Problem 1. Write the suffix array for the string ACACIA$ and compute its Burrows-Wheeler transform.

1 ACACIA$	7	$	
2 CACIA$	6	A$	
3 ACIA$	1	ACACIA$	
4 CIA$	3	ACIA$	→ [7,6,1,3,4,2,5]
5 IA$	4	CIA$	
6 A$	2	CACIA$	
7 $	5	IA$	

BWT: AI$CAAC

Problem 2. Show the steps in the inverse Burrows-Wheeler transform for the string N$RSOOCIMPSE.

Problem 2. Show the steps in the inverse Burrows-Wheeler transform for the string N$RSOOCIMPSE.

Sort:	Concat 1st column and BWT and sort	Sort:	Concat 2ND column and BWT and sort	Sort:	Concat 3rd column and BWT	Sort:
$	N$	$C	N$C	$CO	N$CO	$COM
C	$C	CO	$CO	COM	$COM	COMP
E	RE	ES	RES	ESS	RESS	ESSI
I	SI	IO	SIO	ION	SION	ION$
M	OM	MP	OMP	MPR	OMPR	MPRE
N	ON	N$	ON$	N$C	ON$C	N$CO
O	CO	OM	COM	OMP	COMP	OMPR
O	IO	ON	ION	ON$	ION$	ON$C
P	MP	PR	MPR	PRE	MPRE	PRES
R	PR	RE	PRE	RES	PRES	RESS
S	SS	SI	SSI	SIO	SSIO	SION
S	ES	SS	ESS	SSI	ESSI	SSIO

2) N$RSOOCIMPSE

First col is sorted.

$COMPRESSION

Problem 3. Write the suffix array for the string GATTACA$ and compute its Burrows-Wheeler transform.

GATTACA$
12345678

(8,1)	$		$GATTACA	
(7,2)	A$		A$GATTAC	
(5,4)	ACA$		ACA$GATT	
(2,7)	ATTACA$	→	ATTACA$G	→ ACTGA$TA
(6,3)	CA$		CA$GATTA	
(1,8)	GATTACA$		GATTACA$	
(4,5)	TACA$		TACA$GAT	
(3,6)	TTACA$		TTACA$GA	

[8,7,5,2,6,1,4,3]

Problem 4. Show the steps in the inverse Burrows-Wheeler transform for the string TWOIPPR$ENO.

		occ
1	T	0
2	W	0
3	O	0
4	I	0
5	P	0
6	P	1
7	R	0
8	$	0
9	E	0
10	N	0
11	O	1

rank tells where the block of sorted begins

rank

$	E	I	N	O	P	R	T	W
1	2	3	4	5	7	9	10	11

count

$	E	I	N	O	P	R	T	W
1	1	1	1	2	2	1	1	1

POWERPOINT$

Problem 5. Consider the string woolloomooloo$. The BWT of this string is ooolooooolwml$. Show the steps taken by the pattern matching algorithm for the following patterns:

(a) olo

(b) oll

(c) oo

(d) wol

1	$	O1
2	L1	O2
3	L2	O3
4	L3	L1
5	M	O4
6	O1	O5
7	O2	O6
8	O3	O7
9	O4	O8
10	O5	L2
11	O6	W
12	O7	M
13	O8	L3
14	W	$

1	$	O1
2	L1	O2
3	L2	O3
4	L3	L1
5	M	O4
6	O1	O5
7	O2	O6
8	O3	O7
9	O4	O8
10	O5	L2
11	O6	W
12	O7	M
13	O8	L3
14	W	$

WOOLLOOMO O L O O $
123456789 10 11 12 13 14

Step 1. find first and last occurrence of letter 6 and 13 in this case.
Step 2. Find instances of L within that range
(instances of L2, L3)

1	$	O1
2	L1	O2
3	L2	O3
4	L3	L1
5	M	O4
6	O1	O5
7	O2	O6
8	O3	O7
9	O4	O8
10	O5	L2
11	O6	W
12	O7	M
13	O8	L3
14	W	$

Problem 8. Consider the following rank array, obtained during the prefix doubling algorithm applied to the strings ABBAABABBA$. At this point in the algorithm, we have just finished updating the ranks for some value of k.

