۱. [۳۲نمره] درست یا نادرست بودن سوالات زیر را همراه با دلیل مشخص کنید.

(a) **T** The Burrows Wheeler Transform is used for pattern matching.

True. See 2nd lecture in the String module.

(b) **T** The "Trie" data structure can be used to match n patterns of maximum length P against a string of length M in $O(M \times P)$ independent of n.

True. Matching a string against a Trie is in the order of the depth of the tree which is P. Repeating this for each position is $O(M \times P)$.

(c) **T** The Burrows Wheeler Transform is used for data compression.

True. The

$$BW$$

transforms brings repeating symbols/characters together (as much as possible) which enables better data compression.

(d) **F** A Suffix Trie can be stored with $O(N \times \log N)$ memory consumption.

False. For a string of length N, all suffixes have $O(N^2)$ number of characters together. Memory consumption in a Trie is order of the total number of characters in the Trie which is $O(N^2)$.

(e) **F** Storing a Suffix Trie requires more memory than a Suffix Tree, but Suffix Trie is faster for pattern matching.

False. While suffix tree consumes less memory due to the use of indices into the string but Suffix Tree is just as fast as a Suffix Trie.

(f) **F** A SuffixArray cannot be used for pattern matching directly. However for a string of length T it can be used to build a Suffix Tree in $O(T)$ and then used for pattern matching against a pattern of length P in $O(P)$

False. A Suffix Array can be used for pattern matching. See Q3 in assignment 7.

(g) **F** Given algorithms learned in class, for a very long string and many short patterns, using the Knuth-Morris-Pratt algorithm is the most efficient way to find all occurrences of the short patterns in the long string.

False. KMP order is $O(P+T)$ for a single matching. For n patterns with max length of P, it is $O(n \times (P+T))$. Building a Suffix Tree is $O(T \log T)$. Matching a single pattern against suffix tree is $O(P)$. So matching all n patterns against the suffix tree would be $O(P \times n)$. Therefore the total order using a Suffix Tree would be $O((T \log T) + P \times n)$. Since n and T are both large numbers, the KMP solution is less efficient since it contains a $n \times T$ term.

۲. [۳۰نمره] رشته mississippi را در نظر گرفته و در رابطه با آن به سوالات زیر پاسخ دهید.

(آ) [۵نمره] تبدیل را برای این رشته بدست آورید و در کادر زیر وارد کنید.

