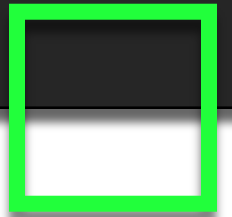


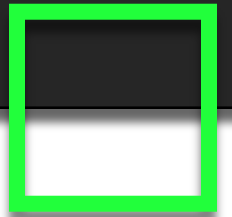
#알고리즘반
#4주차

LIS & LCS

T. 우한샘



#1 LIS





LIS란?

Longest Increasing Subsequence

Longest
Increasing
Subsequence

최장
증가
부분수열



부분수열이 뭔가요?

어떤 수열의 원소 일부를 뽑아내 만든 수열!
연속한 원소는 아니여도 되지만,
순서는 같아야 합니다!

ex) $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$\{1, 3, 5, 9\} \subset \text{Sub}(A)$

$\{1, 5, 3, 9\} \not\subset \text{Sub}(A)$



부분수열이 뭘까요?

다음 수열 중 수열 $A = \{2, 3, 5, 11, 7\}$ 의
부분수열은 무엇일까요?

- ① 수열 $B = \{2, 3, 5\}$
- ② 수열 $C = \{2, 4, 5\}$
- ③ 수열 $D = \{2, 7, 11\}$



증가 부분수열은 뭔가요?

부분수열 중 모든 원소가 증가하는 순서로 이루어진 부분수열!

ex) $A = \{2, 3, 1, 4, 7, 5, 6, 8, 9\}$

$\{2, 3, 5, 9\} \subset IS(A)$

$\{2, 3, 1, 9\} \not\subset IS(A)$



그럼 LIS(최장 증가 부분수열)은요?

증가 부분수열 중 가장 긴 부분수열!
(주의: 하나가 아닐 수 있음!)

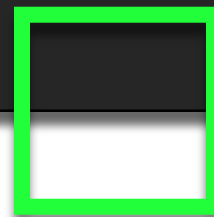
ex) $A = \{2, 3, 1, 7, 5, 4, 6, 8, 9\}$

$\{2, 3, 5, 6, 8, 9\} \subset \text{LIS}(A)$

$\{2, 3, 4, 6, 8, 9\} \subset \text{LIS}(A)$

#2

LIS의 길이 구하기 ($O(N^2)$)



LIS 길이는 어떻게 구할 수 있을까요?

- 전체 문제

길이 N의 수열 안에서 LIS의 길이를 구하는 것

- 부분 문제

길이 $K(1 \leq K < N)$ 의 수열 안에서의 LIS 길이를 구하는 것

- DP 배열

$DP[N] = DP[1] \sim DP[N]$ 을 포함하는 부분 수열 안에서, $DP[N]$ 을 마지막 원소로 갖는 LIS의 길이

LIS 길이는 어떻게 구할 수 있을까요?

이중 for 문을 돌리는 구현 방식이 가장 편합니다! ($O(n^2)$)

1. 바깥쪽 for 문으로 $DP[1] \sim DP[n]$ 를 구한다
2. 안쪽 for 문으로 $DP[1] \sim DP[i-1]$ 를 돌면서
arr[i]보다 작은 원소를 끝값으로 가지는
 $DP[j]$ 값 중 최대 길이를 찾아준다.
=> $DP[i]$ 에는 그 최댓값 + 1을 넣어줌!

헛갈리실 테니까 예시로 보여드리면...




수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value							




헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

							
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
							
DP[i]	1	2	3	4	5	6	7
Value							




헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

							
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
							
DP[i]	1	2	3	4	5	6	7
Value	1						




헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

							
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
							
DP[i]	1	2	3	4	5	6	7
Value	1						




헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

							
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
							
DP[i]	1	2	3	4	5	6	7
Value	1	2					


헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$


							
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
							
DP[i]	1	2	3	4	5	6	7
Value	1	2					

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$





arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$





arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$




arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2			

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$




arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2			

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3		

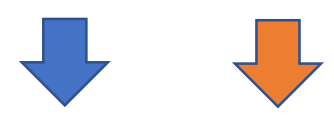
헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$


					↓	↓	
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1
						↓	
DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3		

