Кротков Павел Андреевич,
Ульянцев Владимир Игоревич

ЗАДАЧА «СОБЕСЕДОВАНИЕ»

Этой статьей мы продолжаем цикл публикаций олимпиадных задач для школьников по информатике. Решение таких задач и изучение разборов поможет Вам повысить уровень практических навыков программирования и подготовиться к олимпиадам по информатике.

В этой статье рассматривается задача «Собеседование», которая предлагалась на Четвертой индивидуальной Интернет-олимпиаде по программированию в 2011–2012 учебном году. Материалы этой олимпиады можно найти на сайте http://neerc.ifmo.ru/school/io/.

УСЛОВИЕ ЗАДАЧИ

На собеседовании при поступлении на работу в компанию Sasnart кандидатам предлагается решить задачу о наибольшей общей возрастающей подпоследовательности двух последовательностей чисел.

Суть задачи сводится к следующему: из последовательности чисел a_i необходимо выделить подпоследовательность a_{i_k} такую, что она:

– возрастает, то есть для любого k верно, что $a_{i_k} < a_{i_{k+1}}$;

– является подпоследовательностью последовательности b_i;

– имеет длину не меньшую, чем все последовательности, обладающие предыдущими двумя свойствами.

Вам же, для проверки правильности решения кандидатами этой задачи, необходимо научиться вычислять хотя бы длину такой подпоследовательности.

Формат входного файла

В первой строке входного файла заданы числа n и m ($1 \le m, n \le 5\,000$) – длины последовательностей a_i и b_i соответственно. Вторая и третья строки содержат соответственно по n и m натуральных чисел, не превосходящих 10 000, – сами последовательности.

Формат выходного файла

В выходной файл выведите единственное целое число – длину последовательности, обладающей описанными свойствами.

Примеры входных и выходных данных

interview.in	interview.out
6 5 2 3 1 4 6 5 1 2 5 4 6	3

РАЗБОР ЗАДАЧИ

Приведенная задача является комбинацией двух задач: поиска наибольшей возрастающей подпоследовательности [1] и поиска наибольшей общей подпоследовательности двух последовательностей чисел [2]. Это классические задачи, решаемые с помощью техники динамического программирования. Данная задача также решается с помощью этой техники.

Рассмотрим решение подзадачи, из ответа к которой мы сможем извлечь ответ на исходную задачу. Обозначим за s первые i элементов последовательности a, а за t – первые j элементов последовательности b. Далее обозначим за $answer[i][j]$ длину такой наибольшей общей возрастающей подпоследовательности s и t, что последним ее элементом является b_j. Тогда можно заметить, что ответом на исходную задачу будет $\max_{j=1}^{m}(answer[n][j])$.

Для поиска $answer[i][j]$ воспользуемся упомянутой уже техникой динамического программирования. Будем искать верное значение $answer[i][j]$, основываясь на значениях $answer$ с меньшими параметрами i и j. Заметим, что если $answer[i][j] = k > 1$, то существует такой $x < j$, что $b[x] < b[j]$ и $answer[i][x] = k - 1$. Заметим также, что если $a[i] \neq b[j]$, то $answer[i][j] = answer[i-1][j]$. Из сказанного выше следует общая формула для $answer[i][j]$:

$$answer[i][j] = \begin{cases} answer[i-1][j], & \text{если } a[i] \neq b[j]; \\ \max_{k=1}^{j-1}(answer[i][k]) + 1, & \text{если } a[i] = b[j]. \end{cases}$$

В листинге 1 приведена программа, вычисляющую значения $answer$ по указанной формуле. Приведенная программа будет работать правильно, однако асимптотическое время ее работы равно $O(nm^2)$, и из-за этого программа при тестировании превысит ограничение на максимальное время работы. Значит, решение требует некоторого усовершенствования.

Применим две оптимизации, предложенные в [3]. Во-первых, избавимся от цикла,

Листинг 1. Реализация описанного алгоритма

```
for i := 1 to n do begin
  current := 0;
  for j := 1 to m do begin
    if (a[i] = b[j]) then begin
      max := 0;
      for k := 1 to j - 1 do
        if ((b[k] < b[j]) and (answer[i][k] > max)) then
          max := answer[i][k];
      answer[i][j] := max + 1;
    end else begin
      answer[i][j] := answer[i - 1][j];
    end;
  end;
end;

globalAnswer := 0;
for j := 1 to m do
  if (answer[n][j] > globalAnswer) then
    globalAnswer := answer[n][j];
```

который ищет максимальное значение всех $answer[i][x]$, таких что $x < j$ и $b[x] < b[j]$. Мы знаем, что в тот момент, когда это значение будет использоваться, $b[j]$ будет равно $a[i]$. Значит, на проходе номер i мы можем считать текущий максимум по всем элементам $answer[i][x]$, таким что $b[x] < a[i]$. Этой оптимизацией мы уже достаточно уменьшим время работы алгоритма, сделав его равным $O(nm)$.

Вторая оптимизация не повлияет на время работы, однако уменьшит количество используемой памяти, а также сделает программу короче и элегантнее. Заметим, что при пересчете значения $answer[i][j]$ мы используем только значения $answer[i][x]$, где $x < j$, и $answer[i-1][j]$ Поэтому можно отказаться от использования двумерного массива, хранить только последний его слой и сразу же обновлять значения в нем. Идея данного улучшения применима ко многим алгоритмам динамического программирования. В листинге 2 приведена реализация модифицированного алгоритма, использующего $O(m)$ памяти и работающего за время $O(nm)$.

Листинг 2. Реализация модифицированного алгоритма

```
globalAnswer := 0;

for i := 1 to n do begin
  current := 0;
  for j := 1 to m do begin
    if (a[i] = b[j]) then begin
      if (current + 1 > answer[j]) then
        answer[j] := current + 1;
      if (answer[j] > globalAnswer) then
        globalAnswer := answer[j];
    end;
    if ((a[i] > b[j]) and (answer[j] > current)) then
      current := answer[j];
  end;
end;
```

Источники

1. *Schensted C.* Longest Increasing and Decreasing Subsequences / Classic Papers in Combinatorics. Modern Birkhduser Classics. 1987. P. 299–311.

2. *Скиена С.* Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. СПб.: БХВ-Петербург, 2011.

3. *Yang I., Huang C., Chao K.* A fast algorithm for computing a longest common increasing subsequence // Information Processing Letters. 2005. Vol. 93. P. 249–253.

Кротков Павел Андреевич,
студент третьего курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике,

Ульянцев Владимир Игоревич,
студент шестого курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике.