2주차

Binary Search, Parametric Search, LIS

신지섭

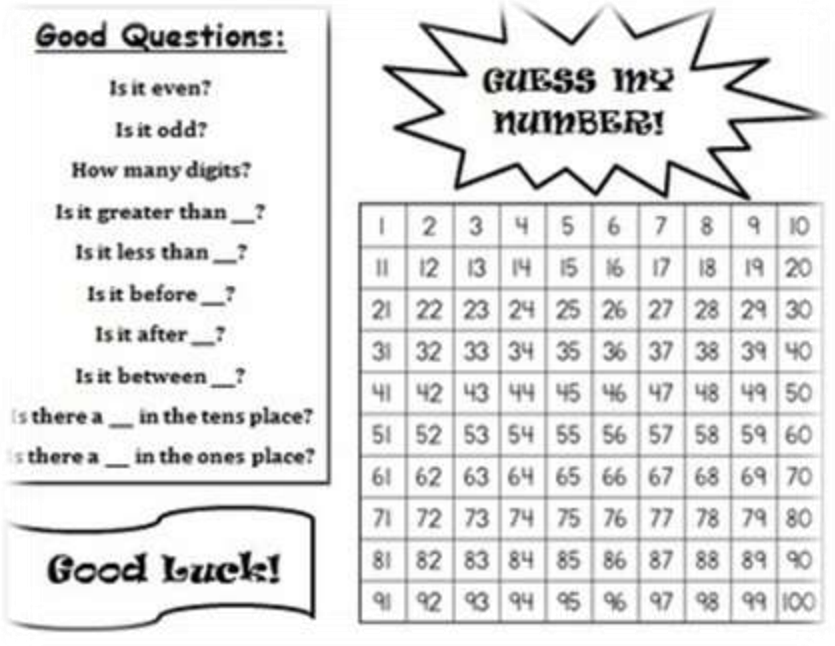
오늘 할 내용은?

1. Binary Search: 선형탐색(Linear Search)보다 빠르게 탐색해 봅시다

2. Parametric Search: Binary Search의 아이디어를 이용해 풀어 봅시다

3. LIS: Binary Search를 응용해 봅시다

Binary Search 이진 탐색 – Guess The Number Game



Q. 1부터 100사이의 숫자 중 x를 찾으려면?

- 방법1: 1부터 차례로 하나씩 물어본다

  => 최악의 경우 100번 물어봐야 한다

- 방법2: 임의의 숫자 k보다 큰지 작은지 물어본다

  ⇒ 가능한 범위의 가운데가 K면 □og100□== 9번만 물어보면 된다

Binary Search 이진 탐색

조건: 수열이 정렬되어 있어야 합니다.

1. Value가 num[mid]보다 크다 => Value가 Mid + 1 ~ Right 구간에 있음

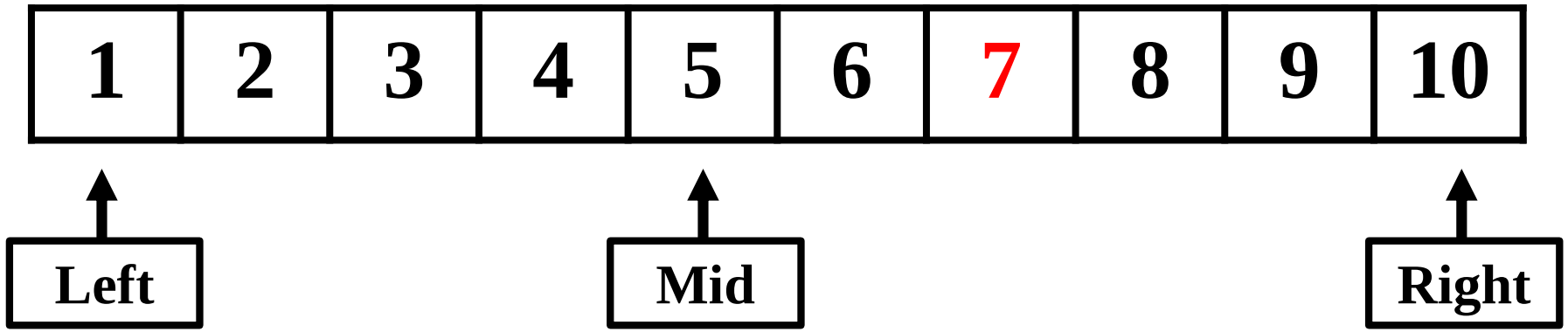
2. Value가 num[mid]보다 작다 => Value가 Left + Mid - 1구간에 있음

