Наибольшее общее субчисло

Решение

Если бы не начальные нули, так было совсем всё хорошо и просто – выбираем самое длинное общее субчисло, и задача решена. А, нет, не совсем решена – из общих субчисел этой самой наибольшей длины надо ещё выбрать наибольшее, но это уже как-нибудь сделаем. Вот только нули... Забудем про них ненадолго.

Начнём с длины самого длинного общего субчисла. Что-то такое уже было... Правильно, было. В материалах к 7-й задаче была статья

М.Рейтман. Динамическое программирование, “Квант”, 10, 1991,

а в ней разбиралась процедура построения наибольшей общей подпоследовательности двух строк (под подпоследовательностью данной строки понимается любая строка, полученная вычеркиванием из данной строки любого количества символов, возможно, что и 0 символов). Соответствующий фрагмент этой статьи приведен и в учебных материалах к данной задаче.

Вообще, задачу о поиске (длины) наибольшей общей подпоследовательности приходится решать очень часто. Эта задача появляется во многих задачах, связанных с сопоставлением и распознаванием образов, с классификацией, распознаванием. В теории кодирования наибольшая общая подпоследовательность естественно возникает при передаче исследовании каналов связи, в которых могут происходить ошибки типа вставок и выпадений символов. Соответственно, поиск в сети по ключевым словам “наибольшая общая подпоследовательность” или ”longest common subsequence” даёт массу содержательных результатов. Приведу только несколько интересных ссылок:

http://www.ics.uci.edu/~eppstein/161/960229.html

http://program.rin.ru/razdel/html/853.html

http://rain.ifmo.ru/cat/view.php/theory/algorithm-analysis/dynamic-programming-2004

Но вернёмся к нашей задаче о субчисле. В ней задаче слегка модифицируем алгоритм, предложенный в вышеупомянутой статье М.Рейтмана, – будем вычислять длину наибольшей общей подпоследовательности, двигаясь от конца к началу строк. Более точно:

пусть n1 и n2 – данные строки, а len1 и len2 – их длины, соответственно;

будем обозначать через s[k] символ, стоящий в строке s на месте номер k, а через s[a..b] строку, которая получается, если отбросить в строке s все символы от 1-го до (a-1)-го и от (b+1)-го до последнего;

пусть r[i,j] – это длина самой длинной общей подпоследовательности строк n1[i..len1] и n2[i..len2].

Заполняем массив r так, как это описано в статье, только двигаемся от конца слов к началу:

если n1[i] <> n2[j], то r[i,j] = max (r[i+1,j], r[i,j+1]);
если n1[i] = n2[j], то r[i,j] = max (r[i+1,j], r[i,j+1], r[i+1,j+1]+1).

А теперь, когда массив r уже заполнен мы легко можем ответить на вопрос: «А с какой цифры начинается наибольшее общее субчисло строк n1 и n2 ?». И сделаем мы это вот так.

Пусть p1[1] – это номер самой первой (самой левой) цифры 1 в строке n1, p2[1] – это номер самой первой (самой левой) цифры 1 в строке n2, а R[1]= r[p1[1], p2[1]]. Это означает, что имеется общее субчисло строк n1 и n2, начинающееся на 1 и имеющее длину R[1]. Аналогично, найдём величины R[2], R[3], …, R[9]. Разумеется, мы должны выбрать ту цифру D, для которой величина R[D] наибольшая. Если наибольшее значение R достигается для нескольких различных цифр, то выбираем наибольшую из этих цифр. Понятно, что это и будет первая цифра искомого субчисла.

А дальше всё уже понятно. Итак, первая цифра наибольшего общего субчисла – это цифра D. В первом числе мы взяли ту из цифр D, которая находится на месте номер p1[D], а во втором – на месте номер p2[D]. Понятно, что в строках n1[p1[D]+1..len1] и n2[p2[D]+1..len2] имеется общая подпоследовательность длины R[D]-1, что подпоследовательности большей длины в этих строках нет.

Точно также, как и раньше, ищем в n1[p1[D]+1..len1] и n2[p2[D]+1..len2] первую цифру их наибольшего общего субчисла, вернее, не совсем субчисла – теперь эта цифра может быть и 0. Находим p1[0] – номер самой первой (самой левой) цифры 0 в строке n1[p1[D]+1..len1], p2[0] – номер самой первой (самой левой) цифры 0 в строке n2[p2[D]+1..len2] и R[0]= r[p1[0], p2[0]], затем, аналогично, p1[1], p2[1] и R[1], …, p1[9], p2[9] и R[9]. Выбираем аналогично вторую цифру – это цифра D, которой соответствует наибольшее значение R[D], а если таких цифр несколько, то наибольшая из них.

И повторяем процесс, пока не найдём все цифры ответа. А количество цифр в ответе мы знаем уже давно – как только мы нашли первую цифру ответа.

Оценим сложность алгоритма. Пусть N – наибольшая длина входных строк. Тогда заполнение массива r требует $O(N^2)$ времени. Вторая часть алгоритма требует O(N) операций. В самом деле, каждый из индексов p1[0], …, p1[9] пробегает один раз всю строку n1, и, конечно, каждый из индексов p2[0], …, p2[9] пробегает один раз всю строку n2), т.е. на вычисление всех индексов занимает не более 10(len1+len2) = O(N) действий (обратите внимание на то, что каждый индекс только увеличивается в процессе работы ; вычисление каждой цифры ответа требует не более 10 сравнений, а длина ответа не превосходит min(len1, len2), то есть и на это действие уходит в сумме не более O(N) операций. Итого, получаем, что предложенное решение требует $O(N^2)$ времени при затратах памяти $O(N^2)$.

Паскаль-реализация приведена в файле **`common.pas`**