Algorithm I
Magic Potion
Algorithm 해결

주어진 두 문자열 X 와 Y 의 길이 k 짜리 subsequence 집합이 동일하게 되는 최대의 k 값을 구하는 문제이다. 이 때, 한 문자열의 모든 subsequence에 대해 다른 문자열의 subsequence인지 테스트해보는 것은 k 에 대해 지수적인 시간이 들기 때문에 문자열을 따로 처리하여 판별이 용이하도록 해야 한다. k 의 범위가 0 이상 m 과 n 이하여야 하므로, 각각의 가능한 k 에 대해 이진 탐색을 수행해, X 와 Y 의 길이 k subsequence 집합이 동일한지 반복적으로 판단한다. 알고리즘의 총 실행 시간은 $O(m+n)$ 연산을 $O(\log\min(m,n))$번 반복하므로 $O\big((m+n)\log\min(m,n)\big)$이다. 이 때, 하나의 k 값에 대하여 $O(m+n)$ 시간 내에 subsequence set의 동일 유무를 판단하는 방법은 다음과 같다.

문자열 X 가 주어졌고, X 와 길이 k 짜리 subsequence 집합이 동일한 모든 문자열의 집합을 $L_k(X)$라고 하자. $L_k(X)$의 모든 문자열 중 길이가 가장 짧은 문자열, 그리고 그 중에서 사전순으로 첫번째인 문자열은 유일하기 때문에 $L_k(X)$을 대표하는 문자열이 된다. 따라서 X 로부터 $L_k(X)$를 대표하는 문자열, 그리고 Y 로부터 $L_k(Y)$를 대표하는 문자열을 계산하여 두 문자열이 동일한지 판단하면 된다.

$L_k(X)$ 를 대표하는 문자열을 구하기 위해서는, 우선 제거할 수 있는 모든 문자를 제거하여 $L_k(X)$ 내의 길이가 가장 짧은 문자열을 구한다. X 의 문자 $X[i]$ $(i = 0,1,\dots,|X|-1)$를 제거할 수 있는 경우는 왼쪽과 오른쪽으로 각각 그 문자를 사용해야만 얻을 수 있는 subsequence의 최소 길이의 합이 $k+1$ 을 초과하는 경우이다. 이를 각각 Xvec$[i]$, Yvec$[i]$ 라고 하자. 이 때, X 를 왼쪽에서부터 오른쪽으로 훑으며 각 문자의 왼쪽으로 그 문자를 사용해야만 얻을 수 있는 subsequence의 최소 길이를 저장해 둔 뒤, 다시 오른쪽에서 왼쪽으로 돌아오며 제거할 수 있는 모든 문자를 제거하는 방식으로 한 번에 처리가 가능하다.

```
vec ← [1,1, ...,1]
Xvec ← length |X| array}
for (i = 0; i < |X|; i++)
        Xvec[i] ← vec[X[i]]
vec[X[i]] ++
for (all characters c)
        vec[c] = min(vec[c],vec[X[i]])
```

위 수도코드는 X 의 각 문자에 대해 왼쪽으로 그 문자를 사용해야만 얻을 수 있는 subsequence의 최소 길이를 배열 Xvec에 저장한다. 같은 방법, 반대 방향으로 배열 Yvec을 계산할 수 있으며, 그 합이 $k+1$ 을 넘는 문자는 없는 문자 취급하여 지워야 할 모든 문자를 $O(|X|)$시간 내에 지울 수 있다.

이후 계산한 최소 길이의 문자열에서 연속한 문자 $X[i_1], X[i_2], \ldots, X[i_j]$가 다음 세 조건을 만족할 때, 이 문자들을 사전식으로 재배열하여 사전 순서로 가장 첫번째 문자를 계산할 수 있다.

- $\mathrm{Xvec}[i_1] = \mathrm{Xvec}[i_2] = \cdots = \mathrm{Xvec}[i_j]$
- $\mathrm{Yvec}[i_1] = \mathrm{Yvec}[i_2] = \cdots = \mathrm{Yvec}[i_j]$
- $\mathrm{Xvec}[i_1] + \mathrm{Yvec}[i_1] = k + 1$

위 과정을 X 와 Y 에 대해 반복하고 구한 두 문자열이 같다면, X 와 Y 는 길이 k 의 subsequence 집합이 같다. 그렇지 않다면 두 집합이 같지 않다. 이 과정을 $\log(\min(m,n))$번 반복하여 최적의 k 를 구하면 된다.