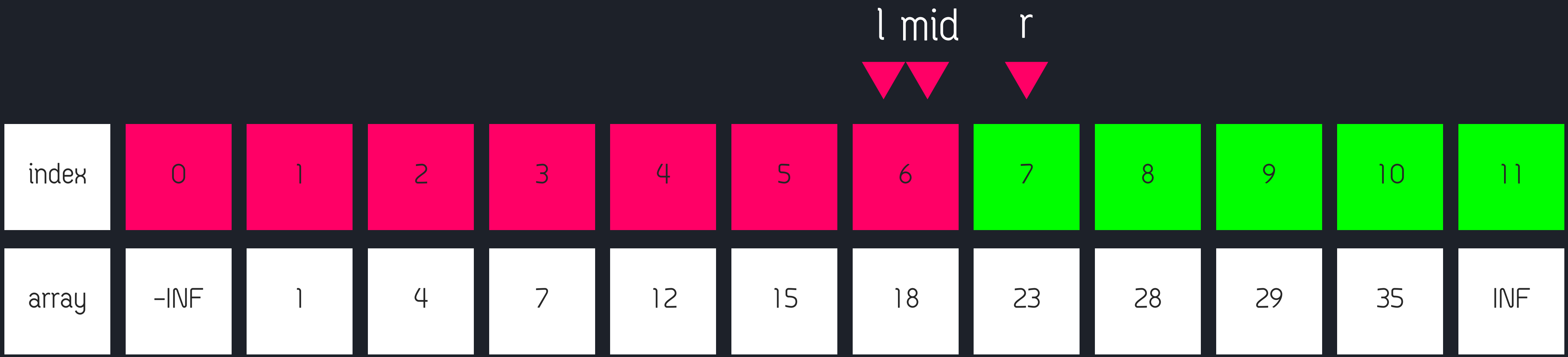


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search

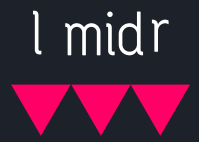


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array(mid) >= 23 then r = mid
 else array(mid) < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search

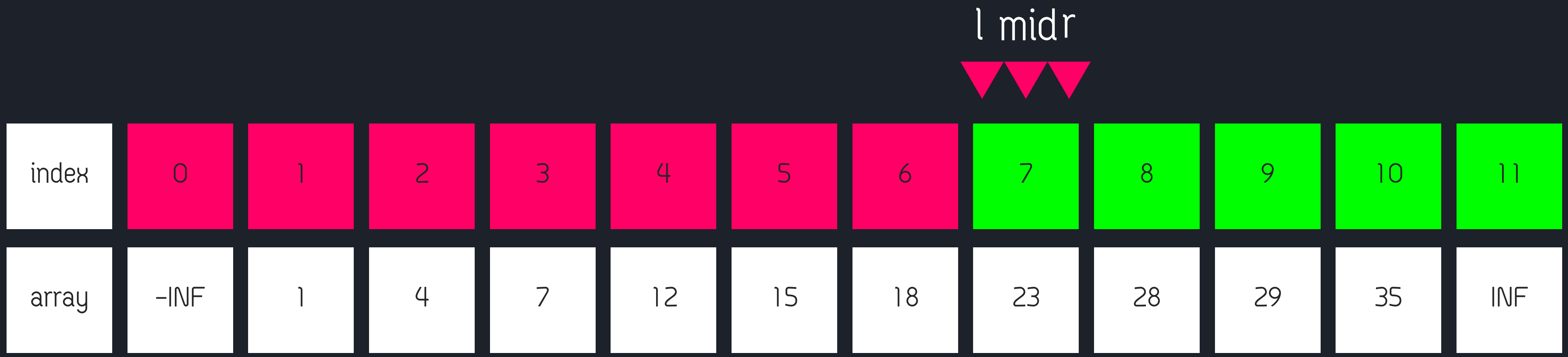


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search

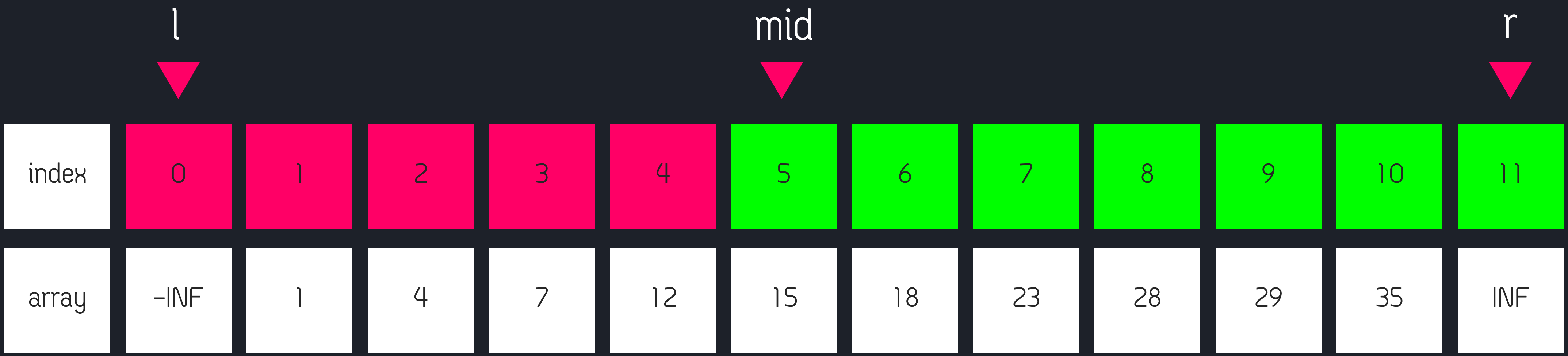


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

The index of 23 is 7

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array(mid) >= 23 then r = mid
  else array(mid) < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 14 ?