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



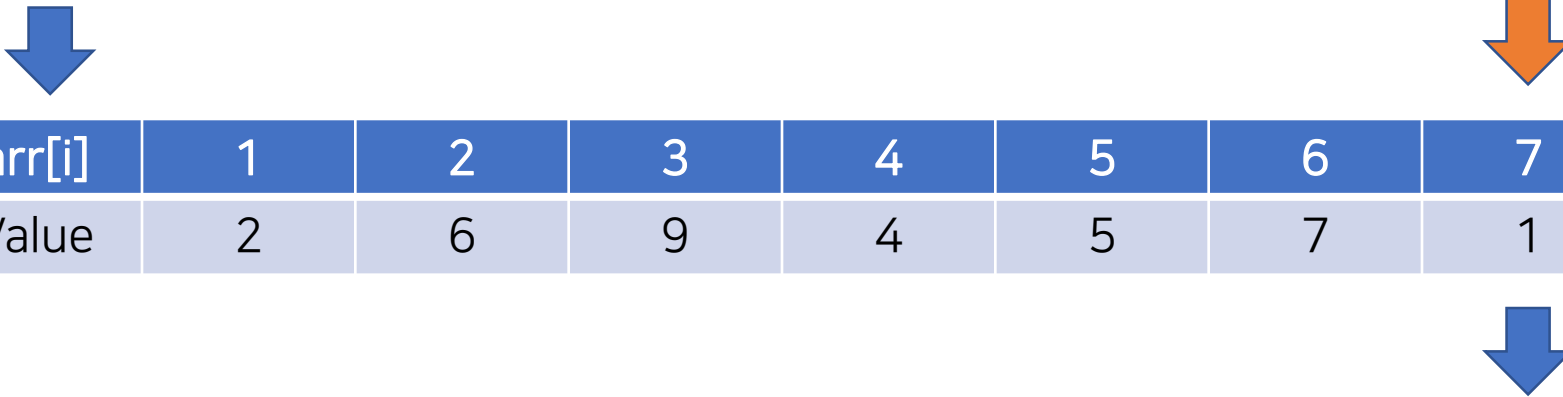
arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

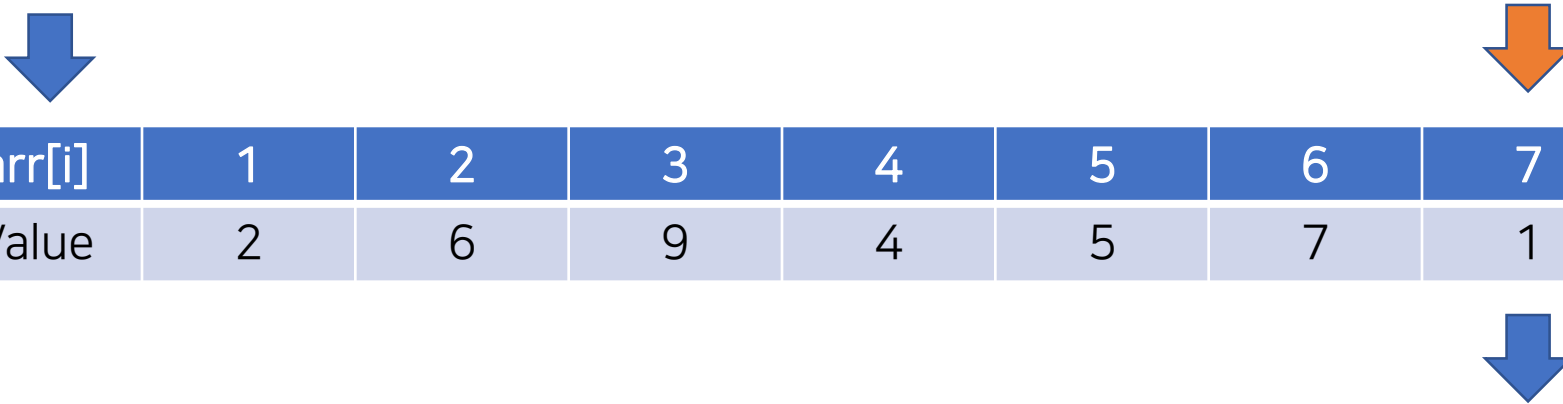


arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

DP 배열에 저장된 값 중 가장 큰 값이 바로 LIS의 길이!

C로 구현해 볼까요?

```
1 #include <stdio.h>
2 #define MAX(a, b) ((a) > (b) ? (a) : (b))
3 int arr[1001], DP[1001], n, maxVal;
4 int main() {
5     scanf("%d", &n);
6     for (int i = 1; i <= n; i++) scanf("%d", arr + i);
7     for (int i = 1; i <= n; i++) {
8         int curNum = arr[i];
9         int curMaxPos = 0;
10        for (int j = 1; j < i; j++) {
11            if (arr[j] < curNum && DP[j] > DP[curMaxPos])
12                curMaxPos = j;
13        }
14        maxVal = MAX(maxVal, (DP[i] = DP[curMaxPos] + 1));
15    }
16
17    printf("%d", maxVal);
18    return 0;
19 }
```

7
2 6 9 4 5 7 1
4
계속하려면 아무 키나
누르십시오...



