2011 Summer Entrance Examination

Department of Creative Informatics
Graduate School of Information Science and Technology
The University of Tokyo

Programming

INSTRUCTIONS

1. Do not open this problem brochure until the signal to begin is given.
2. Write your examinee ID below on this cover.
3. An answer sheet and a draft sheet accompany this brochure. Write down your examinee ID on these sheets.
4. The USB memory delivered beforehand to each examinee contains ASCII text files `c1.txt`, `c2.txt`, `s1.txt`, `s2.txt` and `s3.txt`.

   Before the examination starts, copy these files to your PC and browse them. Verify that you can see the text files and then remove your hands from your PC. If you cannot read the files properly, consult the test supervisor. The contents of the USB memory are common to all examinees.
5. You may choose your favorite programming languages.
6. You may consult only one printed manual of a programming language during the examination. *You can use or copy any libraries or program segments existing in your PC, but you cannot connect to the Internet.*
7. By the end of the examination, make a directory/folder on your PC, whose name is the same as your examinee ID, and put your program files and related files into the directory/folder. Copy the directory/folder onto the delivered USB memory.
8. At the end of the examination, the USB memory, the answer sheet and the draft sheet are collected.
9. After these are collected, stay at your seat, until all examinee program results have been checked briefly by the test supervisor.
10. After the brief check, try to save your program execution environment on the PC so that you can run your program as soon as possible during the oral examination in the afternoon.
11. *Leave your PC and this brochure together in the room* for the oral examination and leave the room until you are called.

Examinee ID _____

In this problem, we discuss the topic of dictionary compression which is one type of lossless data compression. The dictionary compression algorithm utilizes the repeated occurrences of the same substring and compresses a given string by replacing the substring (replacement-target string) with another shorter string (replacement-indication string). Decompression means restoring the compressed string to the original string. We define the source string as a string before compression and the compressed string as a string after compression.

In this problem, for the n-length source string S ($n \leq 1000$), we denote the substring from a to b as $S[a;b]$ ($0 \leq a \leq b \leq n-1$). From Q1 to Q5, the length of the source strings is $n \leq 1000$. We will remove this restriction on the length of the source string in Q6. Source strings contain only lower case letters (a-z), spaces, periods and commas. In addition to these characters, compressed strings contain the replacement-indication strings each of which is formed by three decimal digits ranging from 000 to 993. Note that the leading '0' characters cannot be eliminated. A replacement-target string has a fixed length of six characters.

For i, j ($0 \leq i < j \leq n-6$), when the substrings $S[i;i+5]$ and $S[j;j+5]$ are equivalent, $S[j;j+5]$ is a replacement-target string and replaced with the replacement-indication string represented by three decimal digits giving the index i (Figure 1). Afterwards, the substring $S[j;j+5]$ in the source string is no longer a valid area for replacement. The substring is compressed from 6 characters to 3 characters by replacement.

We add the following restriction such that there is a unique i corresponding to every replacement-indication string: Increment j from $j = 1$ to seek $S[j;j+5]$. If there are more than one possibility for $S[i;i+5]$, choose the minimum of i as the replacement-indication string.

When $i + 5 \geq j$, the replacement-target string overlaps with the source substring for replacement. If there are such overlaps during the decompression, the source string can be generated by decompressing the non-overlapped part sequentially, as shown in Figure 2.

Example	Source string	Compressed string
	`vwabcdefxyabcdefst`	`vwabcdefxy002st`
	`abababababababab`	`ab000000ab`