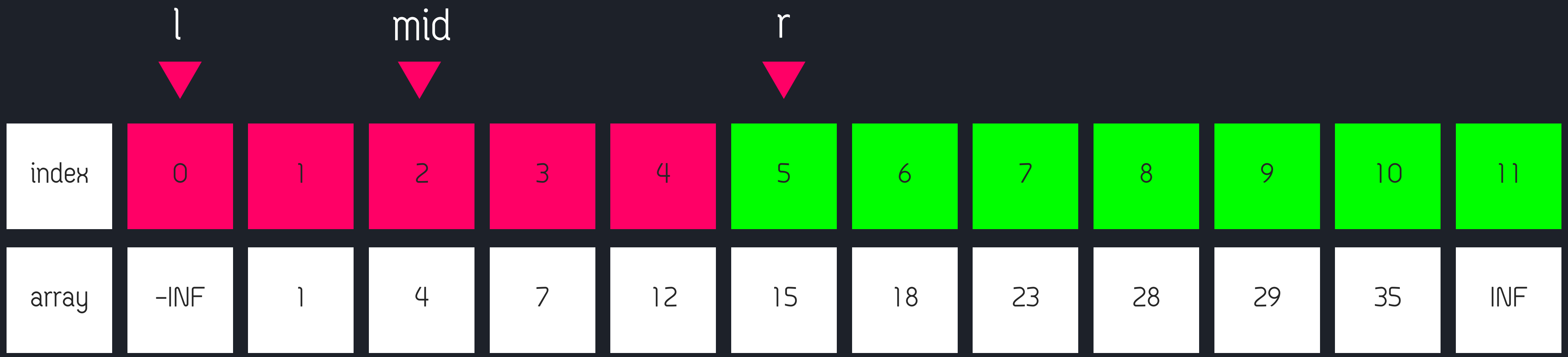
```
1.let l = 0 , r = 11

2.let mid = (l + r) / 2

3.If array[mid] >= 23 then r = mid
 else array[mid] < then l = mid + 1

4.Repeat 2-3 until l = r
```

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 14 ?

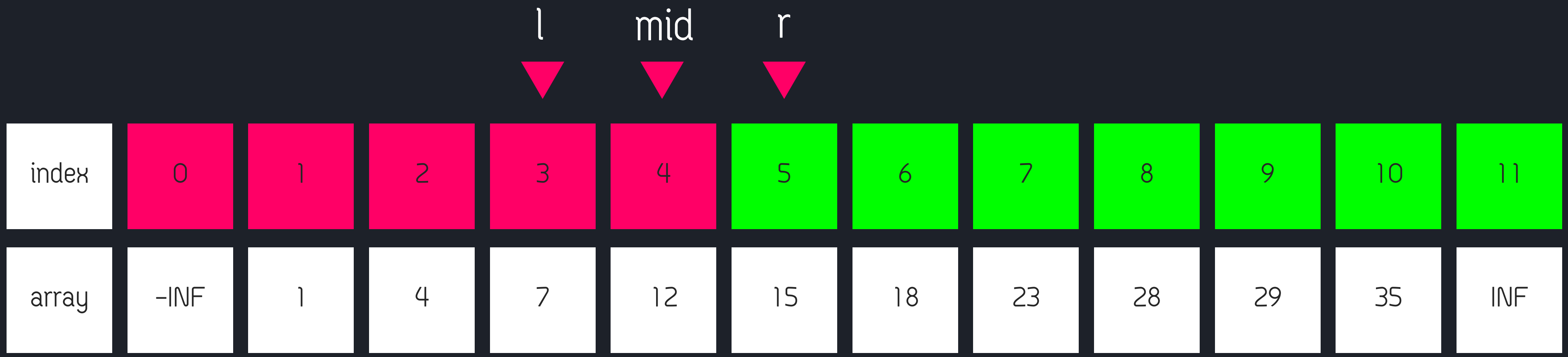
```
1.let l = 0 , r = 11

2.let mid = (l + r) / 2

3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1

4.Repeat 2-3 until l = r
```

Binary Search

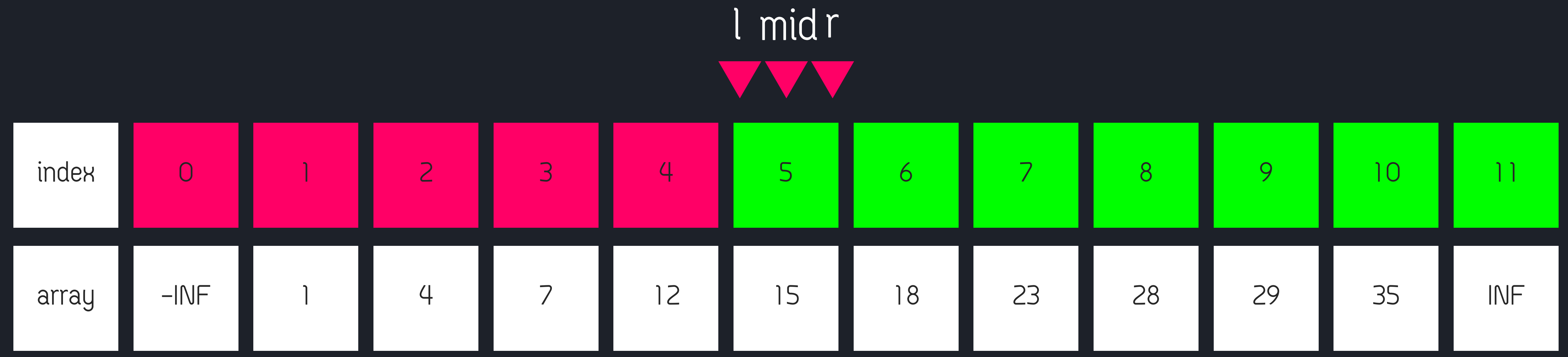


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 14 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array(mid) >= 23 then r = mid
 else array(mid) < then l = mid + 1
4.Repeat 2-3 until l = r
```

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 14 ?

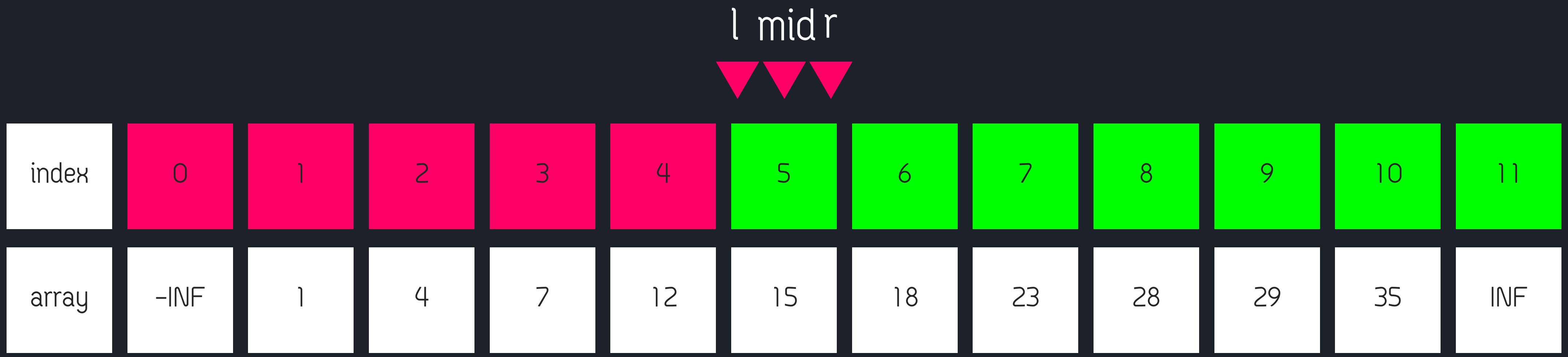
```
1.let l = 0 , r = 11

