# 서울시립대학교 소모임 알

컴퓨터과학부 김희중

## Dynamic Programming 2

# 가장 긴 바이토닉 부분 수열

# 가장 긴 바이토닉 부분 수열



https://www.acmicpc.net/problem/11054

- LIS 문제와 흡사한 문제
- LIS와 LDS(Longest Decreasing Subsequence)를 구한다.
- LIS[i] = i번째 숫자를 끝으로 하는 LIS의 길이
- LDS[i] = i번째 숫자를 시작으로 하는 LDS의 길이
- B[i] = i번째 숫자에서 감소가 시작되는 바이토닉 부분 수열

# 가장 긴 바이토닉 부분 수열



https://www.acmicpc.net/problem/11054

- Bi[i] = LIS[i] + LDS[i] - 1;

| SEQ | 1 | 5 | 2 | 1 | 4 | 3 | 4 | 5 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| LIS | 1 | 2 | 2 | 1 | 3 | 3 | 4 | 5 | 2 | 1 |
| LDS | 1 | 5 | 2 | 1 | 4 | 3 | 3 | 3 | 2 | 1 |
| BI | 1 | 6 | 3 | 1 | 6 | 5 | 6 | 7 | 3 | 1 |

- BI[i] 배열에서 가장 큰 값을 찾으면 된다.

# 가장 긴 바이토닉 부분 수열



https://www.acmicpc.net/problem/11054

- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/11054.cc

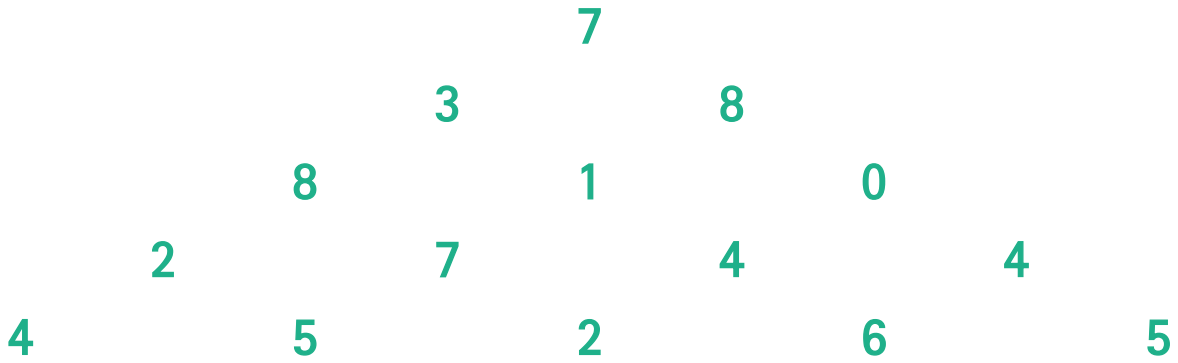
# 숫자삼각형

# 숫자삼각형

https://www.acmicpc.net/problem/1932



- d[i][j] = i행 j열이 도착점일 때 합의 최대값

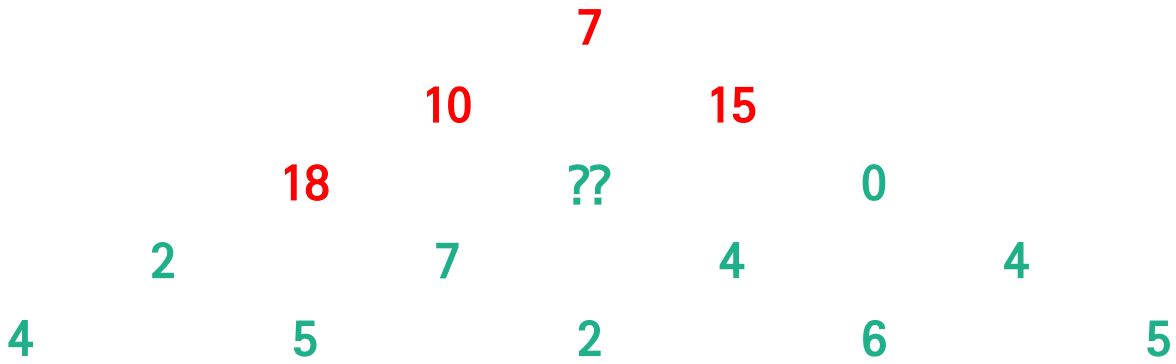


# 숫자삼각형

https://www.acmicpc.net/problem/1932



- d[i][j] = max(d[i-1][j-1], d[i-1][j]) + s[i][j];



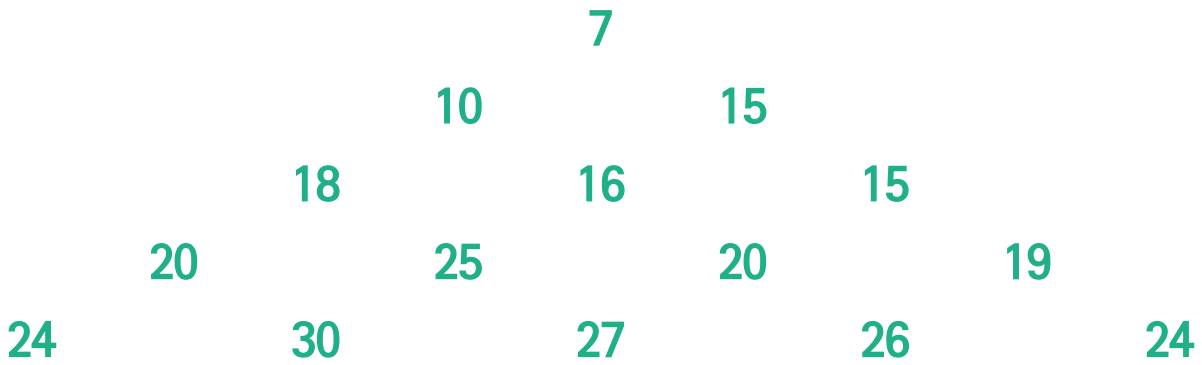

- d[i-1][j-1](10)과 d[i-1][j](15)중 큰 값과 자신을 더한 값으로

# 숫자삼각형

https://www.acmicpc.net/problem/1932



- d[i][j] = max(d[i-1][j-1], d[i-1][j]) + s[i][j];




- 맨 아랫줄에서 가장 큰 값인 30이 이번 문제의 답이 된다.

# 숫자삼각형

https://www.acmicpc.net/problem/1932



- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/1932_bottomup.cc
- https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/1932_topdown.cc

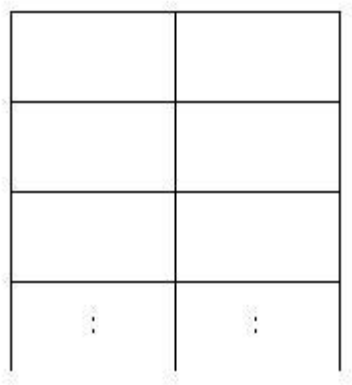
# 동물원

# 동물원

https://www.acmicpc.net/problem/1309



- N번째 줄이 비어있는 경우, 그렇지 않은 경우로 나누어서 생각한다.
- d[i] = 세로 i칸에 동물을 배치하는 경우
- d[i] = i번째 줄이 비어있는 경우 + 비어있지 않은 경우



# 동물원

https://www.acmicpc.net/problem/1309



- d1[i] = i번째 줄이 비어있을 때의 경우의 수
- d2[i] = i번째 줄이 비어있지 않을 때의 경우의 수
- 세로로 i칸인 동물원을 채우는 경우의 수는 d1[i] + d2[i]가 된다.

# 동물원

https://www.acmicpc.net/problem/1309



- d1[i](i번째 줄이 비어있는 경우의 수)
- 는 i번째 줄에 아무 동물도 배치하지 않기 때문에 i-1번째 칸이 어떤 상태인지 고려할 필요 없이 i-1칸을 채우는 경우의 수와 같다.

- 따라서 d1[i] = d1[i-1] + d2[i-1]; 이 된다.

# 동물원

https://www.acmicpc.net/problem/1309



- d2[i](i번째 줄이 비어있지 않은 경우의 수)
- 는 i-1번째 줄의 상태에 따라 경우가 달라진다.