3. Value가 num[mid]보다 같다 => 찾았으므로 끝낸다

Mid = (Left + Right) / 2

Binary Search 이진 탐색

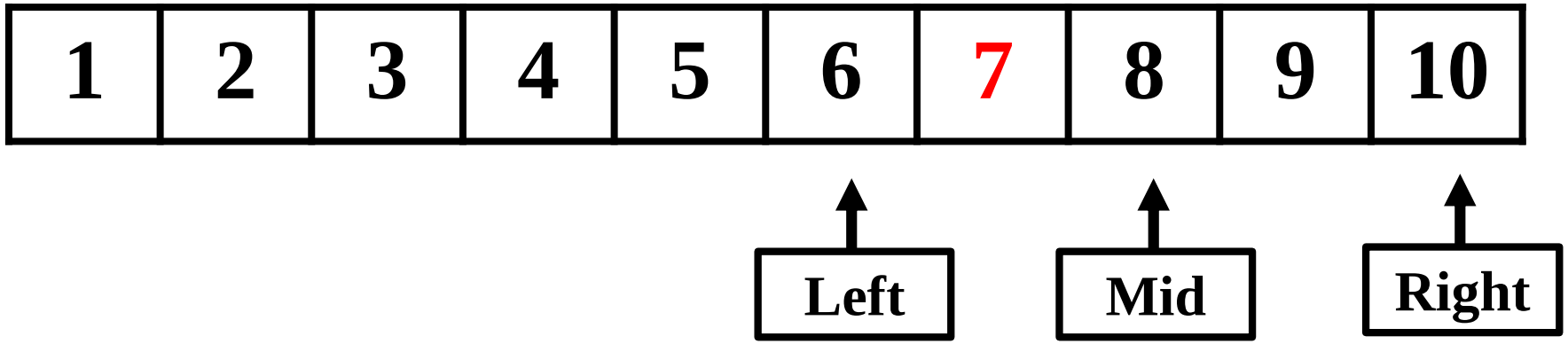
찾고 싶은 값 x: 7



num[mid] < x ⟶ Mid + 1 ~ Right 구간 탐색

Binary Search 이진 탐색

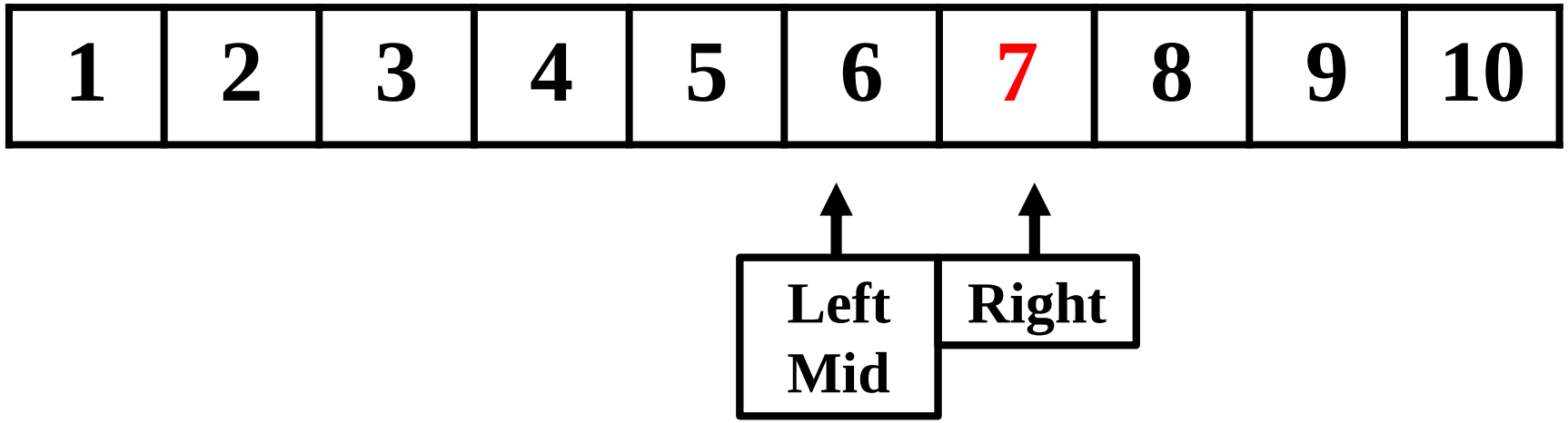
찾고 싶은 값 x: 7