2.let mid = (l + r) / 2

3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1

4.Repeat 2-3 until l = r
```

Binary Search

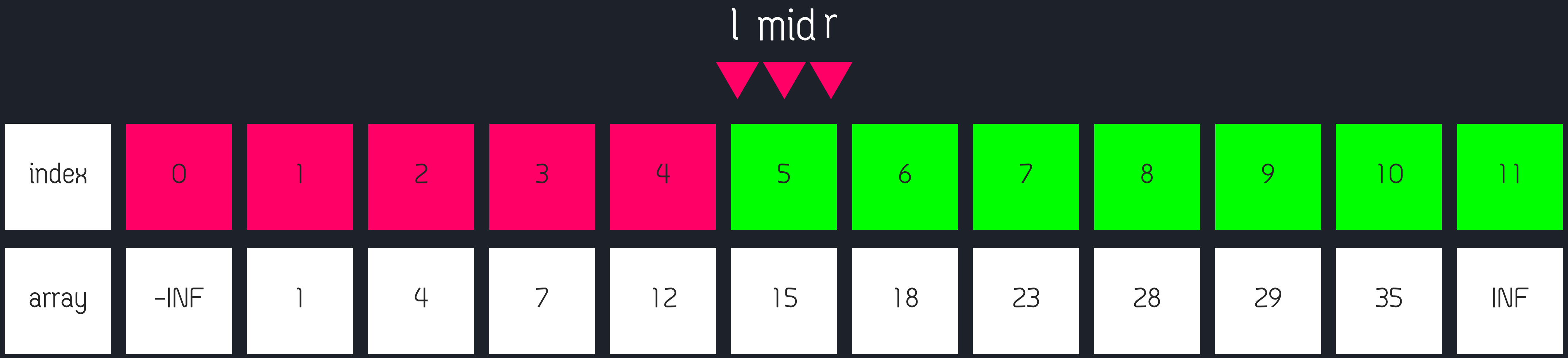


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Is the index of array[5] is 14 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1
4.Repeat 2-3 until l = r
5.If array[l] == 5 : found
  else  : not found
```

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

The index of array[5] is 15

```
1.let l = 0 , r = 11
2.let mid = (l + r) / 2
3.If array[mid] >= 23 then r = mid
  else array[mid] < then l = mid + 1
4.Repeat 2-3 until l = r
5.If array[l] == 5 : found
  else  : not found
```

Binary Search

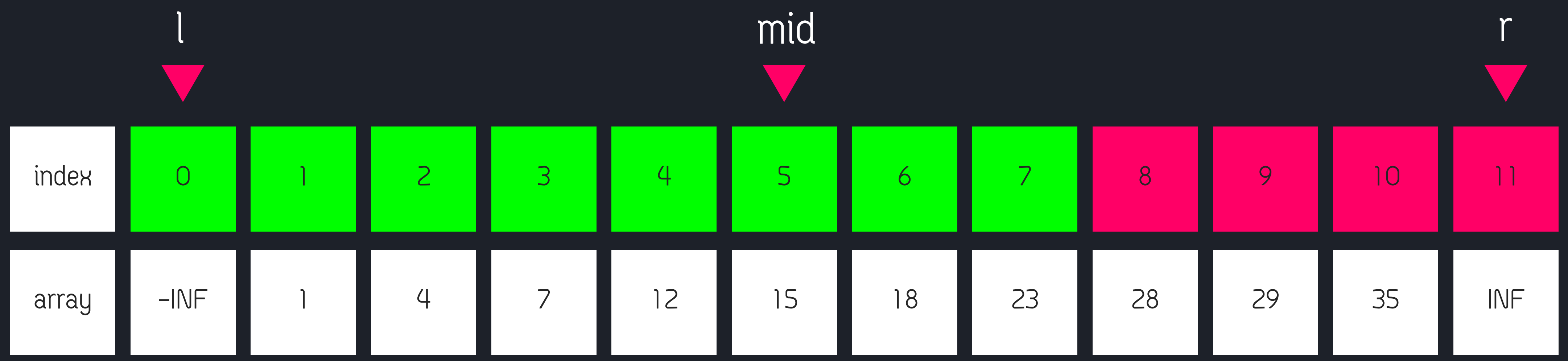
index	🟩 0	🟩 1	🟩 2	🟩 3	🟩 4	🟩 5	🟩 6	🟩 7	🟥 8	🟥 9	🟥 10	🟥 11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