연습 문제

11053 : 가장 긴 증가하는 부분수열
2565 : 전깃줄

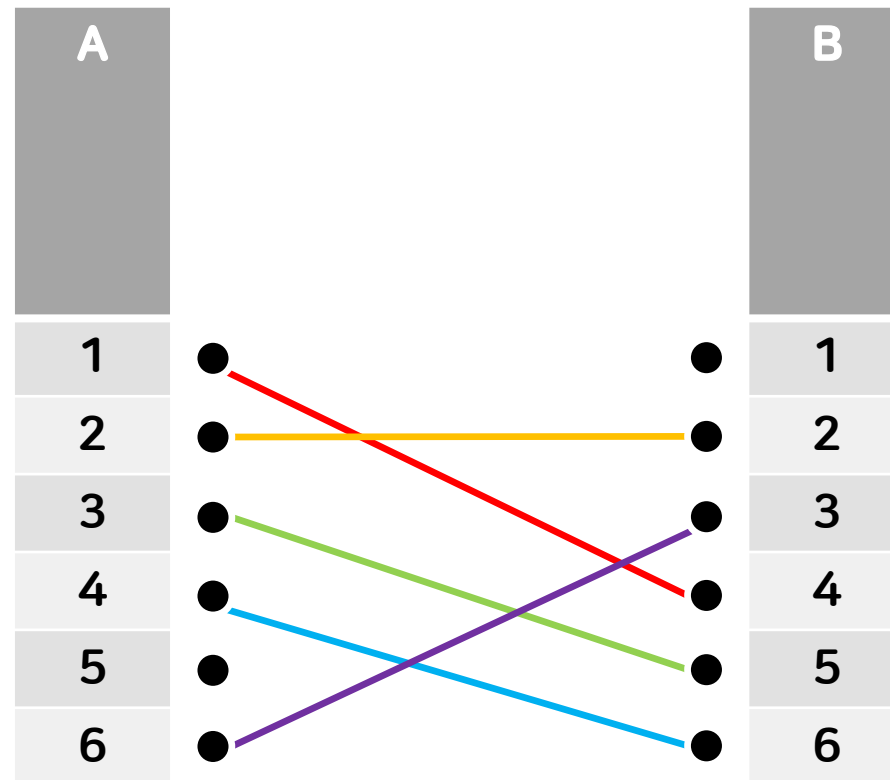
11053 : 가장 긴 증가하는 부분수열

수업을 잘 따라오셨다면 바로 푸실 수 있습니다!

힌트:
26번 슬라이드에 있는 코드를 제출란에 붙여넣으면
어떤 일이 일어날까요...?