num[mid] > x ⟶ Left ~ Mid – 1 구간 탐색

Binary Search 이진 탐색

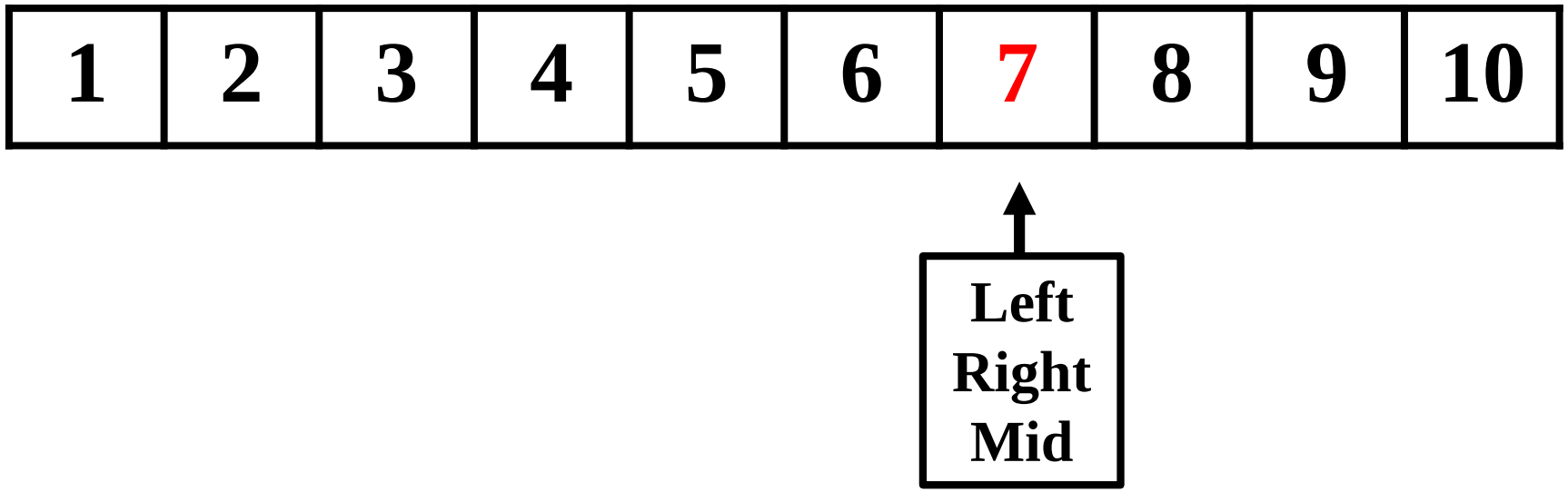
찾고 싶은 값 x: 7



num[mid] < x ⟶ Mid + 1 ~ Right 구간 탐색

Binary Search 이진 탐색

찾고 싶은 값 x: 7



num[mid] == x → 탐색 종료

Binary Search 문제풀이

혹시 푸셨나요?

