Ph.D. Qualifying Exam: Analysis of Algorithms

This is a closed book exam. The total score is 100 points. Please answer all questions.

1. The Burrows-Wheeler transform (BWT) converts a string s of length n to another string t of length n, so that t is more likely to be represented as runs of the same character. We require that * is always the last character in string s indicating its end. We also assume that * alphabetically sort after all other characters. A remarkable property of BWT is that there is an inverse transform to decode t back to s.

   Answer the following questions regarding BWT.

(10 points) (a) The BWT encoding algorithm is given below as BWT(s). Please describe the output of BWT(s) when s =`MISSISSIPPI*`.

function BWT (string s)

Input: s is a string that must end with the * character

1. create a table, where rows are all possible right rotations of s including s
2. sort rows in the table alphabetically // treat each row as a word and sort the words
3. return (last column of the table)

Note: All possible right rotations of abc* including abc* are

abc*
*abc
c*ab
bc*a

Solution:

Table of all right rotations of s:	Sorting the rows, we get
`MISSISSIPPI*`	`IPPI*MISSISS`
`*MISSISSIPPI`	`ISSIPPI*MISS`
`I*MISSISSIPP`	`ISSISSIPPI*M`
`PI*MISSISSIP`	`I*MISSISSIPP`
`PPI*MISSISSI`	`MISSISSIPPI*`
`IPPI*MISSISS`	`PI*MISSISSIP`
`SIPPI*MISSIS`	`PPI*MISSISSI`
`SSIPPI*MISSI`	`SIPPI*MISSIS`
`ISSIPPI*MISS`	`SISSIPPI*MIS`
`SISSIPPI*MIS`	`SSIPPI*MISSI`
`SSISSIPPI*MI`	`SSISSIPPI*MI`
`ISSISSIPPI*M`	`*MISSISSIPPI`

The output of BWT(s) is thus the last column of the table on the right: `SSMP*PISSIII`.

(20 points) (b) The BWT decoding algorithm is given below as inverseBWT(t). Please describe the output of inverseBWT(t) when t=`STNENESE*E`. You must show intermediate steps to derive the final output.

function inverseBWT (string t)

Input: t is a string encoded by applying BWT on some unknown s and contains the special character * but does not necessarily end with *

1. create empty table
2. $n \leftarrow$ length(t)
3. repeat n times
4. insert t as a column of table before the first column of the table
   // Note: the first insert creates the first column
5. sort rows of the table alphabetically
6. return (row that ends with the * character)

Solution: The length of t is $n = 10$.

Table:

t:	0	01	01	012	012	0123	0123	01234	01234	012345	012345
S	E	SE	EE	SEE	EE*	SEE*	EE*T	SEE*T	EE*TE	SEE*TE	EE*TEN
T	E	TE	EN	TEN	ENN	TENN	ENNE	TENNE	ENNES	TENNES	ENNESS
N	E	NE	ES	NES	ESS	NESS	ESSE	NESSE	ESSEE	NESSEE	ESSEE*
E	E	EE	E*	EE*	E*T	EE*T	E*TE	EE*TE	E*TEN	EE*TEN	E*TENN
N	N	NN	NE	NNE	NES	NNES	NESS	NNESS	NESSE	NNESSE	NESSEE
E	N	EN	NN	ENN	NNE	ENNE	NNES	ENNES	NNESS	ENNESS	NNESSE
S	S	SS	SE	SSE	SEE	SSEE	SEE*	SSEE*	SEE*T	SSEE*T	SEE*TE
E	S	ES	SS	ESS	SSE	ESSE	SSEE	ESSEE	SSEE*	ESSEE*	SSEE*T
*	T	*T	TE	*TE	TEN	*TEN	TENN	*TENN	TENNE	*TENNE	TENNES
E	*	E*	*T	E*T	*TE	E*TE	*TEN	E*TEN	*TENN	E*TENN	*TENNE

