Algorithm Lab

Week 7: Sequence Alignment

We have 2 sequences $A = (a_1, a_2, \dots, a_m)$ and $B = (b_1, b_2, \dots, b_n)$. We can insert void symbols into either sequence at arbitrary place. An alignment is to make 2 sequences with the same length by do previous operation on A and B; Assume the length of made sequences is k, we can denote the sequence from A by $A' = (a'_1, a'_2, \dots, a'_k)$ and denote the sequence from B by $B' = (b'_1, b'_2, \dots, b'_k)$. When we align A' and B', for each pair symbol have 3 different states: **match** if $a'_i = b'_i$; **gap** if each of them is a void symbol; **mismatch** if $a'_i \neq b'_i$. Every state will correspond to a score (or penalty), named s_{match}, s_{gap}, and $s_{mismatch}$. Score of the alignment is the summation of score of each paired symbol. Sequence alignment problem ask to find an alignment can maximize the alignment score. Longest common subsequence problem is a specialized version of sequence alignment problem that $s_{match} = 1$, $s_{gap} = 0$, and $s_{mismatch} = 0$.

Instance: sequences A, B and score $s_{match}, s_{gap}, s_{mismatch}$

Result: sequences A', B' where $|A'| = |B'|$ and maximize the alignment score

---

Description

We denote the substring of A that contains i leading symbols by A_i and substring of B that contains j leading symbols by B_j. For convenient, we define 3 functions as followed:

- $f(i,j) \rightarrow A'_i, B'_i$: Find the alignment that can maximize alignment score of A_i and B_j.
- $g(i,j) \rightarrow Score$: The alignment score of $f(i,j)$.
- $p(i,j) \rightarrow Score$: Th alignment score of a_i and b_j. Should be s_{match} or $s_{mismatch}$.

Since every symbol of A and B will be the trailing symbol of some A_i or B_j, thus, we can consider only trailing symbols. For $f(i,j)$. Under the policy, the only operation we need to try is append(insert) void symbol to A_i, append void symbol to B_j, or leave the original paring. If b_j maps to a void symbol, means we'll append a void symbol to A_i; if a_i maps to a void symbol, we'll append a void symbol to B_j; If we append nothing to A_i and B_j, then a_i and b_j should be paired and be match or be mismatch.

If nothing to match, our score should be 0, thus, $f(0,0) = ((\,),(\,))$ and $g(0,0) = 0$.

$$g(i,j) = max\begin{cases} g(i-1,j) + s_{gap} \\ g(i,j-1) + s_{gap} \\ g(i-1,j-1) + p(i,j) \end{cases}$$

$$f(i,j) = \begin{cases} f(i-1,j)\ appends\ (a_i, void), & g(i,j)\ choose\ g(i-1,j) \\ f(i,j-1)\ appends\ (void, b_j), & g(i,j)\ choose\ g(i,j-1) \\ f(i-1,j-1)\ appends\ (a_i, b_j), & g(i,j)\ choose\ g(i-1,j-1) \end{cases}$$

When implement this algorithm, we usually cache values of $g(i,j)$ by a 2-dimension array.

Questions

1. In normally implementation, $f(m,n)$ can find one of optimal alignment. How to find out all of optimal alignments?
2. How many optimal alignments may exist? Please construct a set of input to explain your answer.
3. Suppose both A and B are very long, that we can't maintain all $m \times n$ scores in memory. Please find the way which only cache n values.
4. Analyze space complexity, time complexity in best case and worst case in Q1 and Q2.
5. Solve http://oj.csie.ndhu.edu.tw/problem/ALG04C.