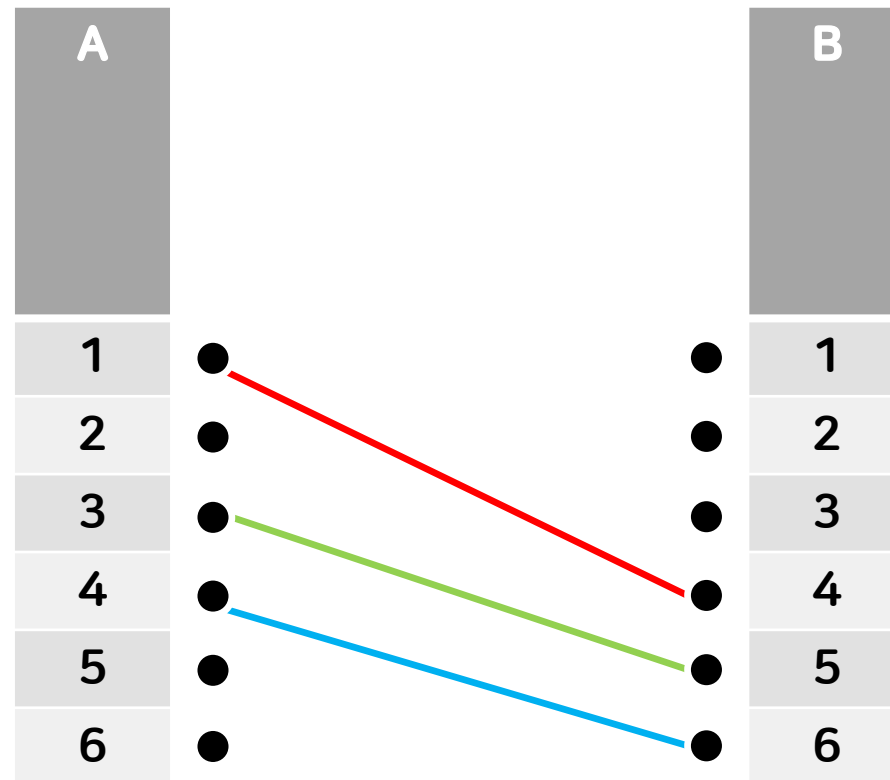
2565 : 전깃줄

전깃줄이 꼬이지 않으려면 어떤 조건이 필요할까요?



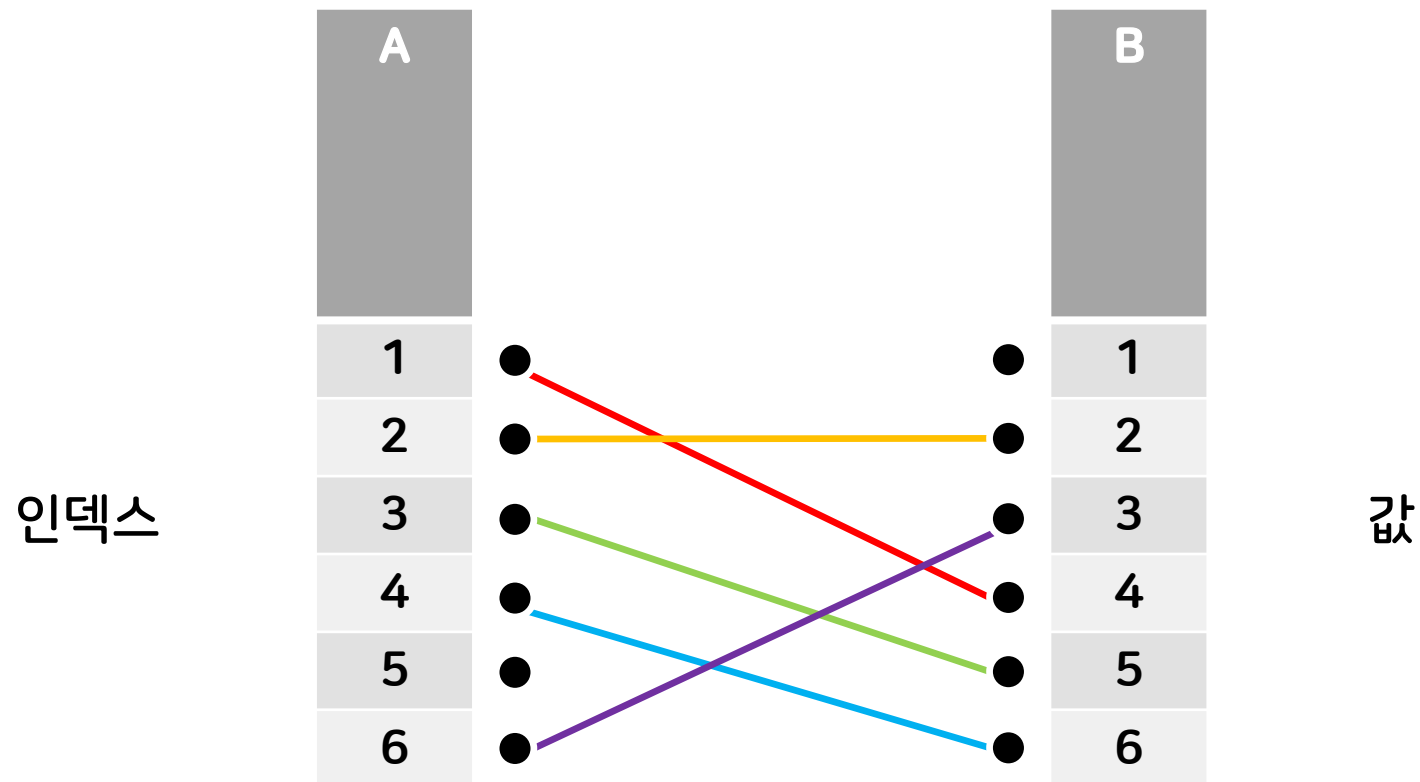
2565 : 전깃줄

전깃줄이 꼬이지 않으려면 어떤 조건이 필요할까요?



