Radix and Suffix Sorting

- Radix Sort
- Suffix Sort

Philip Bille

Radix and Suffix Sorting

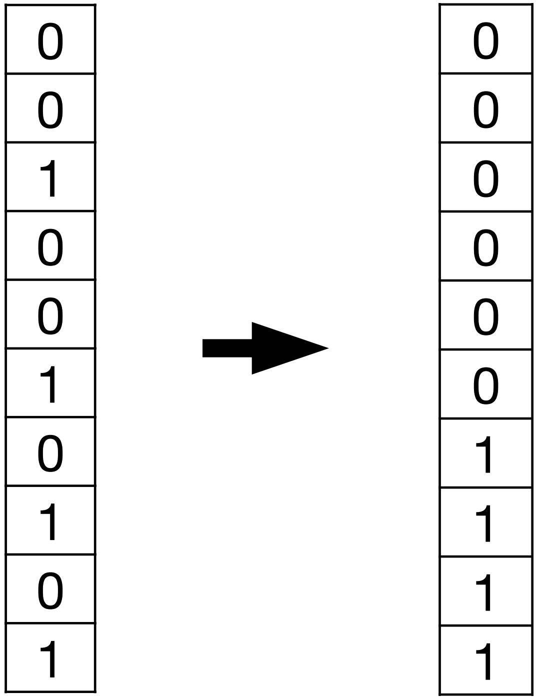
- Radix Sort
- Suffix Sort

Radix Sort

- Sorting small universes. Given a sequence of n integers from a universe U = {0, 1, ..., u-1}.
- How fast can we sort sequence if the size of the universe is not too big?

Radix Sort

n = 10, U = {0,1}

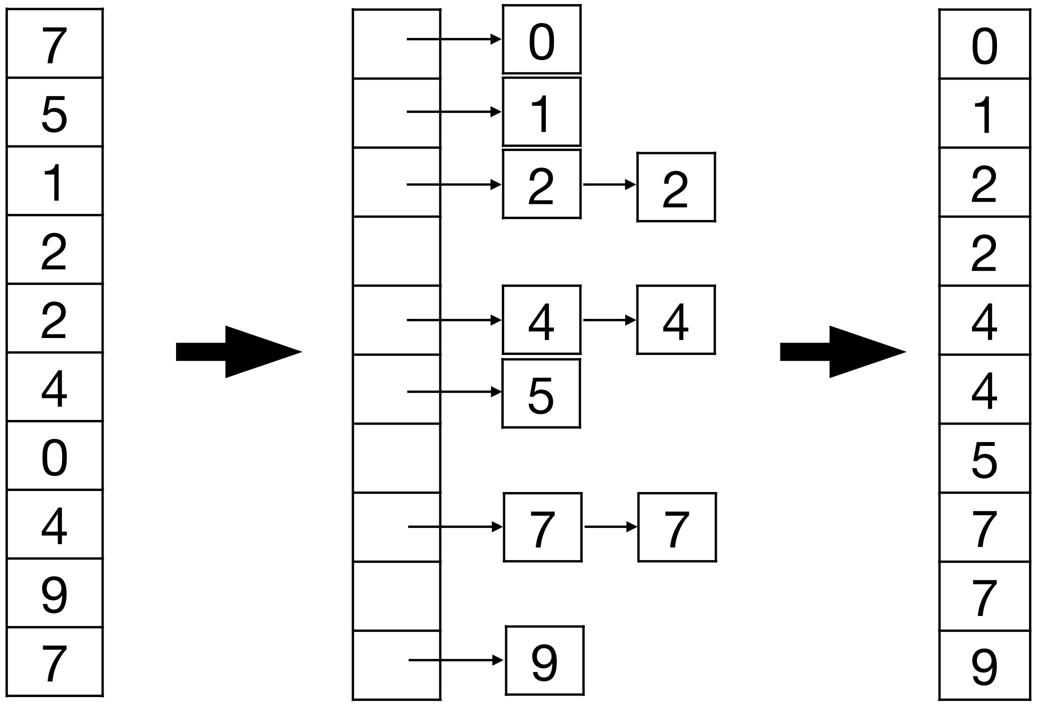


0	0
0	0
1	0
0	0
0	0
1	0
0	1
1	1
0	1
1	1

- Algorithm. Count 0s and 1s.
- Time. O(n).