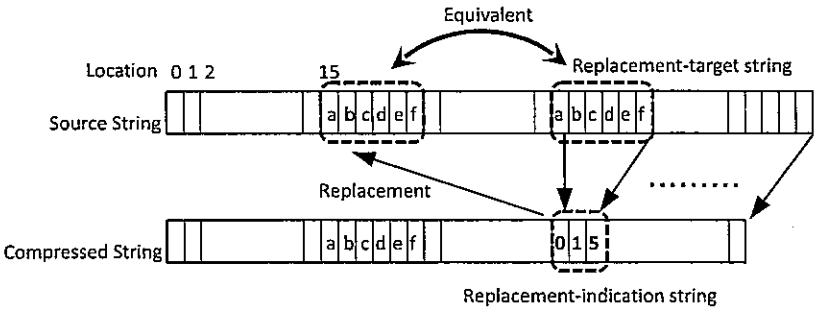

Figure 1: Replacement of a string

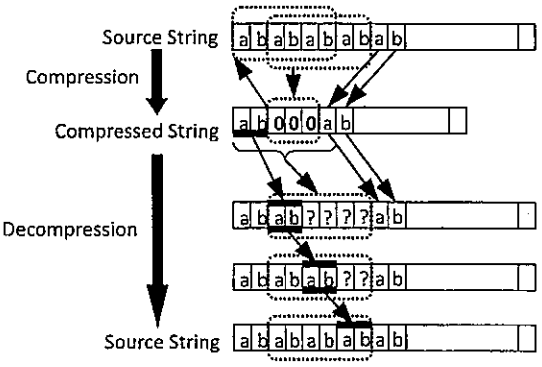

Figure 2: An example of overlapping replacement

Q1 Answer the following questions without making a program.

1-1 Write the source string for the compressed string "`aabbba000c001008a`".

1-2 Write the compressed string for the source string "`aabbccddaabbccddbbccddaa`".

Q2 Calculate the numbers of replacement-indication strings represented by three decimal digits existing in the compressed strings stored in `c1.txt` and `c2.txt` by writing and executing a program. In ASCII code, 'a' = 0x61 (97 in decimal notation), ' ' = 0x20 (32 in decimal notation), ',' = 0x2C (44 in decimal notation), '.' = 0x2E (46 in decimal notation) and '0' = 0x30 (48 in decimal notation).

Q3 Make a program which creates a dictionary to compress the source string. The dictionary is a data structure containing 6-length substrings and the minimum value of the locations of corresponding substrings. Figure 3 shows an example of the structure of the compression-dictionary.

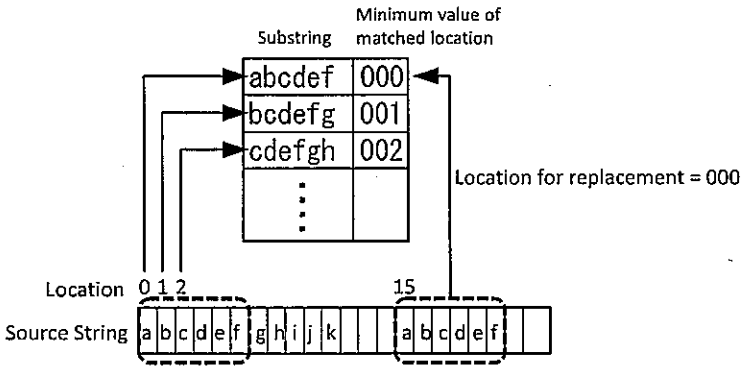

Figure 3: An example of the structure of the compression-dictionary

Create a dictionary to look up the 6-length substrings from the n-length string S ($n \leq 1000$). To be specific, take a substring $S[i; i+5]$ and look up using this substring in the dictionary while i is incremented from 0. If looking up succeeds, return the minimum value of the location of this substring. If looking up fails, add this substring and the location i to the dictionary.

Using this program, make a dictionary from the string stored in the `s1.txt` (89-length). Write the number of locations for substrings in the dictionary.

Q4 Make a program which compresses a given source string. Write the length and the last 10 characters of the compressed strings processed by this program for the source strings stored in `s1.txt` and `s2.txt`, respectively.

Q5 Make a program which decompresses a given compressed string. Write the length and the last 10 characters of the source strings decompressed by this program for the compressed strings stored in `c1.txt` and `c2.txt`, respectively.

Q6 To handle strings with a length over 1000, extend the compression program from Q4 to split the strings into blocks of 1000 characters with any remaining characters at the end. Similarly, extend the decompression program written in Q5 for source strings with a length over 1000.

Use the source strings stored in `s3.txt` to confirm the correct behavior by compressing a source string using the extended compression program and decompressing the generated compressed string using the extended decompression program.

This page is empty.

This page is empty.