2565 : 전깃줄

전깃줄이 꼬이지 않으려면 어떤 조건이 필요할까요?



2565 : 전깃줄

이를 이용하여 LIS의 길이를 구할 경우, 겹치지 않고 배치 가능한 가장 많은 전깃줄의 개수가 된다.
따라서, 없애야 하는 전깃줄의 개수는 (전체 전깃줄 개수) - (LIS의 길이)가 된다!

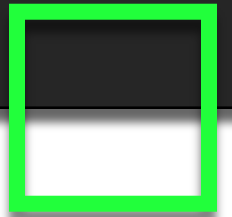
* 주의: arr[i]가 0인 경우는 무시하도록 예외처리 꼭 해주세요!

arr[i]	0	1	2	3	4	5	6
Value	0	4	2	5	6	0	3

DP[i]	0	1	2	3	4	5	6
Value	0	1	1	2	3	0	2

#3

LIS with Tracking ($O(N^2)$)



길이는 알아냈지만... 실제 LIS는 어떻게?

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

여기서 LIS는 어떤 원소를 골라야 나올까요?

길이는 알아냈지만... 실제 LIS는 어떻게?

LIS = {2, 4, 5, 7}

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

그럼 이 index들의 DP 배열 값을 볼까요?

길이는 알아냈지만... 실제 LIS는 어떻게?

LIS = {2, 4, 5, 7}

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

길이는 알아냈지만... 실제 LIS는 어떻게?

LIS = {2, 4, 5, 7}

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

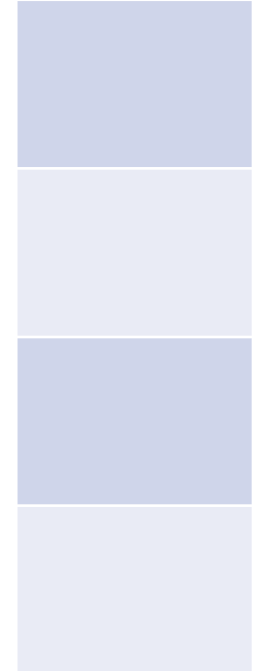
LIS의 길이를 저장해 두고 DP 배열을 역으로 순회하면서,
현재 저장된 숫자보다 더 큰 값이 올 때마다 stack에
push하고 저장된 값에서 1을 빼준다!

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 4

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

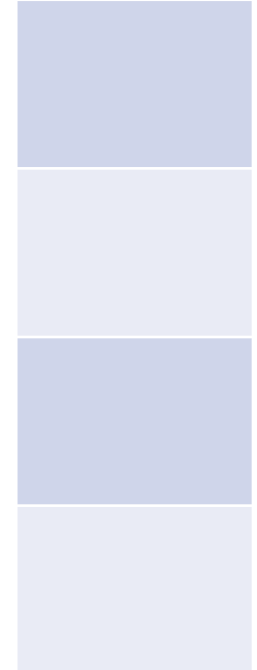
Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 4

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

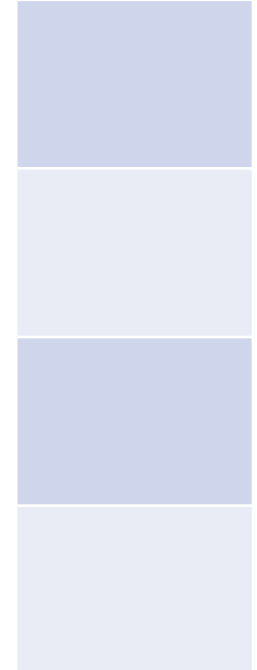
Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 4

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1




stack

Stack을 이용한 LIS with Tracking

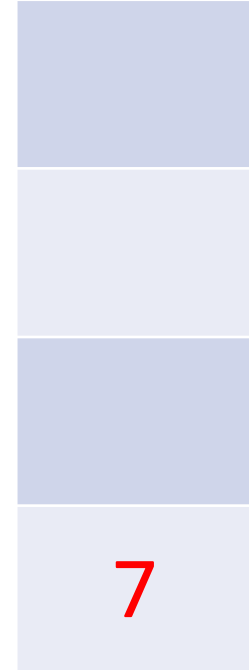
LIS_length = 4
curPos = 3



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

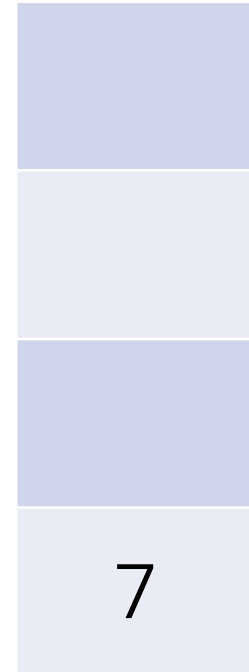
Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 3