Radix Sort

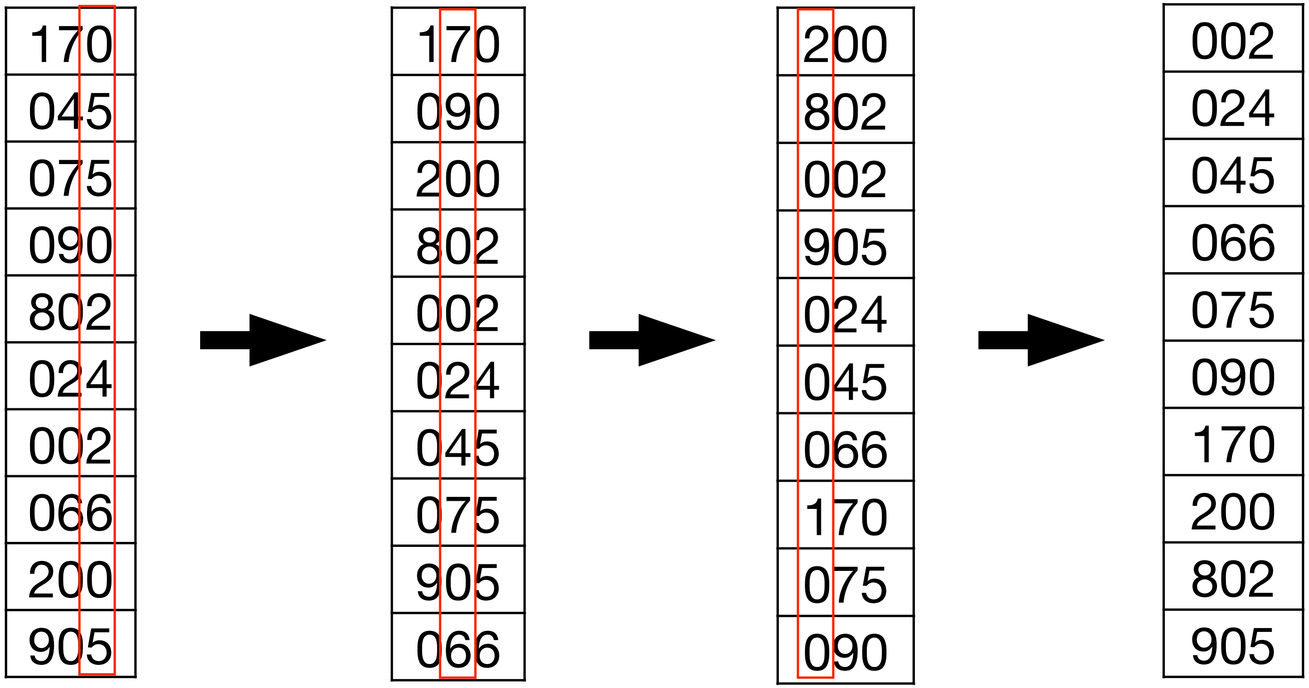
n = 10, U = {0,1, ..., n-1 = 9}



- Algorithm. Insert into array of linked list + traverse array of linked list.
- Time. O(n + u) = O(n)
- Sorting can be stable.

Radix Sort

n = 10, U = {0, ..., n^3 - 1 = 999}



170	170	200	002
045	090	802	024
075	200	002	045
090	802	905	066
802	002	024	075
024	024	045	090
002	045	066	170
066	075	170	200
200	905	075	802
905	066	090	905

- Radix Sort. Sort on each digit from right to left using stable sort.
- Time. O(n + n + n) = O(n)

Radix Sort

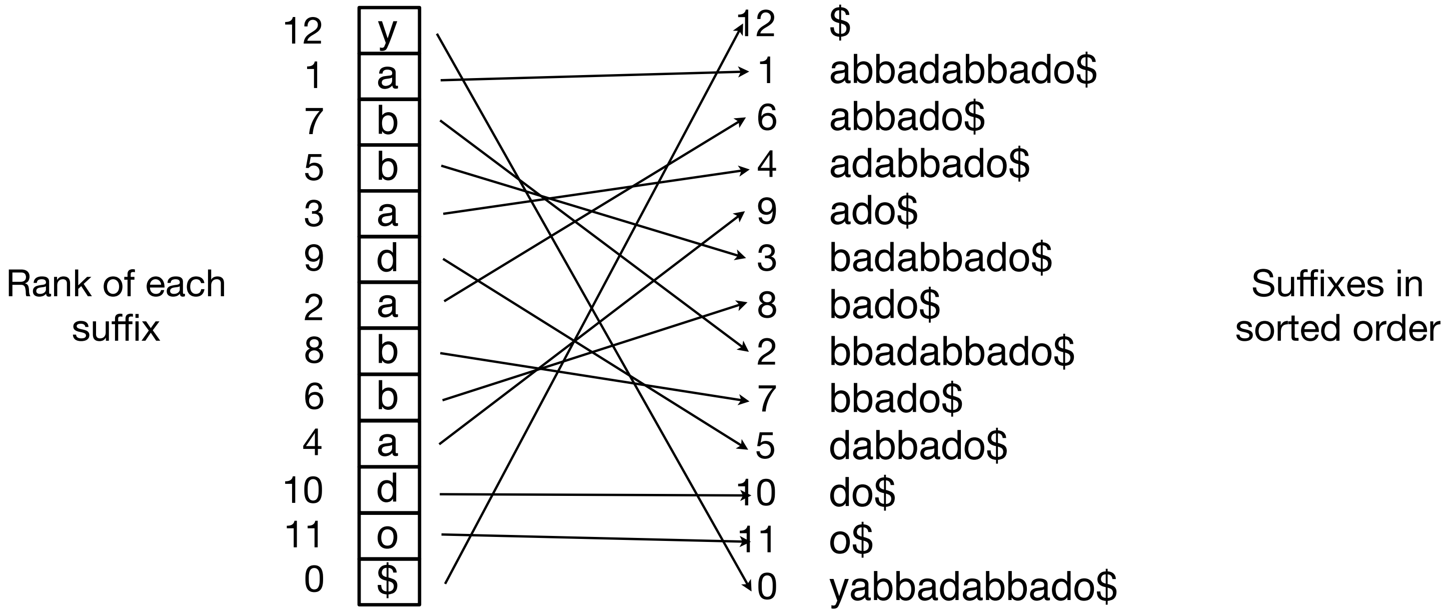
- Radix Sort [Hollerith 1887]. Sort sequence of n integers from $U = \{0, ..., n^k-1\}$.
  - Write each element in sequence as a base n integer $x = (x_1, x_2, .., x_k)$
  - Sort sequence according to each digit from right to left. Sorting should be stable.
- Time. $O(nk)$

