Dynamic Programming 기법을 활용한 문제 풀이

2016\*\*\*\*

* 코드 작성 및 컴파일 환경 : Visual Studio 2015 Community

문제 1. 입력된 두 개의 문자열을 비교하여 점수를 매기는 방식으로 두 문자열의 유사도를 측정하고자 한다. 두 문자열이 일치하면 s점을 추가하고, 틀리면 f점을 감점한다. 이때 비교 대상이 되는 문자열의 위치에 gap을 추가하여 비교를 생략할 수 있는데, 이런 경우에는 p점을 감점한다. 위와 같은 방법으로 문자열을 비교해서 나올 수 있는 최대 점수와 각 문자열에 삽입될 gap의 개수, 위치를 다이나믹 프로그래밍 기법을 활용하여 구한다.

[문제 해결을 위한 아이디어]

1. 유사도 점수 최댓값 구하기

두 문자열 x, y를 비교하여 얻은 점수를 2차 행렬 S에 저장한다. S의 좌표를 S(I,j)라고 하면, S(I,j)에는 x의 i번째 문자와 y의 j번째 문자까지 비교한 점수를 기록한다. 즉, S(1,1)에는 x의 첫 번째 문자와 y의 첫 번째 문자를 비교해서 해당 문자가 일치하면 2를 저장하고, 그렇지 않으면 -1을 저장한다. 그리고 임의의 S(I,j)의 값은 x\_i와 y\_j 비교 결과가 일치하면 행렬에서의 외쪽 대각선 위인 S(i-1,j-1)의 값에 s점을 더한 값을 저장하고, 비교 결과가 일치하지 않으면 S(i-1,j-1)에 f점을 뺀 값을 저장한다. 이때, 만약 행렬 상에서 S(i,j)의 왼쪽인 S(i,j-1)에서 p점을 뺀 값이나 위인 S(i-1,j)에서 p를 뺀 값이 S(i,j)의 값보다 크면 해당되는 큰 값을 S(i,j)에 저장한다. 이는 S(i,j)가 x\_i와 y\_j까지의 유사도 점수를 기록하고 있기 때문이다. 가령 x\_1과 y\_1이 일치해서 2점을 획득했다고 한다면, x\_2와 y\_2의 비교 결과에 x\_1과 y\_1의 결과인 2를 더해야 최종적으로 x\_2ㅘ y\_2까지의 비교 점수를 얻을 수 있는 것이다.

위와 같은 방법으로 행렬을 작성하면, 가장 오른쪽 아래에 저장되는 값은 문자열 x와 y의 최종적인 유사도 점수가 된다.

1. Gap 개수와 위치 구하기

2차 행렬 record에 x와 y에 삽입되는 gap의 위치를 기록한다. 임의의 정수 i, j에 대하여 record(i,j)는 S(i,j)의 값에 따라 0, 1, 2 중 하나가 된다. S(i,j)의 값이 S(i-1,j-1)+s 또는 S(i-1,j-1)-f 와 같으면 0을 저장하고, S(i-1,j)-p와 같으면 1을 저장한다. 그리고 S(i,j-1)-p 와 같으면 2를 저장한다. 이는 x\_i와 y\_i 문자를 비교했을 때 gap을 추가하지 않는 경우 0을 표시하고, gap을 추가하는 경우 1 혹은 2로 표시하는 것을 의미한다. 1)의 방법에 따라 S(i,j)의 값이 왼쪽 혹은 위의 값에서 p를 뺀 것과 같다는 의미는 x\_i와 y\_j를 비교하는 대신 gap을 추가하여 얻은 점수가 더 높다는 의미이기 때문이다. 즉, record(i,j)에 1이 입력된 경우 x\_(i-1)과 y\_j의 값에 p점을 뺐다는 의미이므로, 문자열 y의 i-1번째에 gap이 추가되었다는 의미이다. 반대로 만약 2가 저장되어 있다면, 문자열 x의 j-1번째에 gap이 삽입되었다는 의미이다.

위의 방법으으로 gap의 개수와 위치를 알기 위해 작성된 행렬 record의 가장 오른쪽 아래부터 데이터를 읽는다. 저장된 값이 0이면 왼쪽 위 대각선의 값을 읽고, 1이면 위의 값을, 2면 왼쪽의 값을 읽는다. 1의 개수가 y에 삽입된 gap의 개수이며, 2의 개수가 x에 삽입된 gap의 개수이다.

[문제 해결의 예]

위의 방법으로 문제를 해결한다.

s = 2, f = 1, p = 2

x = ATCGGATCT

y = ACGGACT

라고 가정하면, 문자열 x의 길이는 9, 문자열 y의 길이는 7이다. 문제 해결 아이디어의 1)에 따라 행렬 S를 작성하면 다음과 같다.

0 -2 -4 -6 -8 -10 -12 -14

-2 2 0 -1 -1 2 0 -1

-4 0 1 -1 -2 0 1 2

-6 -1 2 0 -2 -2 2 0

-8 -1 0 4 2 0 0 1

-10 -1 -2 2 6 4 2 0

-12 2 0 0 4 8 6 4

-14 0 1 -1 2 6 7 8

-16 -1 2 0 0 4 8 6

-18 -1 0 1 -1 2 6 10

가장 오른쪽 아래의 값이 10으로, 두 문자열 x, y의 유사도 최대 점수가 10임으로 알 수 있다.

문제 해결 아이디어의 2)의 방법으로 행렬 record를 나타내면 다음과 같다.

