Let M = banaana

0.a) Append sentinel character

M = banaana\0

1.) a.) Append three zeros at end

M = banaana\0 000

0 1 2 3 4 5 6 7 8 9 10 ← indices

b) Find R_{12}_idx

R_{12}_idx ⇒ [1 2 4 5 7 8]

→ [1 4 7 2 5 8]

c) Convert to string triplets and sort

R_{12}_str → [ana ana 000 naa naO 000]

→ [000 000 ana ana naO naa]

X Since there are duplicates, we do find its recursive suffix array, which will be the R_{12} order

d.) Group into buckets

R_{12}_str → [000 000 ana ana naO naa]

start at 1!!! → 1 2 3 4 ← bucket index

e.) Form recursive message $M^{(1)}$ from R_{12}_idx

R_{12}_idx → [1 4 7 2 5 8]

$M^{(1)}$ → [2 2 1 4 3 1] ← convert these into their corresponding bucket indices

2.) a.) Append three zeros at end

$M^{(1)}$ → 2 2 1 4 3 1 0 0 0

0 1 2 3 4 5 6 7 8

b.) Find R_{12}_$idx^{(1)}$

R_{12}_$idx^{(1)}$ → [1 2 4 5]

→ [1 4 2 5]

c) Convert to string triplets and sort

R_{12}_$str^{(1)}$ → [214 310 143 100]

→ [100 143 214 310] ← $R_{12}^{(1)}$

X Since there are no duplicates, we can directly get $R_{12}^{(1)}$

d.) Find R_{12}

$R_{12}^{(1)}$ → [5 2 1 4]

e) Find $R_0^{(1)}$_idx

$R_0^{(1)}$_idx → [0 3 6]

f.) Convert to 2-ary tuples and sort

$R_0^{(1)}$_str ⇒ [20 43 06]

→ [06 20 43]

value in $M^{(1)}$ ↑ ↑ index in $M^{(1)}$

g.) Get second elements as $R_0^{(1)}$

$R_0^{(1)}$ → [6 0 3]

h) Merge $R_0^{(1)}$ and $R_{12}^{(1)}$

$R_0^{(1)}$ $R_{12}^{(1)}$

[6 0 3] [5 2 1 4]

↑ ignore this ↑

5 i) [6 0 3] [5 2 1 4] X $M^{(1)}$ in R_0 is smaller, so push this value to the suffix array

$M^{(1)}$→2 $M^{(1)}$→1

2 ii) [6 0 3] [5 2 1 4] X $M^{(1)}$ for both indices are the same!

$\hat{M}^{(1)}$→2 $\hat{M}^{(1)}$→1 X R_{12} mod 3 is 1, so compare relative orders in R_{12}

1 iii) [6 0 3] [5 2 1 4] X pos of R_{12} is smaller, so push this value to the suffix array

$\hat{M}^{(1)}$→2 $\hat{M}^{(1)}$→2

next idx = 1 next idx = 2

pos in R_{12} → 2 pos in R_{12} = 1

0 iv.) [6 0 3] [5 2 1 4]

$\hat{M}^{(1)}$→2 $\hat{M}^{(1)}$→3

4 v.) [6 0 3] [5 2 1 4]

$\hat{M}^{(1)}$→4 $\hat{M}^{(1)}$→3

3 vi) [6 0 3] [5 2 1 4]

push remaining elements at end of suffix array ↑ no more elements!

i.) Collate suffix array

$SA^{(1)}$ → [5 2 1 0 4 3]

3.) a.) Convert $SA^{(1)}$ back to R_{12} using R_{12}_idx

$SA^{(1)}$ → [5 2 1 0 4 3] ← treat these as positions in R_{12}_idx

R_{12}_idx → [1 4 7 2 5 8]

0 1 2 3 4 5

R_{12} → [8 7 4 1 5 2]

0 1 2 3 4 5

b) Find R_0_idx

R_0_idx → [0 3 6]

c) Convert to 2-ary tuples and sort

R_0_str → [b0 a3 a6]

→ [a3 a6 b0]

X Since these are duplicate first elements, we have to convert these duplicates into their next elements

R_0_str → [a3 a6 b0]

sort these

R_0_srtd → [93 a6] → [43 76]

replace with second element plus one

→ [43 76] → [23 16]

replace with position in R_{12} plus one

→ [23 16]

sort and merge with original R_{12}_str

→ [1 6 23]

R_0_str → [16 23 b0]

d) Get second elements as R_0

R_0 → [6 3 0]

4) Merge R_0 and R_{12}

M = banaana\0 000
0 1 2 3 4 5 6 7 8 9 10

	R_0	R_{12}	
	[6 3 0]	[8 7 4 1 5 2]	
	^	^ ignore this	
7	[6 3 0]	[8 7 4 1 5 2]	· X · M in R_{12} is smaller, so push this value to the suffix array
	^ M→a	^ M→0	
6	[6 3 0]	[8 7 4 1 5 2]	· X · M for both values are the same!
	^ M→a next idx→7 M→0	^ M→a next idx→5 M→n	· X · R_{12} mod 3 is 2, so compare M of next indices
			· X · M of R_0 is smaller, so push this value to the suffix array
3	[6 3 0]	[8 7 4 1 5 2]	· X · M for both values are the same!
	^ M→a next idx→4 M→a	^ M→a next idx→5 M→n	· X · R_{12} mod 3 is 2, so compare M of next indices
			· X · M of R_0 is smaller, so push this value to the suffix array
4	[6 3 0]	[8 7 4 1 5 2]	
	^ M→b	^ M→a	
1	[6 3 0]	[8 7 4] [5 2]	
	^ M→b	^ a	
0	[6 3 0]	[8 7 4 1 5 2]	
	^ M→b	^ M→n	
5	[6 3 0]	[8 7 4 1 5 2]	
2	^ no more elements!	^ push remaining elements at end of suffix array	

5) Collate suffix array

SA → [7 6 3 4 1 0 5 2]

6) Map suffix array against M to get the BWT

SA → [7 6 3 4 1 0 5 2]
6 5 2 3 0 -1 4 1 ← these are indices in M

M → banaana\0
0 1 2 3 4 5 6 7

BWT(M) → annab\0aa