Radix and Suffix Sorting

- Radix Sort
- Suffix Sort

Suffix Sort

- Suffix sorting. Given string S of length n over alphabet Σ, compute the sorted lexicographic order of all suffixes of S.



Rank of each suffix

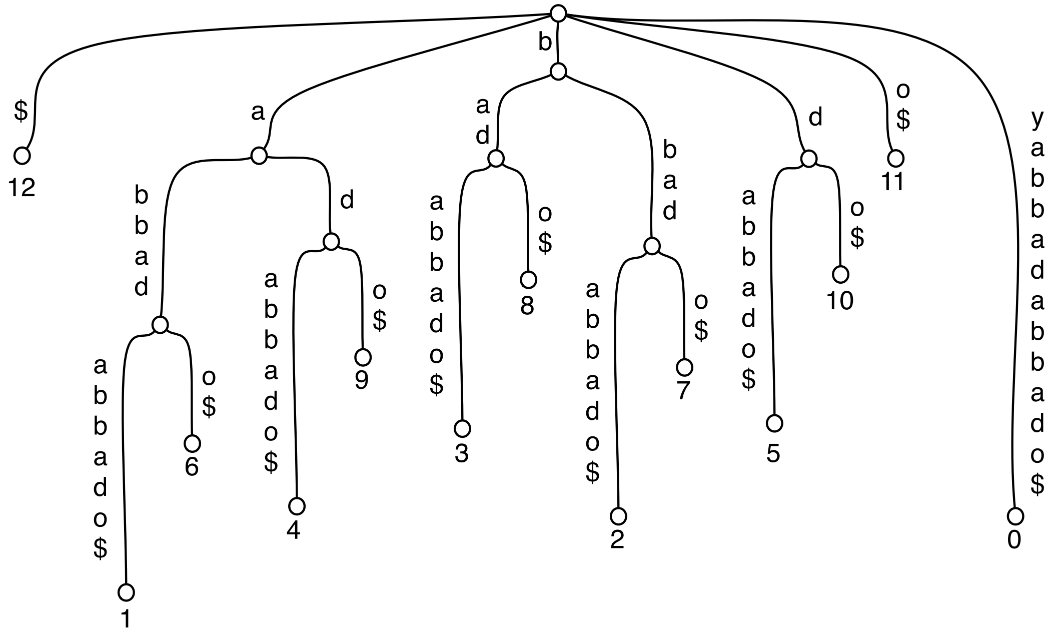
12	y	12	$
1	a	1	abbadabbado$
7	b	6	abbado$
5	b	4	adabbado$
3	a	9	ado$
9	d	3	badabbado$
2	a	8	bado$
8	b	2	bbadabbado$
6	b	7	bbado$
4	a	5	dabbado$
10	d	10	do$
11	o	11	o$
0	$	0	yabbadabbado$

Suffixes in sorted order

- Theorem [Kasai et al. 2001]. Given the sorted lexicographic order of suffixes of S, we can construct the suffix tree for S in linear time.

Suffix Sort

- Suffix trees and sorting. The lexicographic order of the suffixes is the same ordering as suffixes in the leaves of the suffix tree.
- Suffix array. The array of the sorted order of the suffixes.