Table:

t:	0123456	0123456	01234567	01234567	012345678	012345678
S	SEE*TEN	EE*TENN	SEE*TENN	EE*TENNE	SEE*TENNE	EE*TENNES
T	TENNESS	ENNESSE	TENNESSE	ENNESSEE	TENNESSEE	ENNESSEE*
N	NESSEE*	ESSEE*T	NESSEE*T	ESSEE*TE	NESSEE*TE	ESSEE*TEN
E	EE*TENN	E*TENNE	EE*TENNE	E*TENNES	EE*TENNES	E*TENNESS
N	NNESSEE	NESSEE*	NNESSEE*	NESSEE*T	NNESSEE*T	NESSEE*TE
E	ENNESSE	NNESSEE	ENNESSEE	NNESSEE*	ENNESSEE*	NNESSEE*T
S	SSEE*TE	SEE*TEN	SSEE*TEN	SEE*TENN	SSEE*TENN	SEE*TENNE
E	ESSEE*T	SSEE*TE	ESSEE*TE	SSEE*TEN	ESSEE*TEN	SSEE*TENN
*	*TENNES	TENNESS	*TENNESS	TENNESSE	*TENNESSE	TENNESSEE
E	E*TENNE	*TENNES	E*TENNES	*TENNESS	E*TENNESS	*TENNESSE

Table:

t:		
	0123456789	0123456789
S	SEE*TENNES	EE*TENNESS
T	TENNESSEE*	ENNESSEE*T
N	NESSEE*TEN	ESSEE*TENN
E	EE*TENNESS	E*TENNESSE
N	NNESSEE*TE	NESSEE*TEN
E	ENNESSEE*T	NNESSEE*TE
S	SSEE*TENNE	SEE*TENNES
E	ESSEE*TENN	SSEE*TENNE
*	*TENNESSEE	TENNESSEE* ¡-- The solution
E	E*TENNESSE	*TENNESSEE

(15 points) (c) Let t be the BWT transform of s, i.e., t=BWT(s). Study the solution in (b). From the insight gained, what is the relationship between s and inverseBWT(t)? Prove your claim.

Solution: We show that s and inverseBWT(t) are two equal strings. Using loop invariant argument, we can show that the decoding algorithm inverseBWT() will grow the prefixes of the rows, rotated versions of s, from t until the length of the string t is reached.

(15 points) (d) Let n be the length of the input string s and t in the two functions. Please analyze the time complexity of both the encoding and decoding algorithms.

Solution:

If we use comparison-based sorting, we must also consider that the time cost of each comparison is not constant, but a linear function of the length of the strings. A sorting of n strings each of length n will thus take $O(n^2 \lg n)$ time.

Accordingly, the time complexity of BWT(s) is $O(n^2 \lg n)$; and for inverseBWT(t), the time complexity is $O(n(\lg n)(\sum_{i=1}^{n} i)) = O(n^3 \lg n)$.

(30 points) 2. The longest path problem finds a longest simple path in a graph. We are here concerned with directed and edge-weighted graphs. The length of a path is the summation of the weights of edges on the path.

A Hamiltonian path is a simple path that visits each vertex in a graph exactly once. The Hamiltonian path problem answers whether a Hamiltonian path exists in a graph.

Show that the Hamiltonian path problem is equivalent to a special case of the longest path problem.

Solution:

For a given graph of n vertices in a Hamiltonian path problem, we can formulate a longest path problem by treating the weight of each edge as 1.

Longest path $\Rightarrow$ Hamiltonian path: If solving the longest path returns a simple path of length $n-1$, it must be a Hamiltonian path by definition;

Hamiltonian path $\Rightarrow$ longest path: If there is at least one Hamiltonian path in the graph, it must be a longest simple path in a graph of n-nodes and solving the longest path problem thus must return some Hamiltonian path.

(10 points) 3. Contemplate in what way algorithm design & analysis may benefit your future doctoral study. You will need to give a concrete research topic that could evolve into your dissertation and describe how you may address the topic with a technique in algorithm analysis.

Solution: Grading: naming a topic area (5 points); naming a relevant algorithm technique that is defendable (5 points);