🟥 Value > 23

🟩 Value <= 23

Binary Search

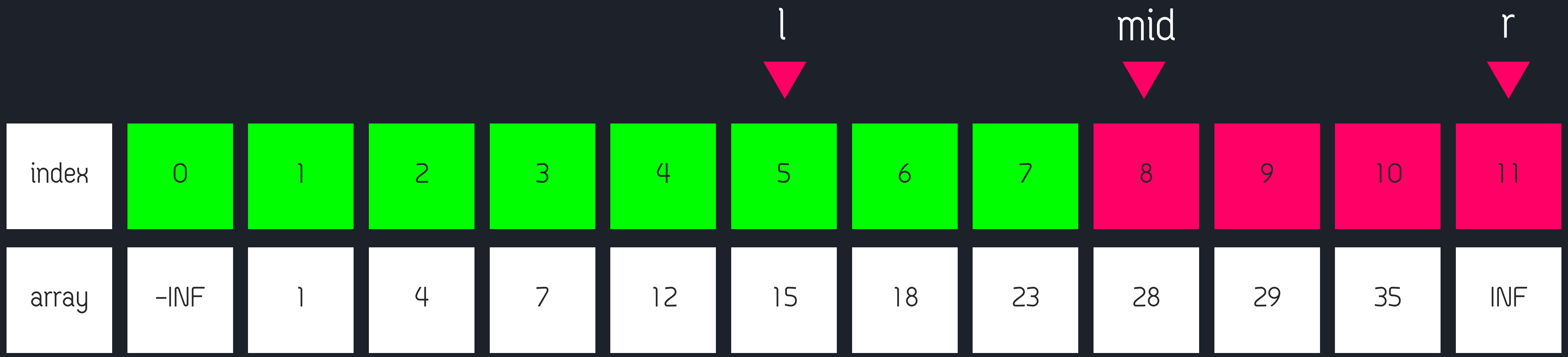


	l					mid						r
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r + 1) / 2
3.If array(mid) <= 23 then l = mid
  else array(mid) > 23 then r = mid - 1
4.Repeat 2-3 until l = r
5.If array(l) == 23 : found
  else  : not found
```

Binary Search

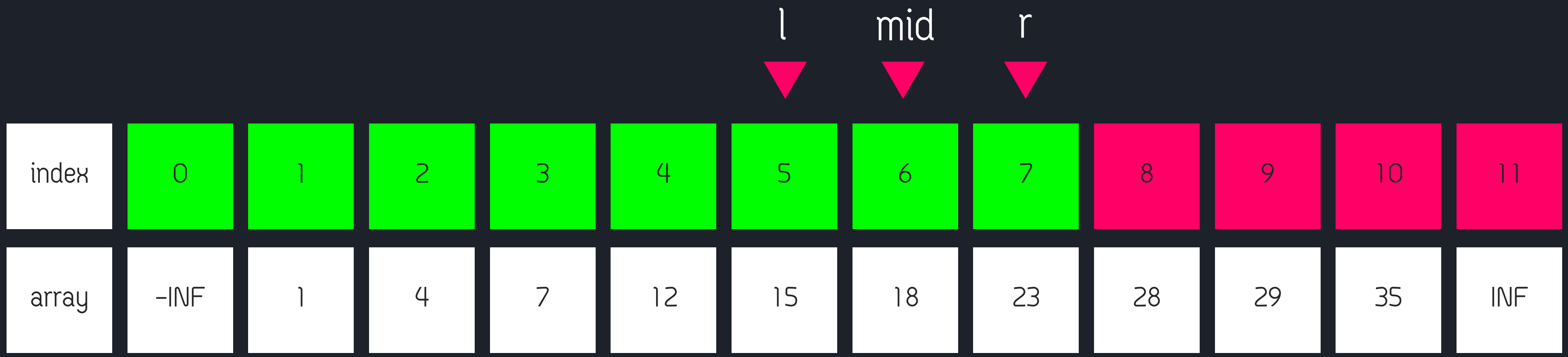


index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r + 1) / 2
3.If array(mid) <= 23 then l = mid
  else array(mid) > 23 then r = mid - 1
4.Repeat 2-3 until l = r
5.If array(l) == 23 : found
  else  : not found
```

Binary Search

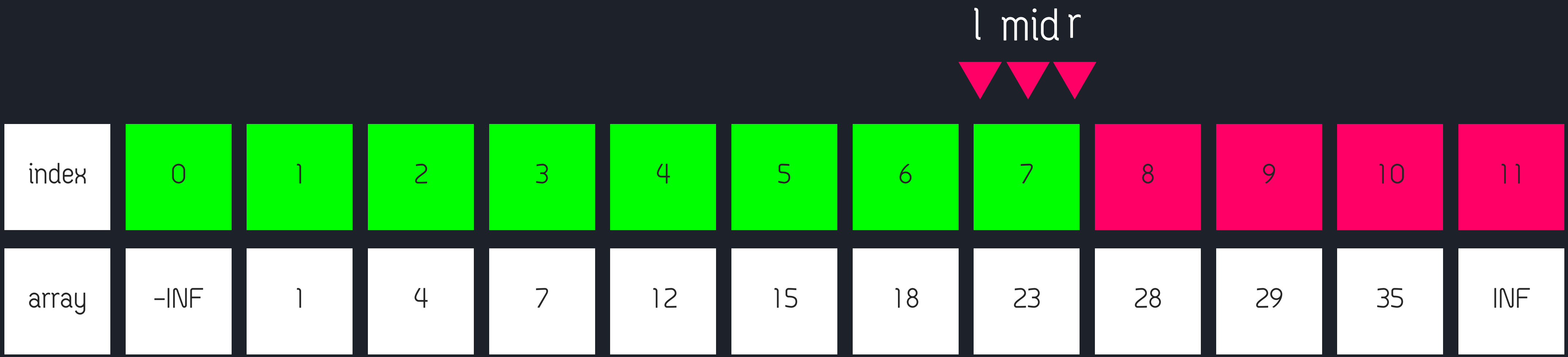


						l	mid	r				
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Where is the index of 23 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r + 1) / 2
3.If array(mid) <= 23 then l = mid
  else array(mid) > 23 then r = mid - 1
4.Repeat 2-3 until l = r
5.If array(l) == 23 : found
  else  : not found
```

Binary Search



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

The index of 23 is 7 ?