12	1	6	4	9	3	8	2	7	5	10	11	0

Suffix Sort

- Goal. Compute the lexicographic order of all suffixes of S fast.
- For simplicity assume $|\Sigma| = O(n)$
- Solution in 3 steps.
  - Solution 1: Radix sorting
  - Solution 2: Prefix doubling
  - Solution 3: Difference cover sampling

Solution 1: Radix Sort

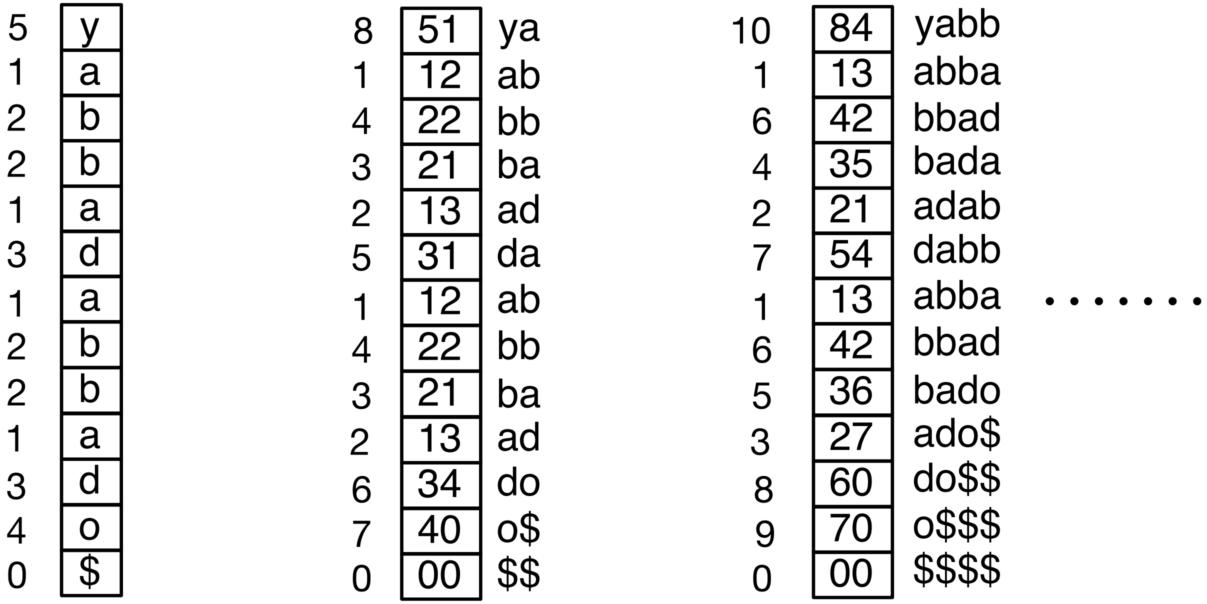
- Radix Sort.
  - Generate all suffixes (pad with $).
  - Radix sort.

```
yabbadabbado$
abbadabbado$$
bbadabbado$$$
badabbado$$$$
adabbado$$$$$
dabbado$$$$$$
abbado$$$$$$$
bbado$$$$$$$$
bado$$$$$$$$$
ado$$$$$$$$$$
do$$$$$$$$$$$
o$$$$$$$$$$$$
$$$$$$$$$$$$$
```

- Time. $O(n^2)$

Solution 2: Prefix Doubling

- Prefix doubling [Manber and Myers 1990]. Sort substrings (padded with $) of lengths 1, 2, 4, 8, ..., n. Each step uses radix sort on pair from previous step.



5	y	8	51	ya	10	84	yabb
1	a	1	12	ab	1	13	abba
2	b	4	22	bb	6	42	bbad
2	b	3	21	ba	4	35	bada
1	a	2	13	ad	2	21	adab
3	d	5	31	da	7	54	dabb
1	a	1	12	ab	1	13	abba
2	b	4	22	bb	6	42	bbad
2	b	3	21	ba	5	36	bado
1	a	2	13	ad	3	27	ado$
3	d	6	34	do	8	60	do$$
4	o	7	40	o$	9	70	o$$$
0	$	0	00	$$	0	00	$$$$

- Time. O(n log n)