백준 1920
수 찾기

1920 수 찾기

문제: n개의 수 중에서 x가 존재하는지 확인하면 된다

N <= 100,000 M <= 100,000

⇒ 하나씩 찾아보는 Linear Search를 이용하면 시간 복잡도가 NM로 시간초과!

⇒ **Binary Search**를 이용해 풀자!

1. Main에는 입력을 받고 정렬
2. Binary Search를 하는 함수를 만들자!

수 찾기 코드

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int num[100005];
int n, m;

int bs(int start, int end, int x) {
    if (start > end) { // 범위를 벗어났을 때
        return -1;
    }
    int mid = (start + end) / 2;
    if (x == num[mid]) // 찾으면
        return mid; // 위치를 리턴
    else if (x < num[mid]) // 찾고자 하는 값이 작으면
        return bs(start, mid - 1, x);
    else // 찾고자 하는 값이 크면
        return bs(mid + 1, end, x);
}

```

- 전역변수와 Binary search 함수

```cpp
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> num[i];
    }
    sort(num, num + n); // 이분 탐색에서 정렬은 필수!!
    cin >> m;
    for (int i = 0; i < m; i++) {
        int a; cin >> a;
        if (bs(0, n - 1, a) == -1) { // 존재하지 않으면
            cout << "0\n";
        }
        else {
            cout << "1\n";
        }
    }
    return 0;
}
```

- main

Parametric Search

Binary Search를 응용한 방법

문제를 결정문제로 바꾼 다음, Binary Search를 이용해 푸는 방법입니다

=> Yes / NO로 대답할 수 있는 문제

~~이게 무슨 말이죠?~~

문제를 보면서 이해해 봅시다!!

Parametric Search 문제풀이

일단 풀어보자!

백준 1654
랜선 자르기

1654 랜선 자르기

문제: K개의 랜선을 적당히 잘라서 N개의 랜선을 만든다

이때 최대 랜선의 길이는?

■ 문제를 결정문제로 바꾼다

최대 랜선의 길이는? → 최대 랜선의 길이를 M이라고 하자,

길이가 M인 랜선을 N개 만들 수 있는가? 없는가?

L 길이의 조각을 N개 만들 수 있으면 1~L-1 길이의 조각도 N개 이상 만들 수 있다!

1654 랜선 자르기

Parametric Search로 풀기

1. 이진탐색으로 L 결정
2. L의 길이로 최대 몇 조각을 만들 수 있는지 계산
3. N개 이상의 조각을 만들 수 있다면 => ret에 L값 저장, Left = Mid + 1
4. N개의 조각을 만들 수 없다면 => Right = Mid - 1

반복

※시간복잡도: KlgL

랜선 자르기 코드

```cpp
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    long long e = 0;
    cin >> k >> n;
    for (int i = 0; i < k; i++) {
        cin >> num[i];
        e = max(e, num[i]);
    }
    cout << bs(1, e);
    return 0;
}
```

- main

```cpp
int k, n;
long long num[10005];