```
1.let l = 0 , r = 11
2.let mid = (l + r + 1) / 2
3.If array(mid) <= 23 then l = mid
  else array(mid) > 23 then r = mid - 1
4.Repeat 2-3 until l = r
5.If array(l) == 23 : found
  else   : not found
```

Upperbound & Lowerbound

Upper and Lower bound

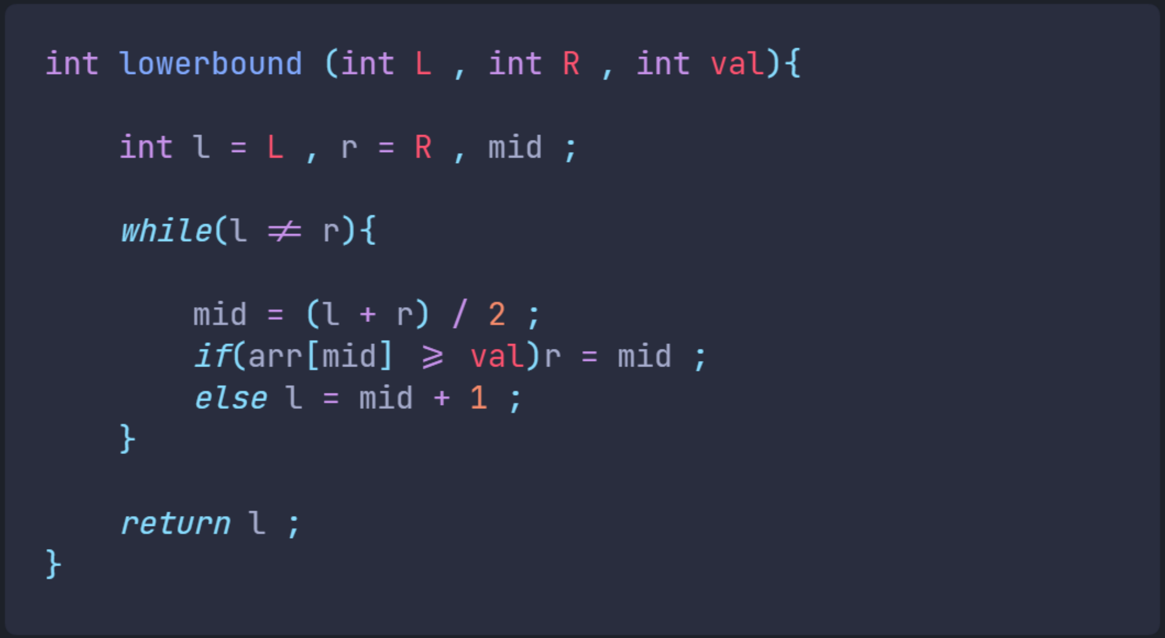
Upperbound(l , r , val)

- first index in range l-r that value more than val

Lowerbound(l , r , val)

- first index in range l-r that value more than equal val

Upper and Lower bound



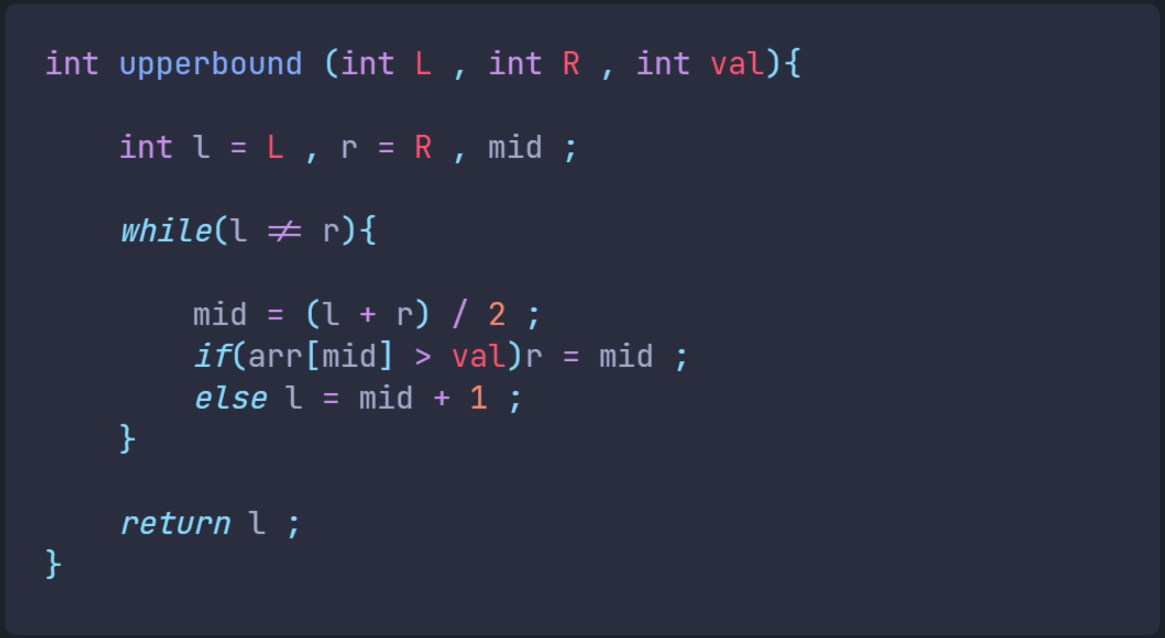
```
int lowerbound (int L , int R , int val){

    int l = L , r = R , mid ;

    while(l ≠ r){

        mid = (l + r) / 2 ;
        if(arr[mid] ⩾ val)r = mid ;
        else l = mid + 1 ;
    }

    return l ;
}
```



```
int upperbound (int L , int R , int val){

    int l = L , r = R , mid ;

    while(l ≠ r){

        mid = (l + r) / 2 ;
        if(arr[mid] > val)r = mid ;
        else l = mid + 1 ;
    }