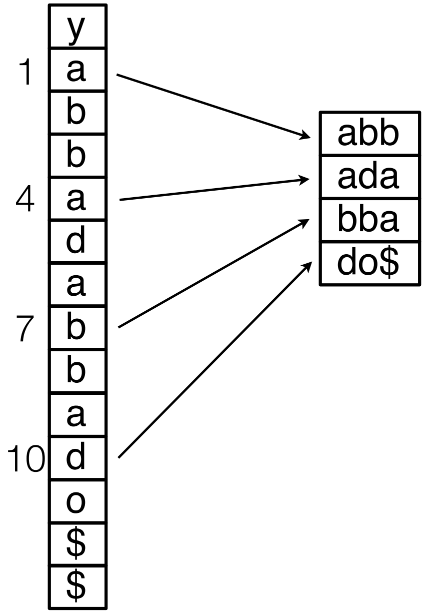
Solution 3: Difference Cover Sampling

- DC3 Algorithm [Karkkainen et al. 2003]. Sort suffixes in three steps:
  - Step 1. Sort sample suffixes.
    - Sample all suffixes starting at positions i = 1 mod 3 and i = 2 mod 3.
    - Recursively sort sample suffixes.
  - Step 2. Sort non-sample suffixes.
    - Sort the remaining suffixes (starting at positions i = 0 mod 3).
  - Step 3. Merge.
    - Merge sample and non-sample suffixes.

