CSCI-550 Projects 4 and 5 Suffix Array by Induced Sorting

The goal of Projects 4 and 5 is to implement SAIS algorithm that calculates Suffix Array for a given string T of length *n* in linear time, O(n). Your program will read the text T from an input file, then will calculate the Suffix Array for T, and then will calculate BWT (Burrows-Wheeler's Transform) string using the Suffix Array and T and will print out BWT.

Project 4 is worth 200pts and Project 5 is worth 400pts (and of course, to do Project 5, you need complete Project 4 anyway).

For Project 4, implement Step 0 and Step 1 of SAIS provided in the file "SAIS algorithms Elena simplified.pdf".

Specification Requirements for Project 4.

(1) Read in input file using ***cin***. Input file might contain ***endl*** (end of line) characters, so these characters are not part of the given input string. Your program should read in all the lines from the input file and concatenate strings from the different lines (use C++ ***ostringstream*** object to do this, concatenation using ostringstream is much faster than concatenation of two strings using "+" operator) into a single input string T. For example, if the input file contains:

```
ROAD
STAR
CAT
```

then T = "ROADSTARCAT".

(2) At the beginning of SAIS algorithm, we need to append character $ to the end of T (this character is supposed to be the smallest alphabetically out of characters of T and it does not occur in T).

Before calling SAIS algorithm, you need to convert the input string T into an array (vector) of integers, where each character of T is represented by an integer (this is not ASSI representation, this is your own representation). All distinct characters of T will be renamed by an integer: the character $ will be 0, and the other characters will be given new integer-names of consecutive integers 1, 2, 3, …, k (where k is the size of alphabet of T).

To do this (to convert characters into integers), maintain an array (or a vector) indexed by characters that determines which one out of 256 possible characters is present in the input string T. For example, assume that you declared an integer vector ***count*** of size 256. Then count[(int)T[i]] = 1, if a character *ch* is in T; and count[j] = 0 if *ch* is not in T and *j* = (int)*ch*. In other words, for all characters of T, *ch* = T[i] where i = 0, 1,…,n, convert the character into an integer, (int)T[i], and use it as an index of ***count*** to set count[(int)ch] to 1.

Next, count the alphabet of T (the total number of distinct characters including $ in T) and build a hash table ***amap***, another array (vector), that is indexed by an integer (new name of a distinct character in T) and whose value is a character, the corresponding character. We need this hash table to build BWT from SA after running SAIS. The total number of entries in ***count*** whose values are 1 will give you alphabet size. Name characters 1, 2, 3,…, k according their order in ***count.*** Scan array ***count*** from left to right, and if count[j] = 1 (meaning ***(char)j*** is in T), then assign to character (char)j the next available integer-name ***m***, and set ***amap***[m] = (char)*j*.

Example: for the input T above, T = "ROADSTARCAT$".

Alphabet size is 8 (the number of distinct characters in T is 7: A, C, D, O, R, S, T, and plus character $)

Index	…	65	..	68	..	79	..	82	83	84	…
count		A		D		O		R	S	T	
New integer-name		1		2		3		4	5	6	

So, T is converted to integer-array (each character is given a new integer name as shown in the table above):

"ROADSTARCAT" is converted to {5 4 1 3 6 7 1 5 2 1 7 0}.

The hash table ***amap*** that converts integer-names back to characters is:

Index	0	1	2	3	4	5	6
amap	$	A	D	O	R	S	T

(3) To make implementation easier, delegate different tasks to different functions. Inside main() function, you need to put mostly function calls, and everything else should be calculated inside those functions.

(4) Output of Project 4 is the content of Suffix Array after Steps 0 and 1.

For example, for the input string T = "ROADSTARCAT"

The output is: `11 2 6 9 8 3 1 7 0 4 10 5`
Output format is:

<int><space><int><space>….<int><space><endl>

After the last integer, there is a space and then ***endl.***

Submission: submit the file ***proj4.cpp*** with your program to Project4 on *turnin*.

Project 5.

Implement Step 2, Step 3 and Step 4 of SAIS algorithm provided in the file "SAIS algorithms Elena simplified.pdf". This will finish calculating the Suffix Array in linear time.

In addition, you need to calculate BWT for the input string using the calculated Suffix Array. Recall that for each *index i* = 0, 1, …, n, SA[i] = p, the position *p* at which the suffix T[p…n] starts such that T[p…n] is i-th in the sorted alphabetic order out of all suffixes of T. If SA[i] = p, then BWT[i] = T[p-1].

For example, for T = "ROADSTARCAT", BWT = "TOTCRARADAS".

The output of Project 5 is BWT in format: <BWT><endl>.

Submission: submit the file ***proj5.cpp*** with your program to Project 5 on ***turnin***.

Grading: Grading for both projects will be based on how many tests your program passes and how well your program follows requirements (main() contains function calls, and your program has many functions each is delegated to do a specific task).