

2024 겨울학기 동국대학교SW역량강화캠프

17일차. 다이나믹 프로그래밍 2

### ● 다이나믹 프로그래밍(DP)

- ▶ 값의 재활용 (중복되는 계산은 한 번만)
- ▶ 큰 문제를 작은 문제들로 나누어 해결하는 방법
- ▶ 동일한 형태의 문제들끼리의 관계식을 찾는 것이 핵심

### ● 증가하는 가장 긴 수열 찾기(4079)

#### | 문제

복극곰 율리는 주어진 수열에서 몇개를 뽑아 증가하는 수열을 만들려고 한다.

수열의 순서는 바꿀 수 없다.

수열 {10, 40, 20, 50, 30} 이 있을 경우, 증가하는 수열의 최대길이는 {10, 20, 30}으로 3이다.

율리를 도와 증가하는 수열의 최대 길이를 구하자.

#### | 입력

첫째 줄에 수열의 길이  $N$ 이 주어진다. ( $1 \leq N \leq 1000$ )

둘째 줄에는 수열을 이루는 값  $X_i$ 들이 주어진다. ( $1 \leq X_i \leq 1000$ )

- ▶ DP를 이용한 풀이
- ▶ 출력: 가장 긴 LIS(최장 증가 부분 수열)의 길이
- ▶  $LIS(x)$  = 배열의  $[1, x]$  에서 나오는 LIS의 길이
- ▶ 여전히 관계식을 찾기 힘들다.
- ▶  $LIS(x)$  = 맨 마지막 원소가 배열의  $x$ 번째 원소인 LIS의 길이

- ▶  $LIS(x)$  = 맨 마지막 원소가 배열의  $x$ 번째 원소인 LIS의 길이
- ▶  $LIS(i) = 1 + \max( LIS(j) ) \ (j < i \text{ and } arr[j] < arr[i])$
- ▶  $LIS(*)$ 의 최대값 출력

1	6	2	8	5	7	4	3
---	---	---	---	---	---	---	---

1	2	2	3	3	4	3	3
---	---	---	---	---	---	---	---

```
st = new StringTokenizer(br.readLine());
for(int i=0; i<N; i++){
    X[i] = Integer.parseInt(st.nextToken());

    for(int j=0; j<i; j++){
        if(X[j] < X[i]){
            LIS[i] = Math.max(LIS[i], LIS[j]);
        }
    }
    LIS[i]++;

    maxi = Math.max(maxi, LIS[i]);
}

System.out.print(maxi);
```

## $O(N \log N)$ LIS

$K[l] := (i-1)$  원소까지 고려했을 때, 길이가  $l$ 인 LIS 마지막 원소 중 최솟값  
 $LIS[i] = \max(l) + 1$  ( $K[l] < arr[i]$ )

$K$  배열은 항상 오름차순이 됨  $>$  이분 탐색으로  $\max(l)$ 을 찾을 수 있음  
 $K$  배열 업데이트 :  $K[LIS[i]] = arr[i]$

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS								

0 

0
---

  
K



## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1							

1	4
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1						

1	2
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1					

1	1
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1	2				

2	3
1	1
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1	2	3			

3	5
2	3
1	1
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1	2	3	4		

4	8
3	5
2	3
1	1
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1	2	3	4	4	

4	6
3	5
2	3
1	1
0	0
K	

## $O(N \log N)$ LIS

arr	4	2	1	3	5	8	6	7
LIS	1	1	1	2	3	4	4	5

5	7
4	6
3	5
2	3
1	1
0	0
K	



## 핵심 코드

```
int[] LIS = new int[N];
int maxi=0;
ArrayList<Integer> K = new ArrayList<>();
K.add(0);

st = new StringTokenizer(br.readLine());
for(int i=0; i<N; i++){
    int X = Integer.parseInt(st.nextToken());

    LIS[i] = lower_bound(K, X);

    if(LIS[i] == K.size()) K.add(X);
    K.set(LIS[i], X);

    maxi = Math.max(maxi, LIS[i]);
}

System.out.print(maxi);
```

### ● 욕심 많은 윌리 (4100)

#### | 문제

헬로바자회에서  $N$ 개의 물건을 나눠주는데, 각 물건은 무게  $w_i$ 와 가치  $v_i$ 가 정해져있다.

욕심이 많은 윌리는 물건을 다 가져가고 싶지만, 가방이 낡아  $W$ 보다 무거운 무게를 담으면 가방이 끊어지고 만다.