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1




DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1




stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 2



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 2

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 1



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 1

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 1

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 1

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1



stack

Stack을 이용한 LIS with Tracking

LIS_length = 4
curPos = 0



arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1



DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

2
4
5
7

stack

Stack을 이용한 LIS with Tracking

반복문이 완료되었을 때 stack을 차례대로 출력시키면
LIS의 원소들이 출력됩니다!

arr[i]	1	2	3	4	5	6	7
Value	2	6	9	4	5	7	1

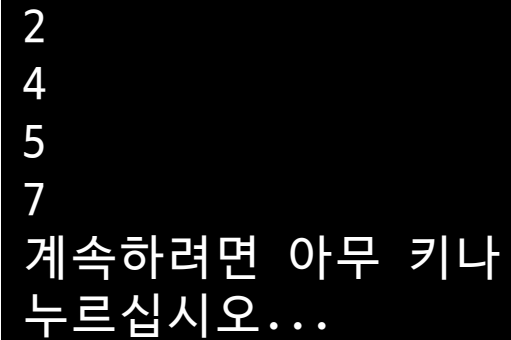
DP[i]	1	2	3	4	5	6	7
Value	1	2	3	2	3	4	1

2
4
5
7

stack

Stack을 이용한 LIS with Tracking

```
1 int arr[] = {0, 2, 6, 9, 4, 5, 7, 1};
2 int DP[] = {0, 1, 2, 3, 2, 3, 4, 1};
3 stack<int> LIS;
4 int curPos = 4;
5 for (int i = 7; i > 0; i--) {
6     if (DP[i] == curPos) {
7         LIS.push(arr[i]);
8         curPos--;
9     }
10 }
11 while (!LIS.empty()) {
12     printf("%d\n", LIS.top());
13     LIS.pop();
14 }
```



```
2
4
5
7
계속하려면 아무 키나
누르십시오...
```



연습 문제

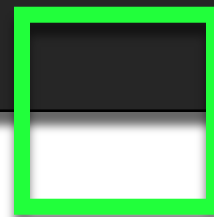
14002 : 가장 긴 증가하는 부분 수열 4

14002 : 가장 긴 증가하는 부분 수열 4

11053번(가장 긴 증가하는 부분수열) 문제에서
Tracking만 넣어주면 바로 풀 수 있는 문제입니다!

#4

LIS의 길이 구하기 ($O(N \log N)$)



와 그러면 이걸로 LIS 문제 다 풀면 되네요?

네... 저도 그런 줄 알았습니다.

앞에 배운 코드로 바로
다음 문제를 풀어봤어요!

12015 : 가장
긴 증가하는
부분 수열 2

바로 맞추겠죠..?

문제

수열 A 가 주어졌을 때, 가장 긴 증가하는 부분 수열을 구하는 프로그램을 작성하시오.

예를 들어, 수열 $A = \{10, 20, 10, 30, 20, 50\}$ 인 경우에 가장 긴 증가하는 부분 수열은 $A = \{10, 20, 10, 30, 20, 50\}$ 이고, 길이는 4이다.

입력

첫째 줄에 수열 A 의 크기 N ($1 \leq N \leq 1,000,000$)이 주어진다.

둘째 줄에는 수열 A 를 이루고 있는 A_i 가 주어진다. ($1 \leq A_i \leq 1,000,000$)

출력

첫째 줄에 수열 A 의 가장 긴 증가하는 부분 수열의 길이를 출력한다.

와 그러면 이걸로 LIS 문제 다 풀면 되네요?

felis

12015

시간 초과

외양도

첫째 줄에 수열 A 의 크기 N ($1 \leq N \leq 1,000,000$)이 주어진다.



편한 방법 == 높은 시간복잡도

이중 for 문을 돌리는 구현 방식이 가장 편합니다! ($O(n^2)$)
그런데 n 이 1,000,000이라면...?

$$n^2 = (1.0E6)^2 = 1.0E12$$

$$= 1,000,000,000,000$$





lower_bound(First, Last, val)

정렬되어 있는 배열에서 찾고자 하는 수 val 이상인 수가 처음으로 나타나는 위치를 Binary Search로 찾아주는 함수 표준 헤더 <algorithm>에 들어있습니다.