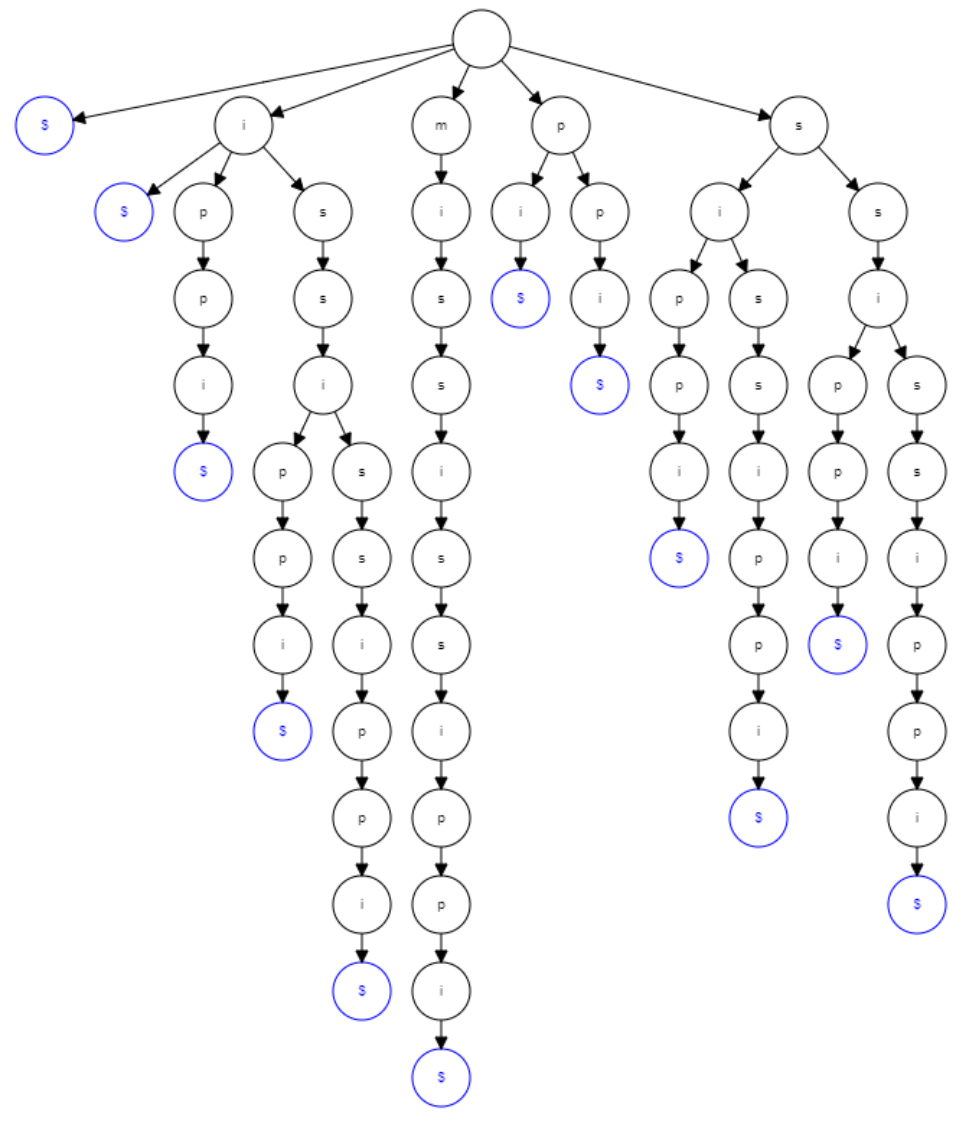
```
Answer:
ipssm$pissii

Rotations:
mississippi$
ississippi$m
ssissippi$mi
sissippi$mis
issippi$miss
ssippi$missi
sippi$missis
ippi$mississ
ppi$mississi
pi$mississip
i$mississipp
$mississippi

Sorted Rotations:
$mississippi
i$mississipp
ippi$mississ
issippi$miss
ississippi$m
mississippi$
pi$mississip
ppi$mississi
sippi$missis
sissippi$mis
ssippi$missi
ssissippi$mi

Transform (Last Column):
ipssm$pissii
```

(ب) [۵نمره] Suffix Trie متناظر با این رشته را رسم کنید.



شکل ۱: Generated By: https://people.ok.ubc.ca/ylucet/DS/Trie.html

(ج) [۵نمره] Suffix Tree متناظر با این رشته در حالتی که پیچیدگی حافظه برای آن $O(|Text|)$

باشد را رسم کنید.

```
     |(1:12)|leaf
tree:|
     |     |     |(6:12)|leaf
     |     |(3:5)|
     |     |     |(9:12)|leaf
     |(2:2)|
     |     |(9:12)|leaf
     |     |
     |     |(12:12)|leaf
     |
     |     |     |(6:12)|leaf
     |     |(4:5)|
     |     |     |(9:12)|leaf
     |(3:3)|
     |     |     |(6:12)|leaf
     |     |(5:5)|
     |     |     |(9:12)|leaf
     |
     |     |(10:12)|leaf
     |(9:9)|
     |     |(11:12)|leaf
     |
     |(12:12)|leaf
Generated By:  http://www.allisons.org/ll/AlgDS/Tree/Suffix/
```

(د) [۱۰نمره] با توجه به الگوریتم $O(n \log n)$ ارایه شده در درس جهت بدست آوردن SuffixArray جداول زیر را پر کرده و سپس آرایه LCP را محاسبه کنید.