long long bs(long long start, long long end) {
    long long ret = 0;
    while (start <= end) {
        long long mid = (start + end) / 2; //overflow 주의
        long long  cnt = 0;
        for (int i = 0; i < k; i++) { // 개수 세기
            cnt += num[i] / mid;
        }
        if (cnt >= n) { // 가능하다면
            ret = max(ret, mid);
            start = mid + 1;
        }
        else //불가능하면
            end = mid - 1;
    }
    return ret;
}
```

- 전역변수와 Binary search 함수

LIS 최장 증가 부분 수열

최장 증가 부분 수열: Longest Increasing Subsequence

일부 원소를 골라내어 만든 부분 수열 중에서, 원소의 값이 증가하고 가장 긴 수열을 말합니다!

ex)

3	8	4	5	6	1	7	2

=> { 3, 4, 5, 6, 7 }

LIS 최장 증가 부분 수열

- LIS 길이를 저장하는 수열을 만들자!

  그 배열을 dp[i]라고 하면...

방법1. i번째 수열을 마지막으로 하는 LIS의 길이

=> O(N^2) 풀이

방법2. 길이가 i인 LIS 중 마지막 수가 가장 작은 LIS의 마지막 수

=> O(Nlog N)풀이

⇒ 예시를 보면서 자세히 알아보자!

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1							
dp	**1**							

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1							
dp	**1**							

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
dp	**1**	**2**						

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
dp	**1**	**2**						

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3					
dp	**1**	**2**	**2**					

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3					
dp	**1**	**2**	**2**					

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
dp	**1**	**2**	**2**	**3**				

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
dp	**1**	**2**	**2**	**3**				

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
dp	**1**	**2**	**2**	**3**	**4**			

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
dp	**1**	**2**	**2**	**3**	**4**			

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6		
dp	**1**	**2**	**2**	**3**	**4**	**1**		

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6		
dp	**1**	**2**	**2**	**3**	**4**	**1**		

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6	7	
dp	**1**	**2**	**2**	**3**	**4**	**1**	**5**	

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6	7	
dp	**1**	**2**	**2**	**3**	**4**	**1**	**5**	

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6	7	8
dp	**1**	**2**	**2**	**3**	**4**	**1**	**5**	**2**

LIS 최장 증가 부분 수열 – $O(N^2)$

num == {3, 8, 4, 5, 6, 1, 7, 2} LIS == { 3, 4, 5, 6, 7 }

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5	6	7	8
dp	**1**	**2**	**2**	**3**	**4**	**1**	**5**	**2**

LIS의 길이: dp 배열에서의 최대값

LIS: dp 배열에서의 최대 위치에서 dp[i]가 1씩 감소하는 인덱스를 구하기

LIS 문제풀이

LIS 구하는 문제이다
$O(N^2)$ 풀이법으로 풀어보자!

백준 11053
가장 긴 증가하는
부분 수열

가장 긴 증가하는 부분 수열 코드

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int n;
int num[1005];
int dp[1005];

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> num[i];
    }
    dp[0] = 1;
    for (int i = 1; i < n; i++) {
        int cnt = 0;
        for (int j = 0; j < i; j++) {
            if (num[j] < num[i] && cnt < dp[j]) { // 이전 숫자보다 크면 업데이트
                cnt = dp[j];
            }
        }
        dp[i] = cnt + 1;
    }
    int an = 0;
    for (int i = 0; i < n; i++) {
        an = max(an, dp[i]);
    }
    cout << an;
    return 0;
}
```

LIS 문제풀이

배열 크기만 늘려서
복사 + 붙여넣기?

백준 12015
가장 긴 증가하는 부분 수열2

12015 가장 긴 증가하는 부분 수열2

31957695	sinphi01	2 12015	시간 초과			C++17 / 수정	502 B	22초 전

$1 \leq n \leq 1{,}000{,}000$

$\Rightarrow O(N^2)$ 으로 풀면 시간초과!

