Практические задания для II-го курса. 2010-2011 уч.год, осенний семестр. Группа 205

Задание №1 Diff

Требуется написать программу, которая находит различия между двумя файлами. В зависимости от варианта, файлы следует интерпретировать как последовательности лексем следующих типов:

- строка
- слово
- байт

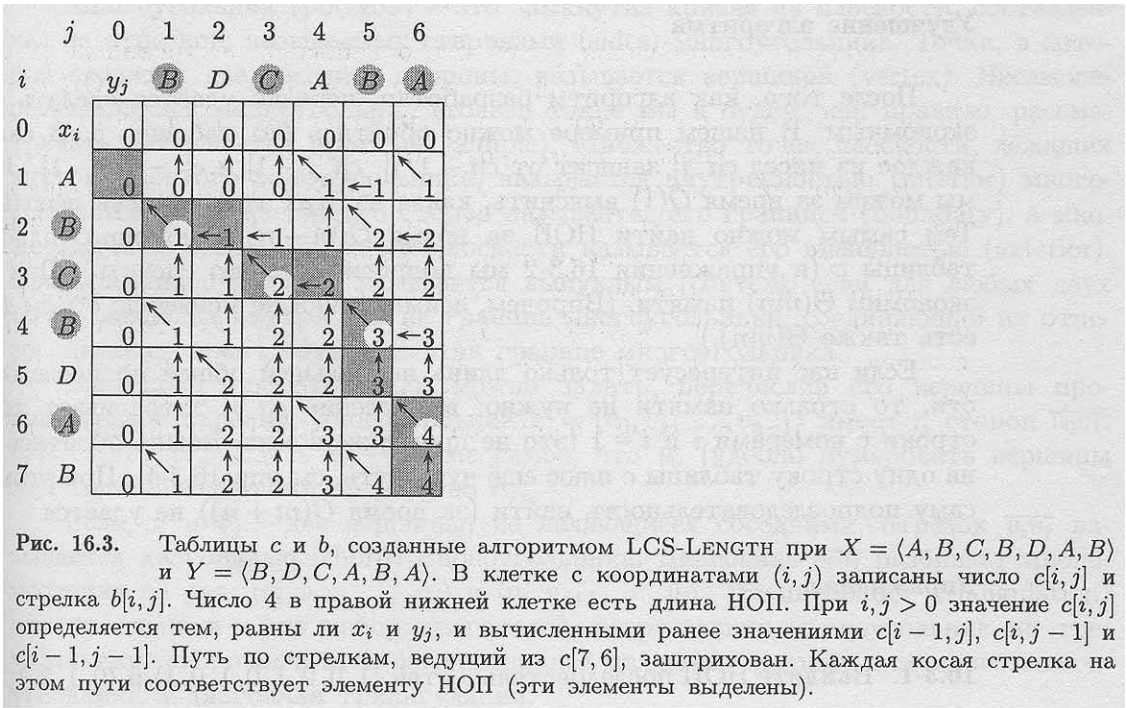
Различия между последовательностями программа должна выдать в виде **скрипта** (последовательность команд), позволяющего получить вторую последовательность из первой с помощью команд трех видов:

- +(лексема) – добавить указанную лексему
- -(количество_лексем) – пропустить указанное количество лексем
- *(количество_лексем) – использовать указанное количество лексем первой последовательности

Пример: пусть **B**D**C**A**BA**, A**BCB**D**A**B – последовательности лексем-символов.
Искомый скрипт, описывающий различия выглядит так:
+A*1-1*1-1*1+D*1+B

Обратите внимание, что таких скриптов может быть найдено несколько. В данной задаче требуется найти скрипт, содержащий максимальное количество операций *. Данная задача сводится к задаче поиска *наибольшей общей подпоследовательности (НОП)* двух последовательностей и может быть решена с использованием следующего алгоритма:

```
LCS-LENGTH(X, Y)
 1  m ← length[X]
 2  n ← length[Y]
 3  for i ← 1 to m
 4       do c[i, 0] ← 0
 5  for j ← 0 to n
 6       do c[0, j] ← 0
 7  for i ← 1 to m
 8       do for j ← 1 to n
 9            do if x_i = y_j
10                 then c[i, j] ← c[i − 1, j − 1] + 1
11                      b[i, j] ← «↖»
12                 else if c[i − 1, j] ⩾ c[i, j − 1]
13                      then c[i, j] ← c[i − 1, j]
14                           b[i, j] ← «↑»
15                      else c[i, j] ← c[i, j − 1]
16                           b[i, j] ← «←»
17  return c, b
```