    return l ;
}
```

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Upperbound(0 , 11 , val)

lowerbound(0 , 11 , val)

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Upperbound(0 , 11 , 13) -> 5

lowerbound(0 , 11 , 13) -> 5

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	4	7	12	15	18	23	28	29	35	INF

Upperbound(0 , 11 , 12) -> 5

lowerbound(0 , 11 , 12) -> 4

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

How many 3 are there ?

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

	▼											
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

		▼										
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

			▼									
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

				▼								
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

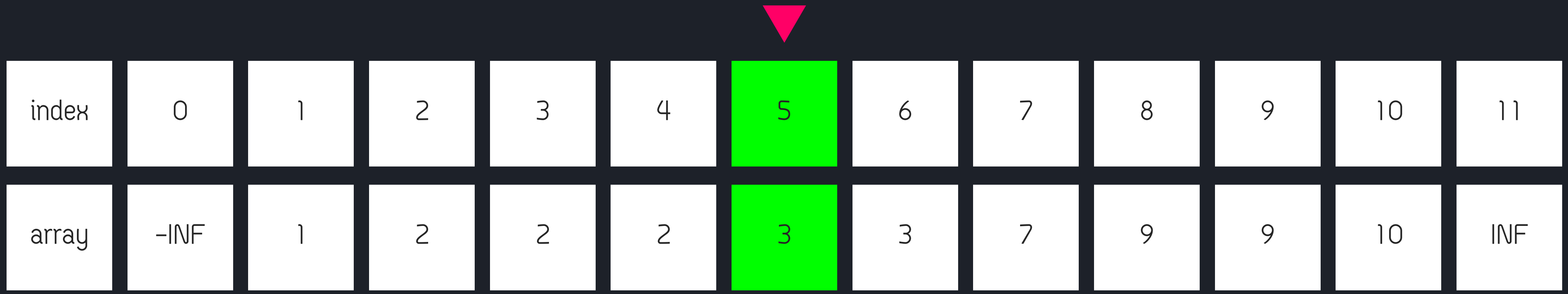
Solution 1 : Linear Search O(n)

Upper and Lower bound

						▼						
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

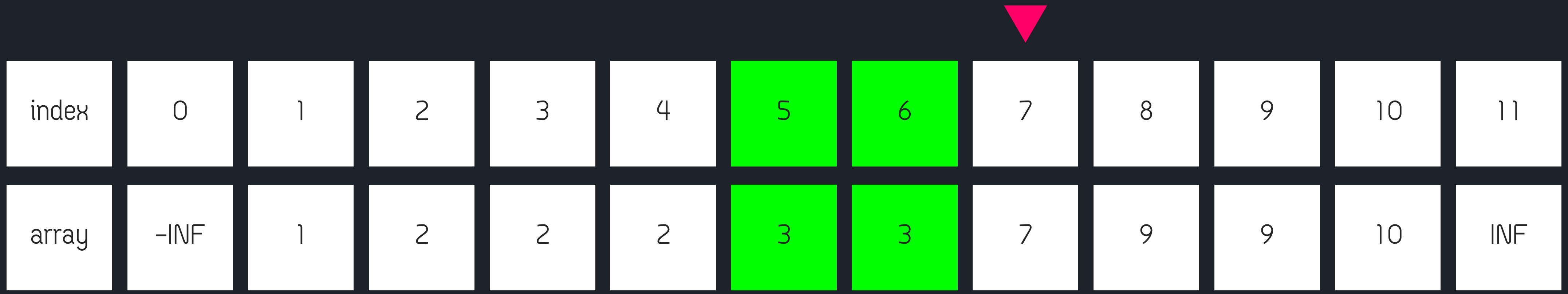
Solution 1 : Linear Search O(n)

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

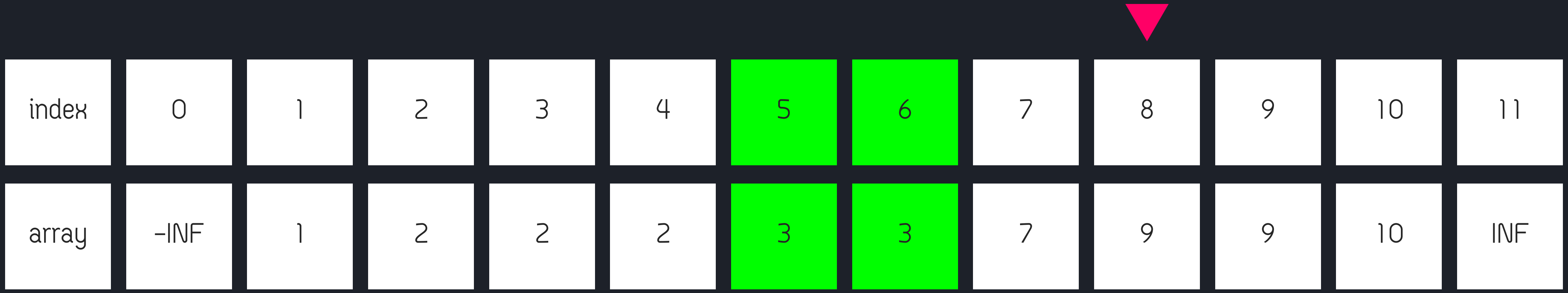
Upper and Lower bound



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

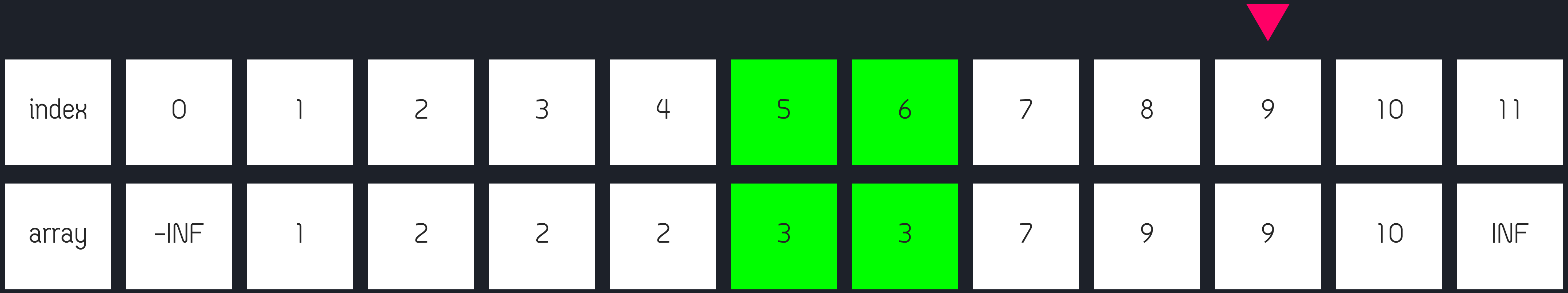
Upper and Lower bound



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

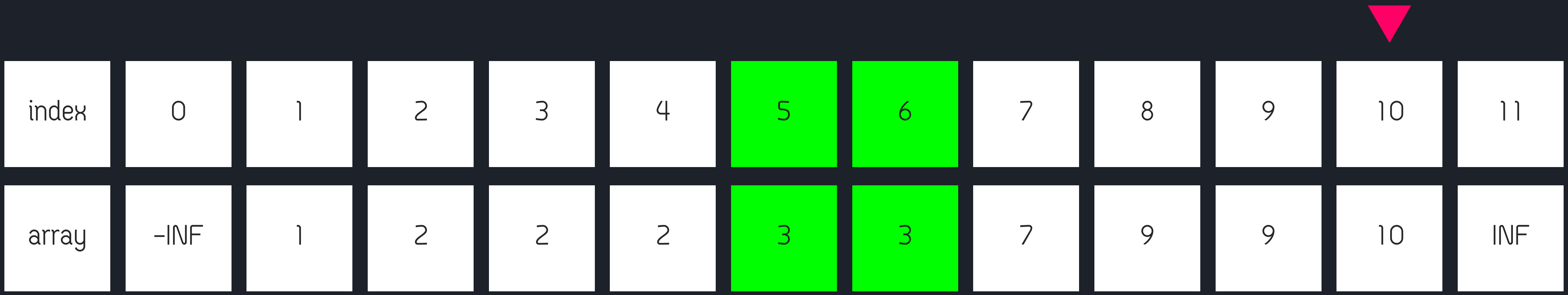
Upper and Lower bound



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound



index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 1 : Linear Search O(n)

Count(3) -> 2

Upper and Lower bound

index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 2 : upper and lower bound $O(\log_2 n)$

Upper and Lower bound

						▼		▼				
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

Solution 2 : upper and lower bound $O(\log_2 n)$

Count(3) -> Upperbound(0 , 11 , 3) - Lowerbound(0 , 11 , 3)

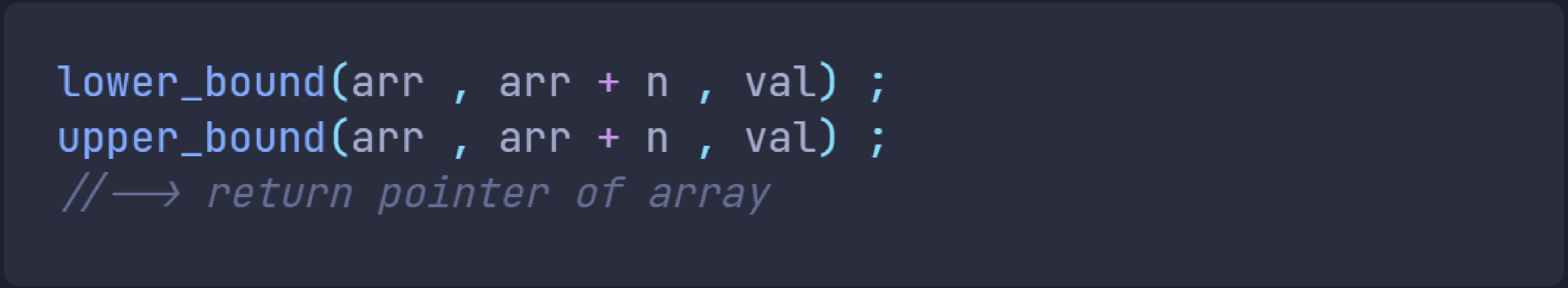
Upper and Lower bound

						▼		▼				
index	0	1	2	3	4	5	6	7	8	9	10	11
array	-INF	1	2	2	2	3	3	7	9	9	10	INF

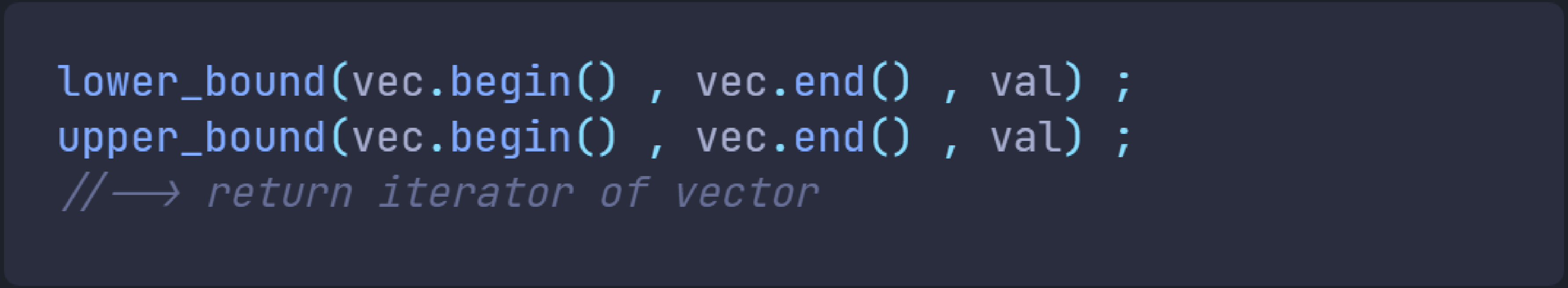
Solution 2 : upper and lower bound $O(\log_2 n)$

Count(3) -> 7 - 5 -> 2

Upper and Lower bound