- i-1번째 줄이 비어있을 경우에는 i번째 동물을 배치할 수 있는 경우가 2가지이다. ( x2)

- i-1번째 줄이 비어있지 않을 때는 i-1번째 동물의 위치에 따라 i번째 줄의 동물의 위치가 1가지로 결정된다.

- 따라서 d2[i] = d1[i-1] * 2 + d2[i-1]; 이 된다.

# 동물원

https://www.acmicpc.net/problem/1309



- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/1309.cc

# 공통 부분 문자열

# 공통 부분 문자열



https://www.acmicpc.net/problem/5582

- Longest Common Substring 이라 불리는 문제이다.
- 문자열 s1, s2가 있다.
- d[i][j] = s1[i], s2[j]로 끝나는 최장 공통 부분 문자열의 길이
- d[i][j] = d[i-1][j-1] + 1 (s1[i] == s2[j]일 때)
  0 (나머지)
- d배열에 있는 값 중 가장 큰 값을 출력한다.

# 공통 부분 문자열



https://www.acmicpc.net/problem/5582

- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/5582.cc

# LCS

# LCS



https://www.acmicpc.net/problem/9251

- Longest Common Subsequence문제로 불린다. 최장 공통 부분 문자열(Longest Common Substring)과 혼동해서는 안 된다. 둘은 비슷하지만 확실히 다른 문제이다.

- 단어들의 앞 글자를 따서 LCS라고 부른다.

- 여러가지로 중요한 문제. 생물학에도 응용이 되는 알고리즘이며 Linux System의 diff유틸리티의 근간이 된다.

# LCS



https://www.acmicpc.net/problem/9251

ABRACADABRA
ECADADABRBCRDARA

- 최장 공통 부분 문자열, 길이 : 5

ACAYKP
CAPCAK

- 최장 공통 부분 수열(LCS), 길이 : 5

# LCS



https://www.acmicpc.net/problem/9251

- 문자열 s1, s2가 있다.
- d[i][j] = s1[i], s2[j]로 끝나는 LCS의 길이
- d[i][j] = d[i-1][j-1] + 1 (s1[i] == s2[j]일 때)
  max(d[i-1][j], d[i][j-1]) (나머지)
- D배열의 가장 끝칸을 출력한다.

# LCS



https://www.acmicpc.net/problem/9251

- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/9251_bottomup.cc
- https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/9251_topdown.cc

# 동전 2

# 동전 2



https://www.acmicpc.net/problem/2294

- 이 문제는 쉬운 점화식과 어려운 점화식의 두가지 풀이가 있다.
- D[i][j] = i번째 동전까지 사용했을 때 j원을 만들 수 있는 최소 동전의 개수
- D[i] = i원을 만들 수 있는 최소 동전의 개수
- 어떤 풀이가 더 어려울까?

# 동전 2



https://www.acmicpc.net/problem/2294

- 동전 n개의 종류를 담고 있는 배열 s가 있다.
- D[i][j] = i번째 동전까지 사용했을 때 j원을 만들 수 있는 최소 동전의 개수
- Top-down 방식을 이용해 해결한다.
- 경우마다 i번째 동전을 선택 하거나, 하지 않거나 2가지의 선택을 할 수 있다.

# 동전 2

https://www.acmicpc.net/problem/2294



- 동전 n개의 종류를 담고 있는 배열 s가 있다.
- d[i] = i원을 만들 수 있는 최소 동전의 개수
- Bottom-up 방식을 이용한다.
- 동전의 조합이 없는 경우에 -1을 출력 해야하기 때문에 D배열을 모두 -1로 초기화 한다.
- d[i]에서 동전 중 i원이 있다면 d[i]는 1로 초기화 한다.

# 동전 2



https://www.acmicpc.net/problem/2294

- 1, 5, 12의 동전이 있다면 i + 1, i + 5, i + 12배열을 방문하면서 방문했다면 **최소값**, 방문하지 않았다면 d[i] + 1을 넣는다.

- D[i+s[j]] = min(d[i+s[j]], d[i] + 1) (d[i+s[j] == -1)
- = d[i] + 1 (d[i+s[j]] != -1)

# 동전 2

https://www.acmicpc.net/problem/2294



- C++ Code : https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/2294_bottomup.cc
- https://github.com/OfficialDominyellow/AlgorithmByDominyellow/blob/master/BackjoonOnlineJudge/2294_topdown.cc