i		j	0	1	2	3	4	5	6
			y_j	B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B		0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C		0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B		0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D		0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A		0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B		0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Рис. 16.3. Таблицы c и b, созданные алгоритмом LCS-LENGTH при $X = \langle A, B, C, B, D, A, B \rangle$ и $Y = \langle B, D, C, A, B, A \rangle$. В клетке с координатами (i, j) записаны число $c[i, j]$ и стрелка $b[i, j]$. Число 4 в правой нижней клетке есть длина НОП. При $i, j > 0$ значение $c[i, j]$ определяется тем, равны ли x_i и y_j, и вычисленными ранее значениями $c[i-1, j]$, $c[i, j-1]$ и $c[i-1, j-1]$. Путь по стрелкам, ведущий из $c[7, 6]$, заштрихован. Каждая косая стрелка на этом пути соответствует элементу НОП (эти элементы выделены).

Таблица b, созданная процедурой LCS-LENGTH, позволяет быстро найти НОП последовательностей $X = \langle x_1, x_2, \ldots, x_m \rangle$ и $Y = \langle y_1, y_2, \ldots, y_n \rangle$. Для этого надо пройти по пути, указанному стрелками, начиная с $b[m, n]$. Пройденная стрелка ↖ в клетке (i, j) означает, что $x_i = y_j$ входит в наибольшую общую подпоследовательность. Вот как это реализовано в рекурсивной процедуре PRINT-LCS (НОП для X и Y печатается при вызове PRINT-LCS$(b, X, length[X], length[Y])$):

```
PRINT-LCS(b, X, i, j)
1  if i = 0 или j = 0
2     then return
3  if b[i, j] = «↖»
4     then PRINT-LCS(b, X, i − 1, j − 1)
5          напечатать x_i
6  elseif b[i, j] = «↑»
7     then PRINT-LCS(b, X, i − 1, j)
8  else PRINT-LCS(b, X, i, j − 1)
```

Требования/выходным данным Программа должна
обеспечивать 2 режима работы:
Режим поиска различий
diff --compare filename1 filename2 Сравнение двух файлов, получение различий в поток стандартного вывода в виде скрипта в формате {+|-|*}<параметр><перевод строки (\n)>

Для рассмотренного примера должен быть получен скрипт:

+A
*1 -
1
*1 -
1
*1
+D
*1
+B
(после последнего символа B также '\n')

Режим обновления
diff --patch filename patch_filename В режиме обновления программа должна применить скрипт из файла patch_filename к последовательности лексем из файла filename.

Варианты

1 строка (разделены '\n')
2 слово (разделитель – символ, для которого верно isspace(…))
3 байт

Литература Т.Кормен, Ч.Лейзерсон, Р.Ривест «Алгоритмы: построение и анализ», МЦНМО, Москва, 2000, стр. 300-304

Правила сдачи

1 Задание ДОЛЖНО быть сдано в МЗ 4-го либо 11-го (с потерей 1-го балла) ноября 2010 г.
2 К моменту сдачи исходные тексты программы должны пройти проверки в ejudge
3 Для успешной сдачи задания необходимо в день сдачи отослать исходные тексты в архиве <ваша_фамилия>.zip на адрес prak@mlab.cs.msu.su
4 При обнаружении плагиата в любом виде (чужие измененные / не измененные исходники, «мне помогали» и т.п.) в первый раз пишется докладная в учебную часть и выдается новое более сложное задание. Меры в случае повторного обнаружения плагиата на усмотрение учебной части.
5 Программа должна быть написана на языке Си (стандарт Ansi C 89)
6 В состав отчетных материалов по заданию входят:
Исходные тексты
Самостоятельно подготовленные и проверенные тестовые сценарии
В случае нескольких исходных файлов – Makefile
Документация по всем функциям (с описанием параметров и возвращающим значениям) и типам (структурам) программы. Допустимо использование Doxygen (http://www.stack.nl/~dimitri/doxygen/)
7 При оценивании задания учитывается: корректность работы, качество кода, наличие и качество всех отчетных материалов.