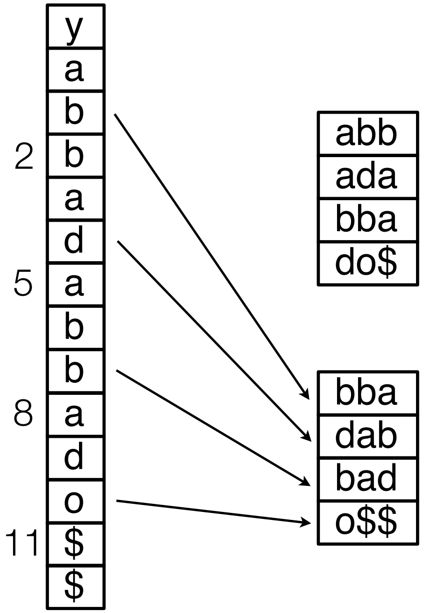
Step 1: Sort Sample Suffixes

y
a
b
b
a
d
a
b
b
a
d
o
$
$

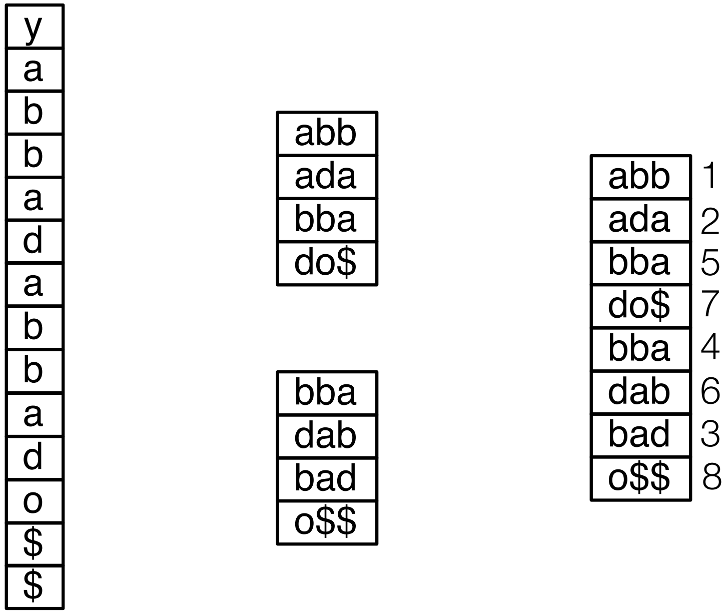
Step 1: Sort Sample Suffixes



Step 1: Sort Sample Suffixes



Step 1: Sort Sample Suffixes



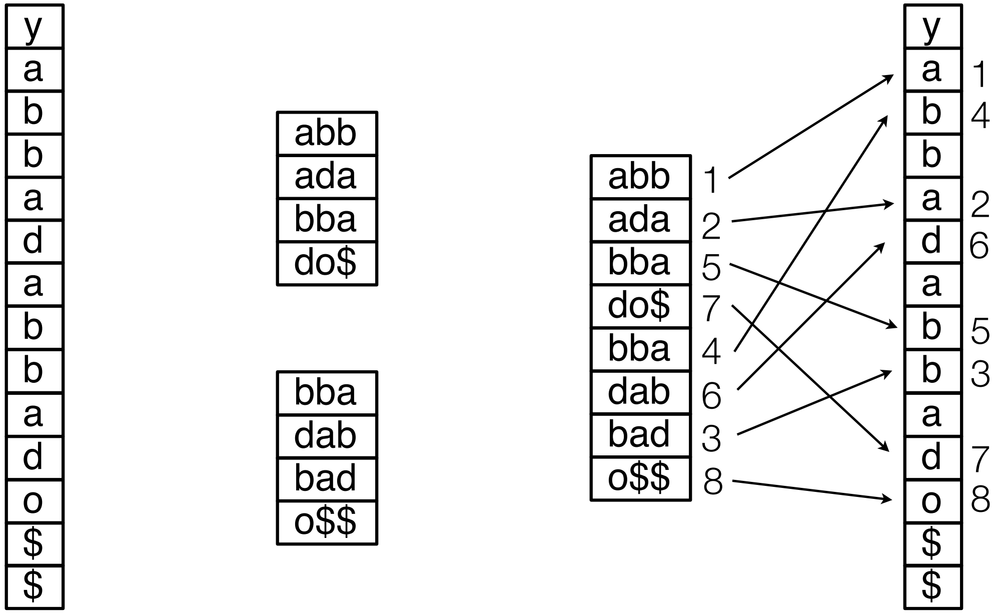
y
a
b
b
a
d
a
b
b
a
d
o
$
$

abb
ada
bba
do$

bba
dab
bad
o$$

abb	1
ada	2
bba	5
do$	7
bba	4
dab	6
bad	3
o$$	8

Step 1: Sort Sample Suffixes



y	a	b	b	a	d	a	b	b	a	d	o	$	$

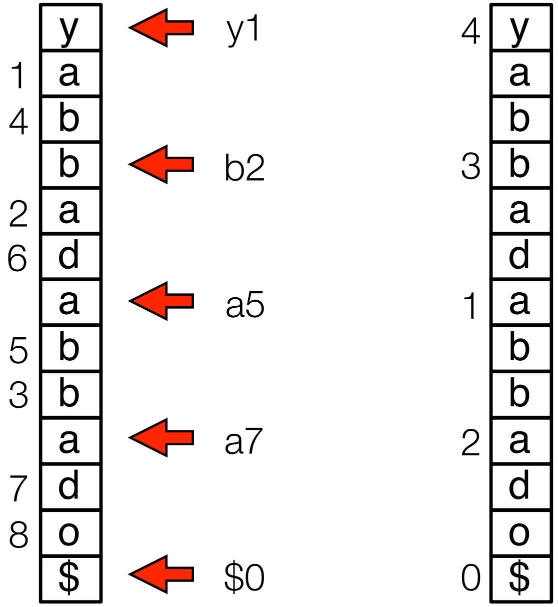
abb
ada
bba
do$

bba
dab
bad
o$$

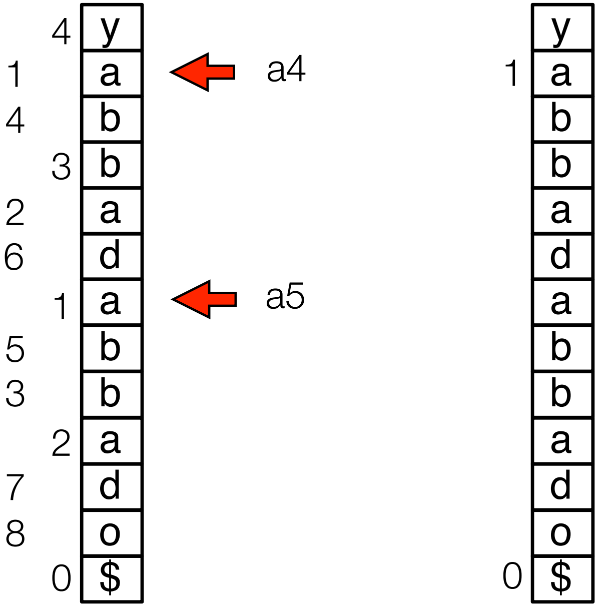
abb	1
ada	2
bba	5
do$	7
bba	4
dab	6
bad	3
o$$	8

