Przemysław Uznański March 20, 2018

Advanced Data Structures

Spring Semester 2018

Exercise Set 5

Assume we have $T[1, n]$ over alphabet Σ. T is such that its last letter is a special character $\#$ such that $\# < c$ for $c \in \Sigma$ and $\#$ does not appear anywhere else in T. Consider following operation: we write matrix M such that its rows $M[1], M[2], \ldots, M[n]$ are all cyclic rotations of T, *sorted lexicographically*, that is $M[1] < M[2] < \ldots < M[n]$.

Definition. *The text written on the* last column *of M is called its* Burrows-Wheeler transformation. *That is,* $\mathsf{BWT}[j] = M[j][n]$.

For example:

$$T = \texttt{mississippi\#}$$

$M =$

#	m	i	s	s	i	s	s	i	p	p	i
i	#	m	i	s	s	i	s	s	i	p	p
i	p	p	i	#	m	i	s	s	i	s	s
i	s	s	i	p	p	i	#	m	i	s	s
i	s	s	i	s	s	i	p	p	i	#	m
m	i	s	s	i	s	s	i	p	p	i	#
p	i	#	m	i	s	s	i	s	s	i	p
p	p	i	#	m	i	s	s	i	s	s	i
s	i	p	p	i	#	m	i	s	s	i	s
s	i	s	s	i	p	p	i	#	m	i	s
s	s	i	p	p	i	#	m	i	s	s	i
s	s	i	s	s	i	p	p	i	#	m	i

$$\mathsf{BWT}(T) = \texttt{ipssm\#pissii}$$

Exercise 1:
Reverse transformation: Show that given only BWT, we can recover text T in time $\mathcal{O}(n)$.

Hint: Each letter of input appears once in the first column $M[\cdot][1]$ and once in the last column $M[\cdot][n]$. Call the relation mapping those occurences LF (last-first). Show that LF can be recovered using rank_c queries, one query per entry. You can use either wavelet trees for $\mathcal{O}(n \log |\Sigma|)$ time, or observe that those offline queries can be batch processed.

Exercise 2:
Show that given only BWT, we can recover its suffix-array in time $\mathcal{O}(n)$.

Hint: Use LF structure from previous exercise.

Exercise 3:
Show that storing only *wavelet tree* of BWT and some $\mathcal{O}(n/t \cdot \log n)$ bits of auxilary data, we can access suffix array values in $\mathcal{O}(t \cdot \log |\Sigma|)$ time per value.

Hint: Store explicitly some n/t values from suffix array.

Exercise 4:
Show that given only *wavelet tree* of BWT, we process the queries of form *is pattern P in T* in time $\mathcal{O}(|P| \cdot \log |\Sigma|)$, and given additionally its suffix array SA, we can list all the occurences in time $\mathcal{O}(|P| \cdot \log |\Sigma| + occ)$.

Hint: Any pattern occurs as a prefix of a suffix and all of its occurences occur on positions $SA[a], SA[a+1], \ldots, SA[b]$ for some a, b. You can go from occurences of P to occurences of xP for some $x \in \Sigma$ using two rank queries (that is, maintain range of occurences in suffix array using right-to-left scan through the pattern).