$\Rightarrow O(N\log N)$ 풀이가 필요하다

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index								
DP								

LIS 최장 증가 부분 수열 – O(Nlog N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1							
DP	**3**							

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1							
DP	**3**							

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
DP	**3**	**8**						

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
DP	**3**	**8**						

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
DP	**3**	**4**						

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2						
DP	**3**	**4**						

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3					
DP	**3**	**4**	**5**					

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3					
DP	**3**	**4**	**5**					

LIS 최장 증가 부분 수열 – O(Nlog N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
DP	**3**	**4**	**5**	**6**				

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
DP	**3**	**4**	**5**	**6**				

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
DP	**1**	**4**	**5**	**6**				

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4				
DP	**1**	**4**	**5**	**6**				

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
DP	**1**	**4**	**5**	**6**	**7**			

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
DP	**1**	**4**	**5**	**6**	**7**			

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
DP	**1**	**2**	**5**	**6**	**7**			

LIS 최장 증가 부분 수열 – O(N log N)

num == {3, 8, 4, 5, 6, 1, 7, 2}

index	1	2	3	4	5	6	7	8
num	**3**	**8**	**4**	**5**	**6**	**1**	**7**	**2**

index	1	2	3	4	5			
DP	**1**	**2**	**5**	**6**	**7**			

LIS의 길이: DP 배열의 길이

※ DP 배열의 원소가 LIS의 원소는 **아니다!**

LIS 최장 증가 부분 수열 - 정리

	$O(N^2)$	$O(N \log N)$
구현 방법	바깥쪽 for문으로 전체를 순회하고, 안쪽으로 현재 원소보다 작은 걸 찾음	배열 전체를 순회하면서 lower_bound로 현재 원소보다 크거나 같은 수를 찾음
dp[i]의 정의	num[i]를 마지막으로 하는 LIS의 길이	길이가 i인 LIS 중 마지막 수가 가장 작은 LIS의 마지막 수
dp table의 크기 (M)	N == M	M == (LIS 길이) <= N
알 수 있는 것	LIS와 그 길이	LIS의 길이 **only**

LIS 문제풀이

$O(N\log N)$으로 다시 풀어보자!