y	
a	1
b	4
b	
a	2
d	6
a	
b	5
b	3
a	
d	7
o	8
$	
$	

Step 2: Sort Non-Sample Suffixes

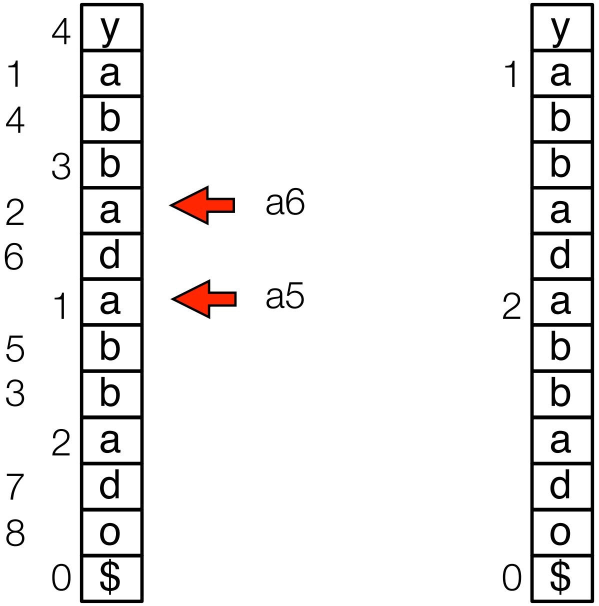


Step 3: Merge

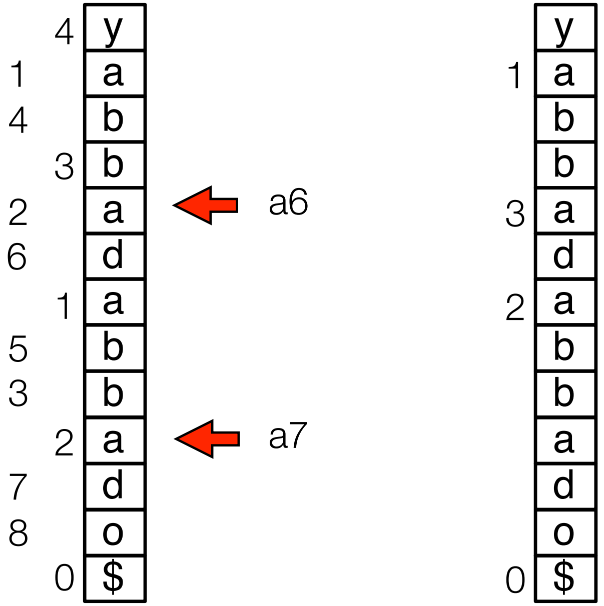


	4	y			y
1		a	a4	1	a
4		b			b
	3	b			b
2		a			a
6		d			d
	1	a	a5		a
5		b			b
3		b			b
	2	a			a
7		d			d
8		o			o
	0	$		0	$

