HW2 2016047883 정성훈

[01] RodCutting.cpp

@실행화면

@컴파일환경

GCC version 8.1.0

@알고리즘설명

(소스코드에 주석을 달아 설명을 보충했습니다)

막대의 길이가 N이라고 하였을 때, 막대길이 1인 경우부터 N인 경우까지 최대의 이익 값을 최적화해가며 문제를 풀었습니다.

이 로직이 구현된 함수의 원형은 아래와 같습니다.

```
int rodCutting(const vector<int> &p, int n, vector<int> &r);
```

인자로 p(초기 막대 길이당 가격 값), n(현재 막대의 길이), r(값 최적화한 것을 저장한 벡터)를 받고

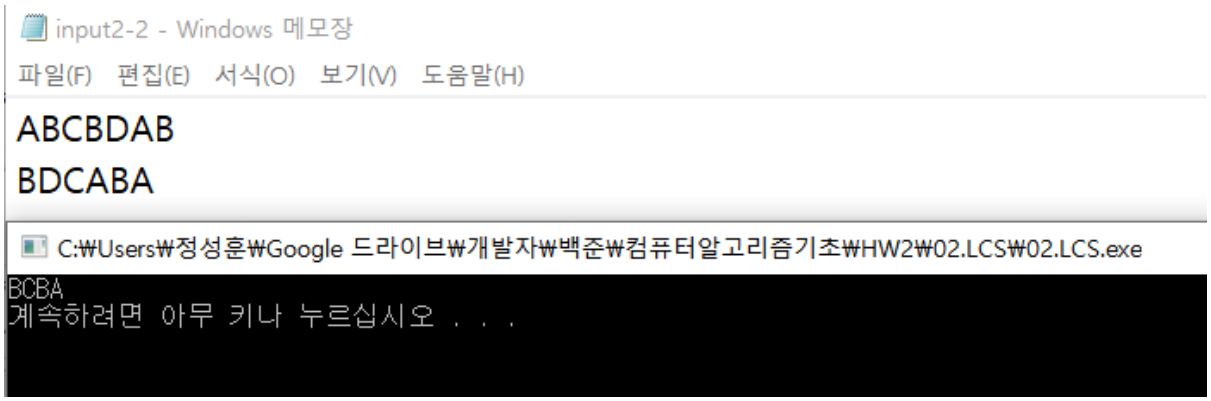
리턴값으로 길이가 n인 경우의 최대 이익 값을 리턴합니다.

최대 이익 값을 가지는 경우의 막대 길이(막대를 두 개로 자르니 x와 y라고 하겠습니다)는 위의 rodCutting 함수 내부에서, 최대 이익 값이라고 판단되는 경우 x와 y를 해당 자른 길이로 값을 반복적으로 갱신해주었습니다. 계속해서 값을 갱신해줘도 1 ~ N까지 rodCutting 함수로 최대 이익 값을 최적화 해주며 갱신하기 때문에 최종적으로 최대 이익 값을 내는 막대의 길이로 갱신되기 때문입니다.

출력 직전 x, y의 크기를 비교해서 오름차순으로 출력해주었습니다.

[02] LCS(Longest Common Subsequence)

@실행화면



@컴파일환경

GCC version 8.1.0

@알고리즘설명

(소스코드에 주석을 달아 설명을 보충했습니다)

(구현한 함수의 이름이 곧 역할이라 설명을 따로 적지 않았습니다)

두 문자열을 각각 2차원 배열의 행(더 긴 문자열 X), 열(더 짧은 문자열 Y) 인덱스에 매핑시켰습니다(정확하게는 문자열 인덱스 시작값을 1로 매핑함). 이후 (문자열 X의 길이 +1 * 문자열 Y의 길이 +1) 크기의 배열 두 개를 만들어 각각 LCS의 길이 값, 그리고 방향값(0: left, 1: up, 2: diagonal)을 저장하는데 사용하였습니다.

배열을 1 ~ X길이, 1 ~ Y길이로 탐색해가며 배열의 행, 열 인덱스 값을 각각 X, Y 문자열의 인덱스 값으로(정확하게는 -1씩해서)하여 문자열이 같은지를 비교하고, 같으면 'LCS 길이 값 배열의 왼쪽 대각선 값 + 1'을 해당 인덱스에 저장합니다. 방향값은 2(대각선)로 저장합니다.

문자열이 다르면 해당 인덱스의 '위쪽 LCS 길이 값(행 인덱스 -1)'과 왼쪽 LCS길이 값(열 인덱스 - 1) 중 큰 값을 선택해서 해당 인덱스에 저장합니다. 방향값은 왼쪽은 0, 위쪽은 1로 저장합니다. 이때 왼쪽과 위쪽의 LCS값이 같은 경우 과제 명세서에 나와있는 대로, 위쪽을 선택합니다.

이후 배열의 탐색이 다 끝나면, 방향값 배열의 오른쪽 맨 밑 끝부터 방향값을 따라 back tracing 합니다. 이때 방향값이 대각선인 경우에만 해당 문자를 정답 문자열에 추가합니다.

정답을 출력할 때는 문자열을 역추적해서 저장하였기 때문에 저장값을 거꾸로 뒤쪽부터 출력하였습니다.