Index	CyclicShifts1
0	m
1	i
2	s
3	s
4	i
5	s
6	s
7	i
8	p
9	p
10	i
11	$

Index	SoretdCyclicShifts1
11	$
1	i
4	i
7	i
10	i
0	m
8	p
9	p
2	s
3	s
5	s
6	s

CyclicShifts1-Idx	CyclicShifts2		CyclicShifts1-Idx
11	$	m	0
1	i	s	2
4	i	s	5
7	i	p	8
10	i	$	11
0	m	i	1
8	p	p	9
9	p	i	10
2	s	s	3
3	s	i	4
5	s	s	6
6	s	i	7

Index	SoretdCyclicShifts2	
11	$	m
10	i	$
7	i	p
1	i	s
4	i	s
0	m	i
9	p	i
8	p	p
3	s	i
6	s	i
2	s	s
5	s	s

CyclicShifts2-Idx	CyclicShifts4				CyclicShifts2-Idx
11	$	m	i	s	1
10	i	$	m	i	0
7	i	p	p	i	9
1	i	s	s	i	3
4	i	s	s	i	6
0	m	i	s	s	2
9	p	i	$	m	11
8	p	p	i	$	10
3	s	i	s	s	5
6	s	i	p	p	8
2	s	s	i	s	4
5	s	s	i	p	7

Index	SoretdCyclicShifts4			
11	$	m	i	s
10	i	$	m	i
7	i	p	p	i
1	i	s	s	i
4	i	s	s	i
0	m	i	s	s
9	p	i	$	m
8	p	p	i	$
6	s	i	p	p
3	s	i	s	s
5	s	s	i	p
2	s	s	i	s

CyclicShifts4-Idx	CyclicShifts8								CyclicShifts4-Idx
11	$	m	i	s	s	i	s	s	3
10	i	$	m	i	s	s	i	s	2
7	i	p	p	i	$	m	i	s	11
1	i	s	s	i	s	s	i	p	5
4	i	s	s	i	p	p	i	$	8
0	m	i	s	s	i	s	s	i	4
9	p	i	$	m	i	s	s	i	1
8	p	p	i	$	m	i	s	s	0
6	s	i	p	p	i	$	m	i	10
3	s	i	s	s	i	p	p	i	7
5	s	s	i	p	p	i	$	m	9
2	s	s	i	s	s	i	p	p	6

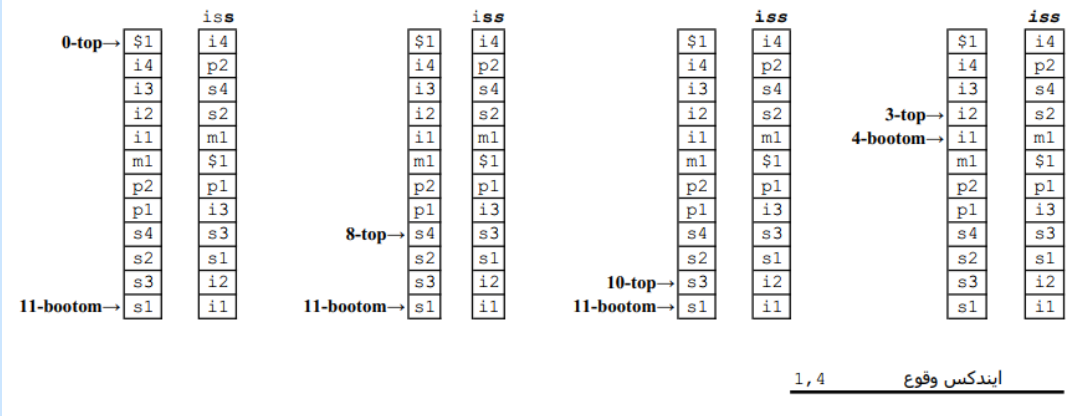
Index	CyclicShifts8							
11	$	m	i	s	s	i	s	s
10	i	$	m	i	s	s	i	s
7	i	p	p	i	$	m	i	s
4	i	s	s	i	p	p	i	$
1	i	s	s	i	s	s	i	p
0	m	i	s	s	i	s	s	i
9	p	i	$	m	i	s	s	i
8	p	p	i	$	m	i	s	s
6	s	i	p	p	i	$	m	i
3	s	i	s	s	i	p	p	i
5	s	s	i	p	p	i	$	m
2	s	s	i	s	s	i	p	p