시간복잡도: $O(\log N)$

lower_bound(First, Last, val)

arr[i]	0	1	2	3	4
Value	5	13	30	54	56

lower_bound(arr, arr + 5, 6)

-> arr + 1

lower_bound(arr, arr + 5, 54)

-> arr + 3

lower_bound(arr, arr + 5, 100)

-> 해당하는 위치가 없으므로 arr + 5 반환!

이걸 응용해서 LIS 문제를 풀어봅시다! ($O(N \log N)$)

- 전체 문제

길이 N 의 수열 안에서 LIS의 길이를 구하는 것

- 부분 문제

길이 K ($1 \leq K < N$)의 수열 안에서의 LIS 길이를 구하는 것

- DP 배열

$DP[i]$ = 길이 i 의 LIS중, 마지막 수가 가장 작은 LIS의
마지막 수

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$

arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1

DP[i]							
Value							

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]							
Value							

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0						
Value	2						

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0						
Value	2						

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1					
Value	2	6					

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1					
Value	2	6					

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	6	9				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	6	9				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	4	9				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	4	9				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	4	5				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2				
Value	2	4	5				

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$




arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1




DP[i]	0	1	2	3			
Value	2	4	5	7			

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$




arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1




DP[i]	0	1	2	3			
Value	2	4	5	7			

헛갈리실 테니까 예시로 보여드리면...

수열 $arr = \{2, 6, 9, 4, 5, 7, 1\}$



arr[i]	0	1	2	3	4	5	6
Value	2	6	9	4	5	7	1



DP[i]	0	1	2	3			
Value	1	4	5	7			

헛갈리실 테니까 예시로 보여드리면...

DP[i]	0	1	2	3			
Value	1	4	5	7			

DP 배열의 길이가 곧 LIS의 길이!
하지만, 수를 계속 덮어쓰우면서 DP 배열을 갱신했으므로
(DP 배열의 원소) \neq (LIS의 원소) 입니다!!

C++로 구현해 볼까요?

```
1 int n, arr[1000000];
2 vector<int> DP;
3 scanf("%d", &n);
4 for (int i = 0; i < n; i++) scanf("%d", arr + i);
5 for (int i = 0; i < n; i++) {
6     auto pos = lower_bound(DP.begin(), DP.end(), arr[i]);
7     // arr[i]의 값보다 크거나 같은 값이 DP 배열에 없을 경우
8     // DP 배열의 끝에 추가해줌
9     if (pos == DP.end())
10         DP.push_back(arr[i]);
11     // 하지만 그 값이 존재할 경우, 해당 위치에 arr[i]를 덮어씌웁!
12     else
13         DP[pos - DP.begin()] = arr[i];
14 }
15 printf("%d", DP.size());
```

7
2 6 9 4 5 7 1
4
계속하려면 아무 키나
누르십시오...

LIS 길이 구하기 비교 - $O(N^2)$ vs $O(N\log N)$

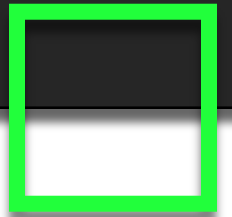
	$O(N^2)$	$O(N\log N)$
구현법	바깥쪽 for문으로 전체를 순회하면서 ($O(N)$), 안쪽 for문으로 현재 원소보다 작은걸 찾음 ($O(N)$)	배열 전체를 순회하면서 ($O(N)$), lower_bound로 현재 원소보다 크거나 같은 수를 찾음 ($O(\log N)$)
DP[i]의 의미	arr[i]를 마지막으로 하는 LIS 길이	길이가 i인 LIS중 마지막 수가 가장 작은 LIS의 마지막 수
DP Table의 크기 M	$N == M$	$M == (\text{LIS 길이}) \leq N$
DP Table의 용도	DP Table로 LIS와 그 길이를 모두 구할 수 있다!	DP Table로는 LIS의 길이만 구할 수 있다!



연습 문제

12015 : 가장 긴 증가하는 부분 수열 2

#5 LCS





LCS란?

Longest Common Subsequence

Longest	최장
Common	공통
Subsequence	부분수열

공통 부분수열이 뭔가요?

두 수열(또는 문자열) A, B를 모두 이용해야
만들 수 있는 부분수열

$$\begin{cases} str1: ADLC OYHPA \\ str2: CAPLODHA \end{cases}$$

ADLC OYHPA
CAPLODHA

APA

ADLC OYHPA
CAPLODHA

COHA

ADLC OYHPA
CAPLODHA

ALOHA

그럼 LCS(최장 공통 부분수열)은요?