```
lower_bound(arr , arr + n , val) ;
upper_bound(arr , arr + n , val) ;
//⟶ return pointer of array
```



```
lower_bound(vec.begin() , vec.end() , val) ;
upper_bound(vec.begin() , vec.end() , val) ;
//⟶ return iterator of vector
```

DP Optimization

Longest Increase Subsequence

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10

LIS(array) ?

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10

LIS(array) -> 5

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	**3**	1	**4**	7	2	**6**	**9**	3	**10**

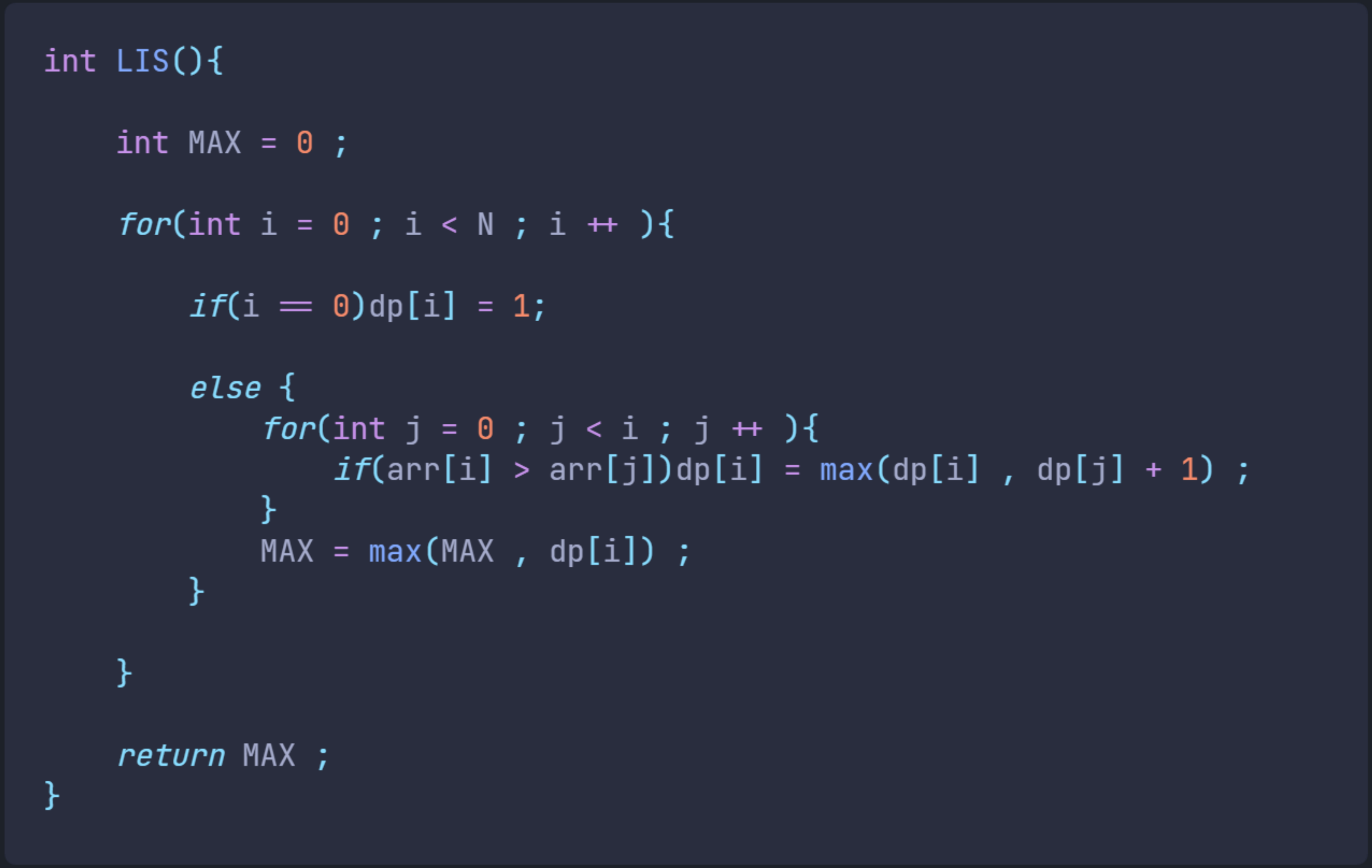
LIS(array) -> 5

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	**1**	**4**	**7**	2	6	**9**	3	**10**

LIS(array) -> 5

DP Optimization