CyclicShifts8-Idx	CyclicShifts16															CyclicShifts8-Idx
11	$	m	i	s	s	i	s	s	i	p	p	i	$	m	i	s	7
10	i	$	m	i	s	s	i	s	s	i	p	p	i	$	m	i	6
7	i	p	p	i	$	m	i	s	s	i	s	s	i	p	p	i	3
4	i	s	s	i	p	p	i	$	m	i	s	s	i	s	s	i	0
1	i	s	s	i	s	s	i	p	p	i	$	m	i	s	s	i	9
0	m	i	s	s	i	s	s	i	p	p	i	$	m	i	s	s	8
9	p	i	$	m	i	s	s	i	s	s	i	p	p	i	$	m	5
8	p	p	i	$	m	i	s	s	i	s	s	i	p	p	i	$	4
6	s	i	p	p	i	$	m	i	s	s	i	s	s	i	p	p	2
3	s	i	s	s	i	p	p	i	$	m	i	s	s	i	s	s	11
5	s	s	i	p	p	i	$	m	i	s	s	i	s	s	i	p	1
2	s	s	i	s	s	i	p	p	i	$	m	i	s	s	i	s	10

Index	CyclicShifts16															LCP
11	$	m	i	s	s	i	s	s	i	p	p	i	$	m	i	s	
10	i	$	m	i	s	s	i	s	s	i	p	p	i	$	m	i	0
7	i	p	p	i	$	m	i	s	s	i	s	s	i	p	p	i	1
4	i	s	s	i	p	p	i	$	m	i	s	s	i	s	s	i	1
1	i	s	s	i	s	s	i	p	p	i	$	m	i	s	s	i	4
0	m	i	s	s	i	s	s	i	p	p	i	$	m	i	s	s	0
9	p	i	$	m	i	s	s	i	s	s	i	p	p	i	$	m	0
8	p	p	i	$	m	i	s	s	i	s	s	i	p	p	i	$	1
6	s	i	p	p	i	$	m	i	s	s	i	s	s	i	p	p	0
3	s	i	s	s	i	p	p	i	$	m	i	s	s	i	s	s	2
5	s	s	i	p	p	i	$	m	i	s	s	i	s	s	i	p	1
2	s	s	i	s	s	i	p	p	i	$	m	i	s	s	i	s	3

suffixArray	11	10	7	4	1	0	9	8	6	3	5	2

(۵) [۵نمره] با استفاده از تبدیل BW که در قسمت الف بدست آوردید الگوی iss را درون رشته پیدا کنید. سپس با کمک ArraySuffix ایندکس وقوع الگو در رشته را مشخص کنید.



ایندکس‌های وقوع الگو در متن را با توجه به SuffixArray مشخص کنید:

۱ و ۴

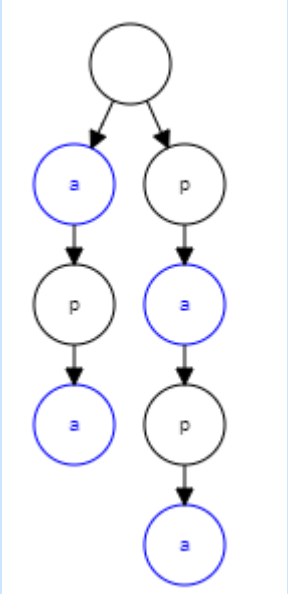
۳. [۱۰نمره] همانطور که در درس دیدید طبق الگوریتم گفته شده، تشکیل تبدیل BW برای یک رشته به طول n با پیچیدگی حافظه $O(n*n)$ امکان پذیر بود. الگوریتمی طراحی کنید که با پیچیدگی حافظه کمتری تبدیل BW برای یک رشته را بدست آورد.

