

근사 문자열 매칭

해당 거리 : 그 자리를 변경

편집 거리 : 문자열 끼워넣기 가능

삽입

삭제

교체

편집 연산의 최소 수.

모든 문자열에 대해서

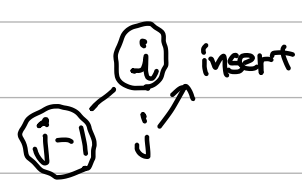
- | | |
|---------------------------|--------|
| ① 안 변경. | match |
| ② 다른 문자로 변경. (교체) | change |
| ③ 사이즈 줄어 인간으로 변경 (삭제) | delete |
| ④ 사이즈 늘어난 인간이 문자로 변경 (삽입) | insert |

마지막 글자가 다르면 match X
같은면 change X

편집 연산의 최소 수

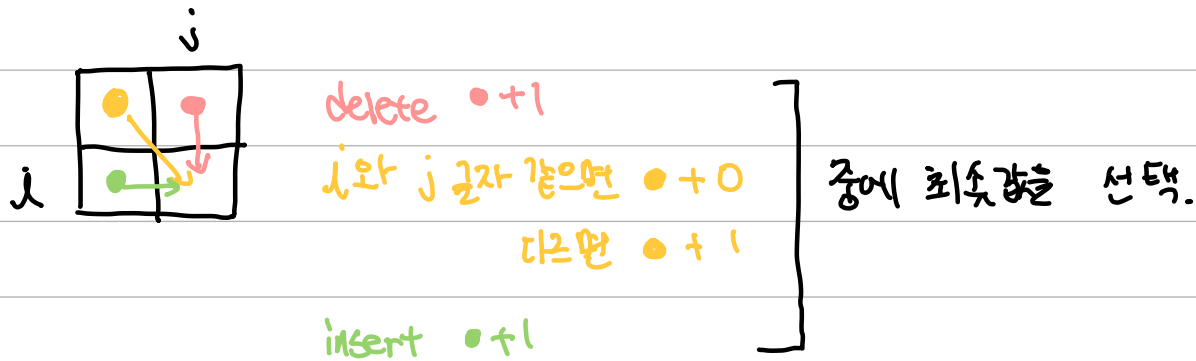
S_1 문자열의 인덱스 i 까지를 편집 연산으로 S_2 문자열의 j 까지 똑같이 만드는데
필요한 편집 연산의 수 $D(i, j)$ 즉, S_1 을 S_2 로 바꾸는 것.

$$D(i, j) = \min \begin{cases} D(i, j-1) + 1 \rightarrow \text{insert} \\ D(i-1, j) + 1 \rightarrow \text{delete} \\ D(i-1, j-1) + t(i, j) \end{cases}$$



i 의 문자와 j 의 문자가 같으면 0 match
다르면 1 change

Tabular Computation



Traceback

Tabular Computation 에서 어느 방향으로서 왔는지 저장하면, 따라가기 가능.

Sequence Alignment

위 문제와 같고, 빈자리 대신 $-$ 로 자리를 차지 시킨다. 편집거리 최소 대신 점수 최대화.

optimal global alignment V 는 점수가 최대인 답.

$$V(0,0) = 0$$

$$V(i,0) = V(i-1,0) + \sigma(a_i, -)$$

$$V(0,j) = V(0,j-1) + \sigma(-, b_j)$$

σ 는 점수 함수.

$$\therefore V(i,j) = \max \begin{cases} V(i,j-1) + \sigma(-, b_j) \rightarrow \text{insert} \\ V(i-1,j) + \sigma(a_i, -) \rightarrow \text{delete} \\ V(i-1,j-1) + \sigma(a_i, b_j) \rightarrow \text{match or change} \end{cases}$$

최장 공통 부분 서열 LCS

두 문자열의 유사도 측정.

연속이 아니어도 허용.

LCS의 길이는?

모든 i, j 에 대해서 $LCS(i, j)$ 를 구한다.

match는 갖것만 세면 됨!

즉, 마지막 문자가 같으면 $X_{i-1}, Y_{j-1} = 1$ $LCS \quad LCS(i-1, j-1) + 1$
다르면 $\max \begin{cases} LCS(i-1, j) \\ LCS(i, j-1) \end{cases}$