공통 부분수열 중 가장 긴 부분수열!

$$\begin{cases} str1: ADLC OYHPA \\ str2: CAPLODHA \end{cases}$$

ADLC OYHPA
CAPLODHA

APA

ADLC OYHPA
CAPLODHA

COHA

ADLC OYHPA
CAPLODHA

ALOHA

LCS는 어떻게 구할 수 있을까요?

- 전체 문제

두 개의 수열(또는 문자열)에서 LCS를 구하는 것

- 부분 문제

길이 i ($1 \leq i < M$)인 수열과 길이 j ($1 \leq j < N$)인 수열 안에서의 LCS 길이를 구하는 것

- DP 배열

$DP[i][j]$ = 문자열 A의 i 번째 원소, 문자열 B의 j 번째 원소까지의 LCS 길이

$\begin{cases} str1: ADLC OYHPA \\ str2: CAPLODHA \end{cases}$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0

$\begin{cases} str1: ADLC OYHPA \\ str2: CAPLODHA \end{cases}$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1

$str1[i] \neq str2[j]$ 인 경우
 $\Rightarrow DP[i][j] = \max(DP[i-1][j], DP[i][j-1])$

$str1[i] == str2[j]$ 인 경우
 $\Rightarrow DP[i][j] = DP[i-1][j-1] + 1$

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4

$\left\{ \begin{array}{l} str1: ADLC OYHPA \\ str2: CAPLODHA \end{array} \right.$

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

DP[M][N] 의 값이 바로 LCS의 길이!

일단 여기까지만 C로 구현해 봅시다!

```
1 char str1[] = "ALDCOYHPA"; // (len1 = strlen(str1)) == 9
2 char str2[] = "CAPLODHA"; // (len2 = strlen(str2)) == 8
3 int DP[9 + 1][8 + 1] = { {0}, };
4 for (int i = 1; i <= len1; i++) {
5     for (int j = 1; j <= len2; j++) {
6         if [redacted]
7             DP[i][j] = [redacted]
8         else
9             DP[i][j] = [redacted]
10    }
11 }
12 printf("%d\n", DP[len1][len2]);
13
14
15
```

5

계속하려면 아무 키나
누르십시오...



연습 문제

9251 : LCS

DP 배열은 완성했는데... LCS는 어디 있죠?

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

DP 배열은 완성했는데... LCS는 어디 있죠?

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

DP 배열은 완성했는데... LCS는 어디 있죠?

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

DP 배열은 완성했는데... LCS는 어디 있죠?

	∅	C	A	P	L	O	D	H	A
∅	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
D	0	0	1	1	1	1	2	2	2
L	0	0	1	1	2	2	2	2	2
C	0	1	1	1	2	2	2	2	2
O	0	1	1	1	2	3	3	3	3
Y	0	1	1	1	2	3	3	3	3
H	0	1	1	1	2	3	3	4	4
P	0	1	1	1	2	3	3	4	4
A	0	1	2	2	2	3	3	4	5

DP[M][N] 에서 DP[0][0] 전까지 거슬러 가면서, DP[i][j] 가 DP[i - 1][j] 및 DP[i][j - 1] 의 값과 다를 때까지 찾아간다!

그럼 이제 코드를 마저 구현해 볼까요?

```
1 char LCS_str[9 + 1] = "\\0";
2 int i = len1, j = len2, curLength = DP[len1][len2];
3 while (curLength != 0) {
4     if (DP[i][j] > DP[i - 1][j] && DP[i][j] > DP[i][j - 1]) {
5         LCS_str[curLength - 1] = str1[i - 1];
6         curLength--;
7         i--;
8         j--;
9     }
10    else if (DP[i][j] > DP[i - 1][j] && DP[i][j] == DP[i][j - 1])
11        j--;
12    else
13        i--;
14 }
15 printf("%s\\n", LCS_str);
```

ALOHA

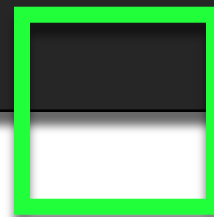
계속하려면 아무 키나
누르십시오...



연습 문제

9252 : LCS 2

#6 과제





과제

1365 : 꼬인 전깃줄

2631 : 줄세우기

1965 : 상자넣기

12738: 가장 긴 증가하는 부분 수열 3

7476: 최대 공통 증가 수열

1958: LCS 3

수고하셨습니다!