با الگو برداری از نحوه ساخت رشته به کمک تبدیل BW در ابتدا اندیس هر کاراکتر را در کنار خود کاراکتر ذخیره می‌کنیم. سپس کاراکترها را مرتب کرده در یک لیست قرار می‌دهیم. از ابتدای لیست اندیس هر کاراکتر را بازیابی کرده و کاراکتر موجود در مقدار اندیس منهای یک را در خروجی قرار می‌دهیم. خروجی تبدیل BW است.

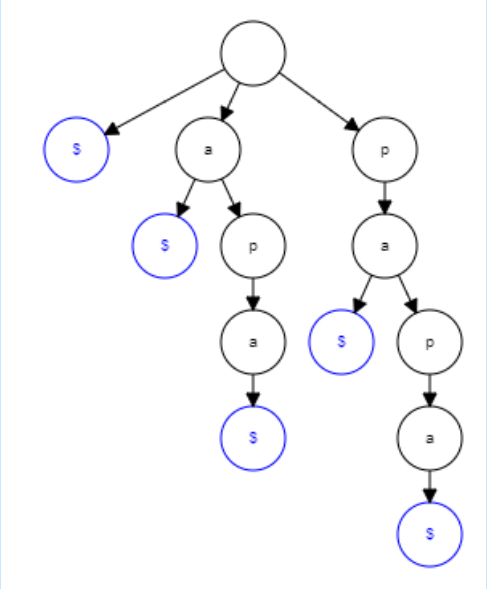
۴. [۱۸نمره] لزوم استفاده از نماد $:

(آ) [۹نمره] در صورتی که برای ساختن Suffix Tree متناظر با یک رشته در انتهای آن نماد $ را اضافه نکنیم، آیا در روند ایجاد درخت مشکلی پیش می آید؟ مثال بزنید.

مثال درخت پسوندی برای رشته papa یکبار بدون $ و یکبار با $ رسم شود. مشاهده می شود برای رشته هایی که یک زیررشته از خودشان درونشان هست به مشکل می خوریم و مثلا برای تطبیق رشته بدون ذخیره اطلاعات بیشتر در کمک درخت نمی توانیم این کار را به درستی انجام دهیم. همچنین روش‌های تولید SuffixArray هم نیاز به بازبینی پیدا می‌کند.



شکل ۳: بدون اضافه کردن نماد ($)



شکل ۲: با اضافه کردن نماد ($)

(ب) [۹نمره] در صورتی که در الگوریتم KMP نماد $ به رشته اضافه نشود چطور؟ چه مشکلی در روند الگوریتم ایجاد می شود؟ مثال بزنید.

به عنوان مثال می خواهیم مکان وقوع الگوی ab را در رشته abab بیابیم. اگر در بین الگو و رشته اقدام به گذاشتن $ نکنیم و آرایه border ها را تشکیل دهیم اعداد به صورت زیر خواهد بود:

$P+T$	a	b	a	b	a	b
π	0	0	1	2	3	4

که به کمک این الگوریتم نمی تواند محل وقوع دوم را تشخیص دهد. در واقع اگر الگوی مد نظر ما در رشته پشت سر هم تکرار شده باشند نمی تواند تکرارهای دوم به بعد را تشخیص دهد.

با تشکر ویژه از خانم مهسا سادات رضوی و آقای امیر خاکپور که در طراحی و آماده سازی سوال‌ها و پاسخنامه و همچنین برگزاری امتحان نقش بسزایی داشتند.