String	A	B	B	A	A	B	A	B	B	A	$
Suffix ID	1	2	3	4	5	6	7	8	9	10	11
Rank	4	6	5	3	4	5	4	6	5	2	1

(a) Determine what the current value of k is. In other words, what lengths of strings are the ranks currently comparing?

suffix id	rank											
1	4	A	B	B	A	A	B	A	B	B	A	$
2	6	B	B	A	A	B	A	B	B	A	$	
3	5	B	A	A	B	A	B	B	A	$		
4	3	A	A	B	A	B	B	A	$			
5	4	A	B	A	B	B	A	$				
6	5	B	A	B	B	A	$					
7	4	A	B	B	A	$						
8	6	B	B	A	$							
9	5	B	A	$								
10	2	A	$									
11	1	$										

(b) Describe in detail how the prefix doubling algorithm would compare the following pairs of suffixes during the next iteration (for strings of length 2k):

- 1 and 2
- 1 and 5
- 2 and 8
- 3 and 9
- 1 and 7

k tells that everything with the first k characters has the same rank.

3 and 9
- rank(3) = 5, rank(9) = 5
- ranks are same, so tiebreak
- rank(3+2) = 4, rank(9+2) = 1
- 9 will go first

1 and 7
- rank(1) = 4, rank(7) = 4
- tie break
- rank(1 + 2) =5, rank(7 + 2) = 5
- tie break
- They will be given the same rank (for now)

Generally we compare the ranks
1. The ranks are the same in which we need to tiebreak
2. The ranks are different we know their ordering

if k=1:
- K cannot be 1 because things with the same 1st character have different ranks

if k=2:
- Yes, k=2 there are problems here

if k=4:
- Counterexample: rank(2)=6 and rank(8)=6 but they have different 4th characters therefore k cannot 4.

1 and 2
- rank(1)=4 and rank(2)=6
- We know that 1 < 2.

1 and 5
- rank(1)=4 and rank(5)=4
- Tiebreak - we can compare based on rank(i+k)
- rank(1+2)=5 and rank(5+2)=4
- So we know that suffix 5 should come first

Tiebreaking is the equivalent of comparing on first 2k (4) characters.

2 and 8
- rank(2) = 6, rank(8) = 6
- Ranks are the same, go to tie break
- rank(2 + 2) = 3, rank(8 + 2) = 2
- 8 will go first

Problem 6. Consider the prefix doubling algorithm applied to computing the suffix array of JARARAKA$. Write the partially-sorted suffix array after the length two prefixes have been sorted. Write the corresponding rank array and perform the next iteration of prefix doubling, showing the partially-sorted suffix array for the length four prefixes.

index	suffix	rank
1	JARARAKA$	5
2	ARARAKA$	5
3	RARAKA$	9
4	ARAKA$	4
5	RAKA$	8
6	AKA$	3
7	KA$	7
8	A$	2
9	$	1

rank index	new ranks
9	1
8	2
6	3
4	4
2	5
1	6
7	7
5	8
3	9

JARARAKA$
123456789

4 and 2 - Tiebreak
- rank(4+2)=3 and rank(2+2)=4
- we know that suffix 4 should be first

5 and 3 - Tiebreak
- rank(5+2)=6 and rank(3+2)=7
- suffix 5 comes first

Problem 7. The *minimum lexicographical rotation* problem is the problem of finding, for a given string, its cyclic rotation that is lexicographically least. For example, given the string banana, its cyclic rotations are banana, ananab, nanaba, anaban, nabana, abanan. The lexicographically (alphabetically) least one is abanan. Describe how to solve the minimum lexicographical rotation problem using a suffix array.

ABCA

ABCA
BCAA
CAAB
AABC

AABC → A$ABC
ABCA → ABCA$
BCAA → BCA$A
CAAB → CA$AB

$

ZABAZABA$

~~$~~
~~A$~~
~~ABA$~~
ABAZABA$
AZABA$
BA$
BAZABA$
ZABA$
ZABAZABA$

ABAZ
AZAB
BAZA
ZABA

$
A$
ABA$
BA$
ZABA$

$
A$ZAB
ABA$Z
BA$ZA
ZABA$

The suffix array does not consider any of the characters after the $
How can make the suffix array consider the characters after the $, or consider the characters the wrap around in the cyclic rotation.

i	BWT	Occ
1	N	
2	R	
3	A	
4	C	
5	O	
6	O	
7	C	
9	$	

Rank

$	A	C	N	O	R

count

$	A	C	N	O	R