0 2 0 0 0 2 0

1 0 2 0 1 0 2

0 1 2 2 1 1 0

0 1 0 2 2 1 0

0 1 1 0 2 2 2

0 2 1 1 0 2 2

1 0 2 1 1 0 2

0 1 2 1 1 1 0

0 1 0 2 1 1 0

오른쪽 아래부터 데이터를 읽으며 위로 올라가면 1의 개수가 2개, 2의 개수가 0개이므로, x에 삽입된 gap은 0, y에 삽입된 gap은 2개임을 알 수 있다. 해당 2의 좌표는 (8,6), (3,2)로, y에 삽입된 gap의 위치는 8-1=7, 3-1=2 이다. 실제로 문자열 y의 2번째와 7번째에 gap(\_)을 삽입하고 점수를 계산하면

x = ATCGGATCT

y = A\_CGGA\_CT

(2\*7 = 14) – (2\*2=4) = 10

으로 실제 결과와 일치함을 알 수 있다.

[코드 작성]

FILE\_NAME\_INPUT 메크로에 정의된 input.txt를 읽어들이고 문자열의 길이와 문자열이 저장되어 있는 파일명과 s,f,p 값을 저장한다. Txt 파일에서 읽어들인 bin 파일을 로드하고 두 문자열의 길이를 m, n에 저장한다. M개 만큼의 문자열을 읽어 x에 저장하고, n개 만큼의 문자를 읽어 y에 저장한다. 그리고 문제 해결 아이디어의 방법으로 행렬 S와 record를 저장한다. 저장이 끝나면 output.txt라는 파일에 그 결과를 적는다. 결과는

최대점수

Gap이 삽입된 이후 문자열의 길이

X에 삽입된 gap의 개수

X에 삽입된 gap의 위치

Y에 삽입된 gap의 개수

Y에 삽입된 gap의 위치

순서로 저장한다.

문제 2. 올바른 카드 섞기의 결과 판단

[문제 해결 아이디어]

전형적인 Longest Common Sequence 해결 기법을 활용한다. 세 번째 줄에 있는 데이터를 수열이라고 생각하고, 첫 번째 줄에 있는 데이터가 세 번째 수열 안에 있는지 없는지를 체크한다. 같은 방법으로 두 번째 줄에 입력된 데이터 역시 세 번째 줄에 있는지 없는지를 확인한다.

첫 번째 줄에 있는 카드의 목록을 a라고 하고, 두 번째 줄에 있는 카드 목록을 b라고 가정한다. A와 b를 섞은 후 얻어진 세 번째 줄의 카드 목록과 네 번째 줄의 카드 목록은 각각 c와 d라고 가정한다.

1. c에 a가 있는지 확인하기 위해 a와 c를 비교한다. 이때 이차 행렬을 작성하여 그 결과를 확인한다. 임의의 정수 i,j에 대하여, 행렬 S(i,j)에 저장되는 값은 a\_i와 c\_j까지를 비교해서 c에 포함된 a의 개수이다. 이를 얻기 위해 a\_i와 c\_j의 값을 비교하고 그 결과가 같으면 왼쪽 대각선 위인 S(i-1,j-1)의 값에 1을 더한다. S(i-1,j-1)은 c\_(i-1)까지 데이터 중 a\_(i-1)이 몇 개 포함되어 이는지를 저장한 값이기 때문에 한 개씩 더 읽었을 때의 값이 같으면 이전의 결과에 1을 추가해 a\_i, c\_j의 값이 같다는 것을 표시하는 것이다. 만약 두 값이 같지 않으면 왼쪽이나 위인 S(i,j-1), S(i-1,j) 중 큰 값을 S(i,j)에 저장한다. 현재 비교하는 데이터가 같지 않기 때문에 이전까지 세고 있던 포함된 데이터의 개수를 변동 없이 사용하는 것이다.

위의 방법으로 행렬을 작성한 후, 작성된 행렬의 오른쪽 마지막 데이터가 수열 a의 개수와 같다면 수열 c에 수열 a가 포함되어있음을 뜻한다.

1. 1번의 방법으로 b와 c를 비교한다.
2. 1번과 2번의 결과가 모두 참이면 카드가 정상적인 방법으로 섞였다는 것을 의미한다.

[코드 작성]

1. INPUT\_FILE\_NAME 에 정의된 input.txt 파일을 열고 한 줄씩 읽어 a, b, c, d를 int 타입의 배열로 저장한다. 그리고 각 배열의 크기 역시 lengtha, lengthb, lengthc, lengthd에 저장한다.
2. 문제 해결 아이디어 1)을 수행하는 함수 check\_mixed\_card를 정의한다. 함수는 배열 b에 배열 a가 속하는지를 확인한다. 이때 m은 a의 길이, n은 b의 길이이다. 함수는 행렬 S의 마지막 값을 반환한다.
3. 작성한 check\_mixed\_card 함수를 이용하여 입력 데이터 input.txt로부터 읽어들인 배열 c에 a가 있는지 확인하고, c에 b가 있는지 확인한다. 두 번의 함수 호출의 결과가 각각 lengtha와 lengthb와 같다면 배열 a와 b가 배열 c에 순서대로 있다는 의미이므로 카드가 올바르게 섞였다는 것을 알 수 있다. 따라서 output.txt에 1을 저장한다. 만약 두 결과 중 하나라도 각각 lengtha와 lengthb와 다르다면 카드가 올바르게 섞이지 않았다는 의미이므로 output.txt에 0을 저장한다.
4. 3번과 같은 방법으로 배열 d에 배열 a와 b가 있는지를 확인하고 그 결과를 0 또는 1로 output.txt에 저장한다.
5. 예외 처리로, 입력된 데이터 배열 a, b 중에 1보다 작거나 1000보다 큰 수가 있으면 이를 알리고 프로그램을 종료한다.