```
int LIS(){

    int MAX = 0 ;

    for(int i = 0 ; i < N ; i ++ ){

        if(i == 0)dp[i] = 1;

        else {
            for(int j = 0 ; j < i ; j ++ ){
                if(arr[i] > arr[j])dp[i] = max(dp[i] , dp[j] + 1) ;
            }
            MAX = max(MAX , dp[i]) ;
        }

    }

    return MAX ;
}
```

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	3								

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1								

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	4							

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	4	7						

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	7						

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	6						

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	6	9					

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	3	9					

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	3	9	10				

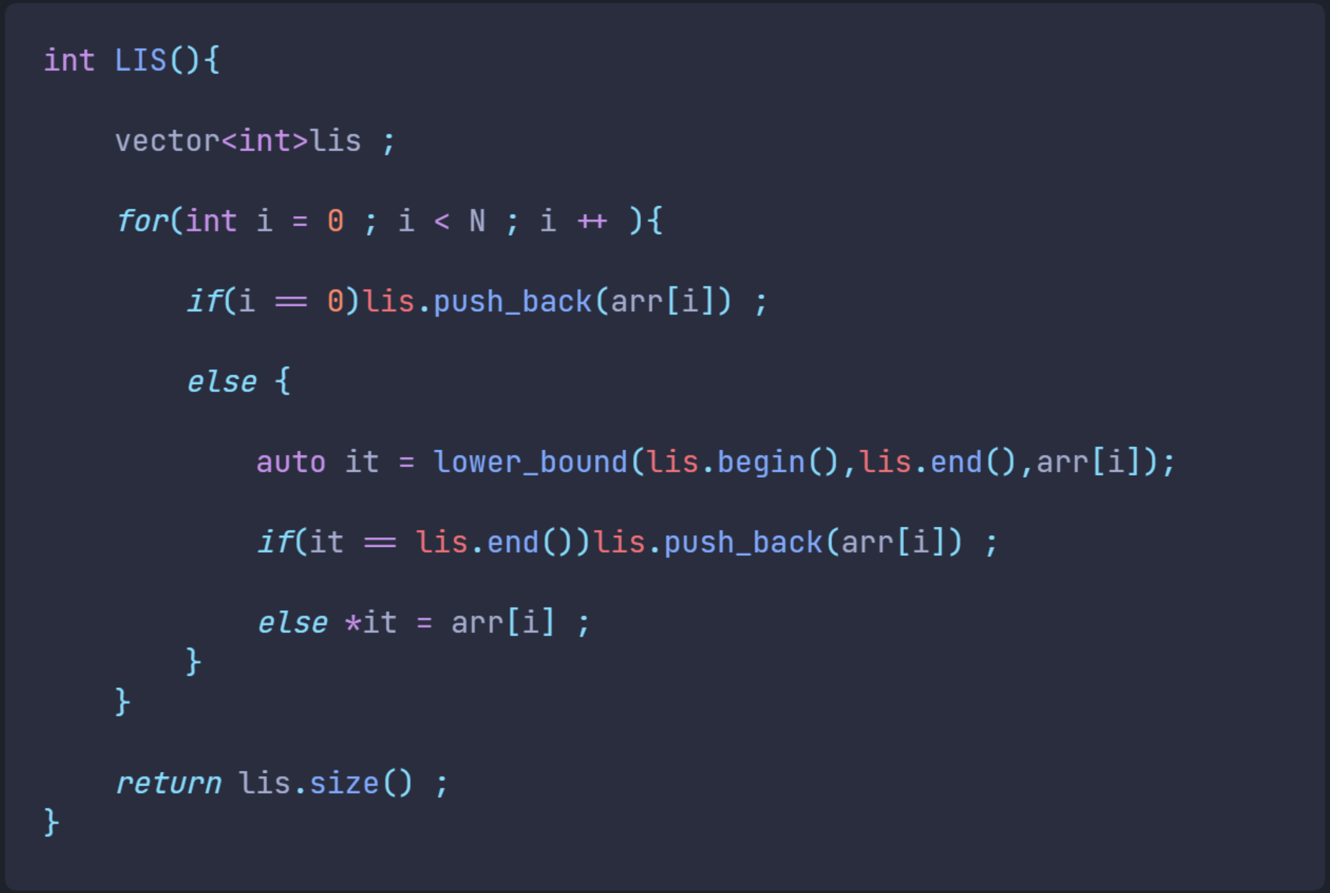
LIS(array) -> 5

DP Optimization

index	0	1	2	3	4	5	6	5	6
array	3	1	4	7	2	6	9	3	10
lis	1	2	3	9	10				

LIS(array) -> 5

DP Optimization



```cpp
int LIS(){

    vector<int>lis ;

    for(int i = 0 ; i < N ; i ++ ){

        if(i == 0)lis.push_back(arr[i]) ;

        else {

            auto it = lower_bound(lis.begin(),lis.end(),arr[i]);

            if(it == lis.end())lis.push_back(arr[i]) ;

            else *it = arr[i] ;
        }
    }

    return lis.size() ;
}
```