윌리가 가방이 끊어지지 않는 선에서 가져갈 수 있는 물건 가치의 합의 최대를 구하자.

#### | 입력

첫째 줄에 물건의 개수  $N$  ( $1 \leq N \leq 100$ )과 가방의 한계  $W$  ( $1 \leq W \leq 10^5$ )가 주어진다.

둘째 줄부터  $N$ 개의 줄에 물건의 무게  $x_i$  ( $1 \leq x_i \leq W$ )와 물건의 가치  $v_i$  ( $1 \leq v_i \leq 10^9$ )이 주어진다.

- ▶ DP를 이용한 풀이
- ▶ 출력: 무게  $W$  제한 이내에서 고를 수 있는 물건의 최대 가치 합
- ▶  $dp(x)$  = 무게  $x$ 로 고를 수 있는 물건의 최대 가치 합
- ▶ 중복해서 고르는 경우 제외 필요 (고르는 순서 상관 X)
- ▶  $dp(x, i)$  = (1~ $i$ )번 물건만 사용하여 무게  $x$ 로 고를 수 있는 물건의 최대 가치 합

- ▶  $dp(x, i) = (1 \sim i)$ 번 물건만 사용하여 무게  $x$ 로 고를 수 있는 물건의 최대 가치합
- ▶  $dp(x, i) = \max(dp(x - w[i], i - 1) + v[i], dp(x, i - 1))$
- ▶  $dp(*, N)$ 의 최대값 출력

	0	1	2	3	4	5	6	7	8	9
(3,6)	0	0	0	6	0	0	0	0	0	0
(4,3)	0	0	0	6	3	0	0	9	0	0
(5,5)	0	0	0	6	3	5	0	9	11	8
(4,7)	0	0	0	6	7	5	0	13	11	12

```
st = new StringTokenizer(br.readLine());
int N = Integer.parseInt(st.nextToken());
int W = Integer.parseInt(st.nextToken());

long[][] dp = new long[N+1][W+1];

for(int i=1; i<=N; i++){
    st = new StringTokenizer(br.readLine());
    int x = Integer.parseInt(st.nextToken());
    int v = Integer.parseInt(st.nextToken());

    for(int j=1; j<=W; j++){
        dp[i][j] = dp[i-1][j];
        if(j-x >= 0) dp[i][j] = Math.max(dp[i][j], dp[i-1][j-x]+v);
    }
}

System.out.print(dp[N][W]);
```

### ● 최장 길이 공통 부분 문자열(4118)

#### 문제

문자열  $S, T$  가 주어졌을 때, 두 문자열의 최장 길이 공통 부분 문자열(LCS)의 길이를 구하자.

#### 입력

$S, T$ 는 알파벳 소문자로 구성되어 있다.  
두 문자열의 길이는 1이상 3000이하이다.

#### 출력

최장 길이 공통 부분 문자열 (LCS)의 길이를 출력한다.

# LCS란?

## LCS란?

- Longest Common Substring(최장 공통 문자열)
- Longest Common Subsequence(최장 공통 부분 수열)

### Longest Common Subsequence

ABCDEF  
GBCDFE → BCDF

ABCDEF  
GBCDFE → BCDE

### Longest Common Substring

ABCDEF  
GBCDFE → BCD

$LCS[i][j]$  := 문자열 S의 i번째, 문자열 T의 j번째로 끝나는 공통 문자열의 최대 길이

- $S[i] == T[j]: LCS[i][j] = LCS[i-1][j-1]$
- $S[i] != T[j]: LCS[i][j] = 0$



$LCS[i][j]$  := 문자열 S의 i번째, 문자열 T의 j번째 문자까지 고려했을 때, LCS 값

- $S[i] == T[j]$ :  $LCS[i][j] = LCS[i-1][j-1]$
- $S[i] != T[j]$ :  $LCS[i][j] = \max(LCS[i-1][j], LCS[i][j-1])$

```
String S = br.readLine();
String T = br.readLine();

int[][] LCS = new int[S.length()+1][T.length()+1];
for(int i=1; i<=S.length(); i++){
    for(int j=1; j<=T.length(); j++){
        if(S.charAt(i-1) == T.charAt(j-1)) LCS[i][j] = LCS[i-1][j-1] + 1;
        else LCS[i][j] = Math.max(LCS[i-1][j], LCS[i][j-1]);
    }
}

System.out.print(LCS[S.length()][T.length()]);
```