백준 12015
가장 긴 증가하는 부분 수열2

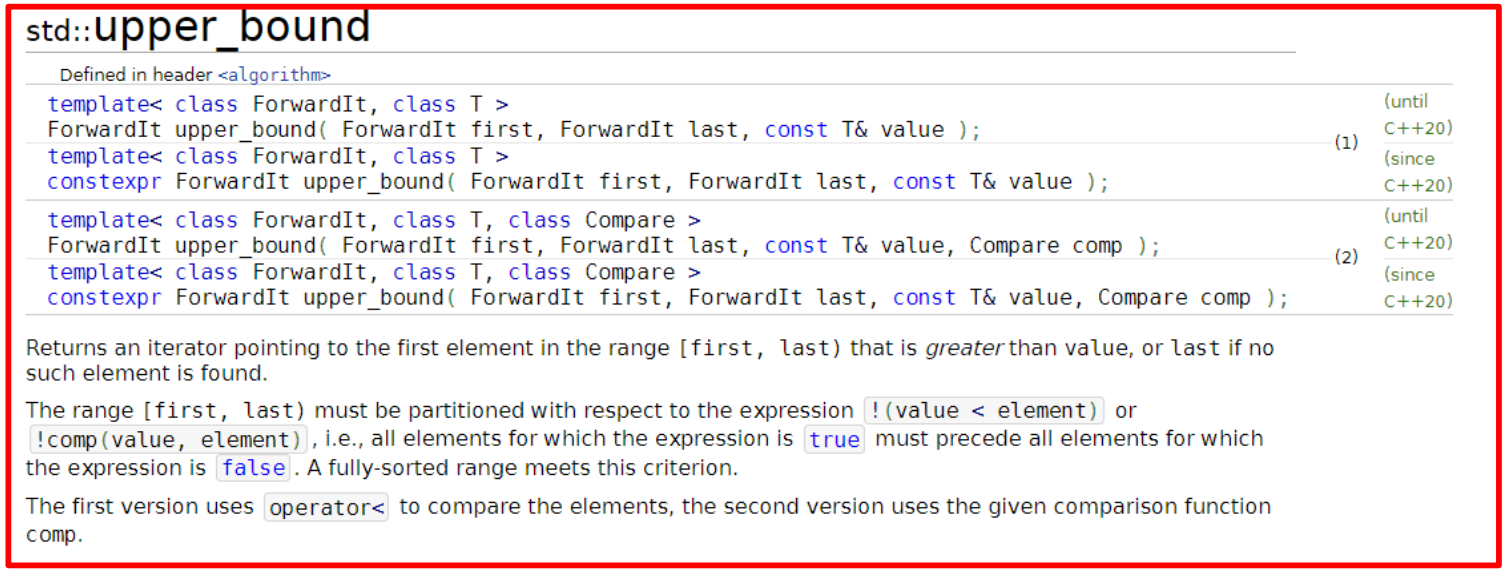
가장 긴 증가하는 부분 수열2 코드

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int n;
int num[1000005];
int dp[1000005];

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> n;
    int cnt = 0; //dp배열의 길이
    for (int i = 0; i < n; i++) {
        cin >> num[i];
    }
    for (int i = 0; i < n; i++) {
        if (cnt == 0 || num[i] > dp[cnt - 1]) { // 넣으려는 값이 크면
            dp[cnt] = num[i];
            cnt++;
        }
        else {
            int index = (lower_bound(dp, dp + cnt, num[i]) - dp); // 적당한 위치 찾기
            dp[index] = num[i];
        }
    }
    cout << cnt;
    return 0;
}
```

참고 - upper_bound



std::upper_bound

Defined in header <algorithm>

```
template< class ForwardIt, class T >
ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value );                    (1)  (until C++20)
template< class ForwardIt, class T >
constexpr ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value );          (1)  (since C++20)

template< class ForwardIt, class T, class Compare >
ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value, Compare comp );      (2)  (until C++20)
template< class ForwardIt, class T, class Compare >
constexpr ForwardIt upper_bound( ForwardIt first, ForwardIt last, const T& value, Compare comp );  (2)  (since C++20)
```

Returns an iterator pointing to the first element in the range `[first, last)` that is *greater* than `value`, or `last` if no such element is found.

The range `[first, last)` must be partitioned with respect to the expression `!(value < element)` or `!comp(value, element)`, i.e., all elements for which the expression is `true` must precede all elements for which the expression is `false`. A fully-sorted range meets this criterion.

The first version uses `operator<` to compare the elements, the second version uses the given comparison function `comp`.

https://en.cppreference.com

upper_bound: 오름차순 정렬되어 있는 배열의 구간 [first, last) 에서 처음으로 x 를 초과하는 수가 나타나는 위치를 찾아주는 함수

lower_bound: 오름차순 정렬되어 있는 배열의 구간 [first, last) 에서 처음으로 x 이상인 수가 나타나는 위치를 찾아주는 함수

참고 - upper_bound

```cpp
int num[1000005];

int upper_bound(int n, int x) {
    int start = 0;
    int end = n;

    while(end > start) {
        int mid = (start + end) / 2;

        if (num[mid] < x) { // x가 크면
            start = mid + 1; //시작점을 올림
        }
        else { // x가 중앙값보다 작거나 같으면
            end = mid; // 끝점을 내림
        }
    }
    return end + 1; //다음 위치 리턴
}
```

{1, 2, 3, 4, 5, 6, 7, 8}

cout << upper_bound(8, 4) << “\n”; //4

cout << lower_bound(8, 4) << “\n”; //3

Problem Set – Binary Search

2792 보석 상자

3020 개똥벌레

Problem Set – Parametric Search

2512 예산

2805 나무 자르기

1300 K번째 수

Problem Set – LIS

14002 가장 긴 증가하는 부분 수열4

11054 가장 긴 바이토닉 부분 수열

3745 오름세

2352 반도체 설계

Problem Set – 도전!

2613 숫자구슬

조건을 잘 보고 parametric search를 해보자

2532 먹이사슬

LIS를 사용하기 위해 전처리(정렬)가 필요한 문제