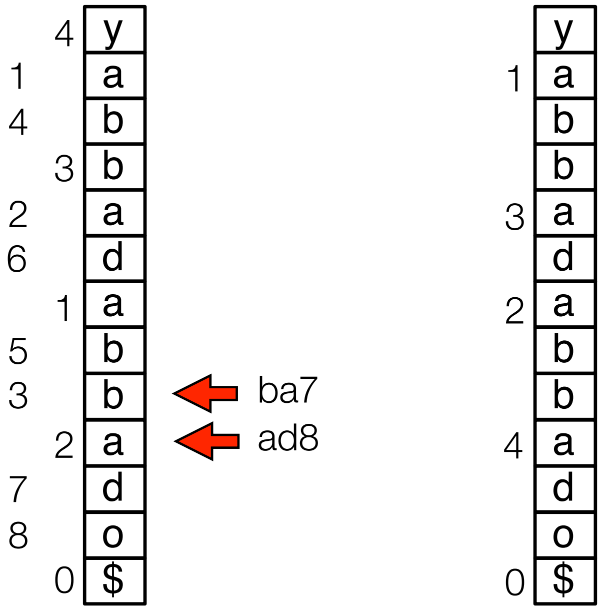
Step 3: Merge



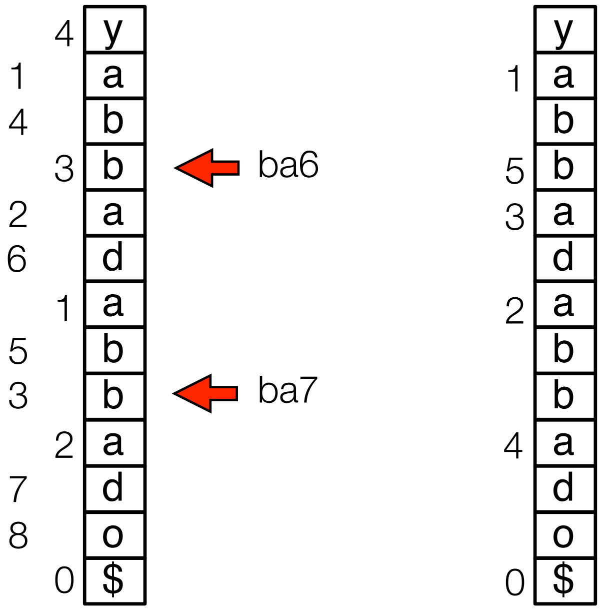
Step 3: Merge



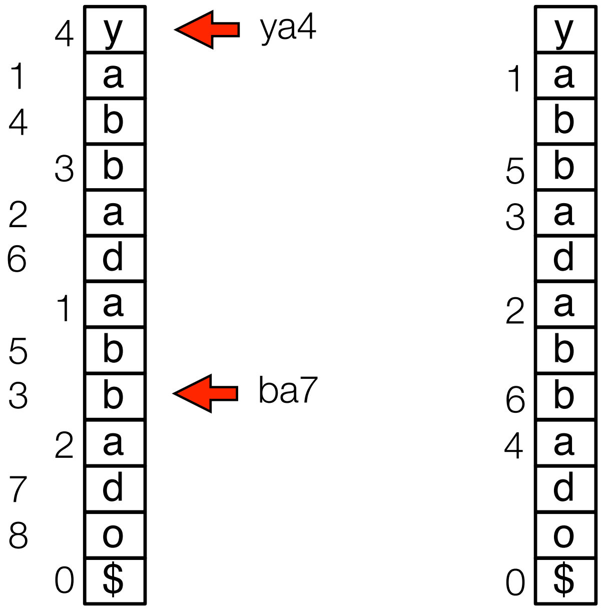
Step 3: Merge



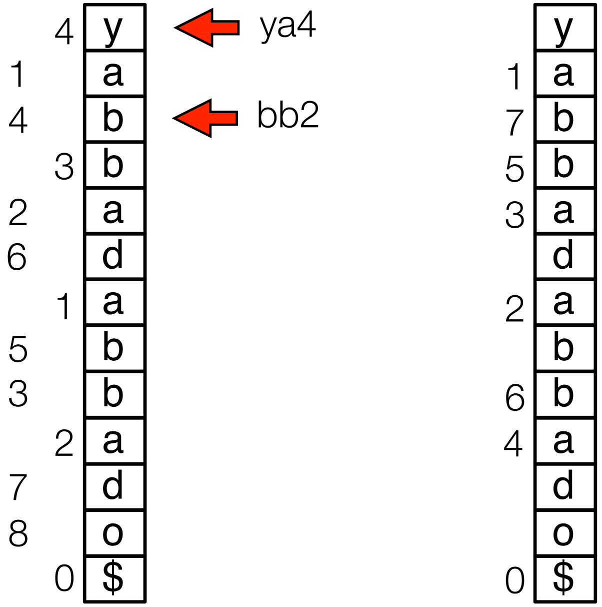
Step 3: Merge



Step 3: Merge

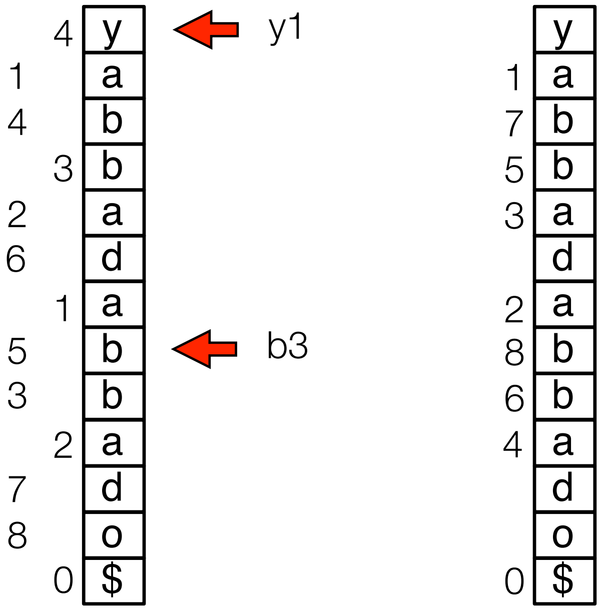

4 y
1 a
4 b
3 b
2 a
6 d
1 a
5 b
3 b
2 a
7 d
8 o
0 $
ya4
ba7
y
1 a
b
5 b
3 a
d
2 a
b
6 b
4 a
d
o
0 $

Step 3: Merge

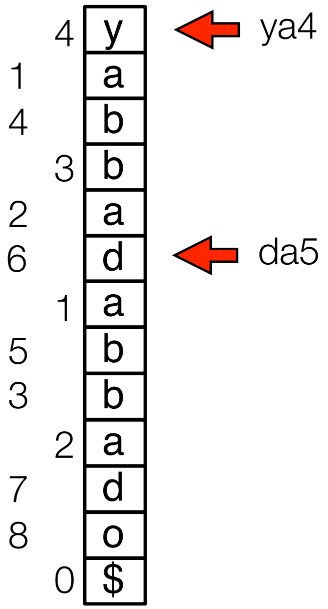


	4	y	ya4		y
1		a		1	a
4		b	bb2	7	b
	3	b		5	b
2		a		3	a
6		d			d
	1	a		2	a
5		b			b
3		b		6	b
	2	a		4	a
7		d			d
8		o			o
	0	$		0	$

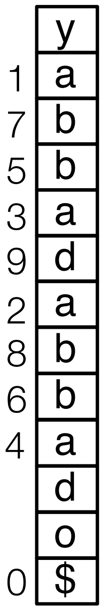
Step 3: Merge


4 y
1 a
4 b
3 b
2 a
6 d
1 a
5 b
3 b
2 a
7 d
8 o
0 $
y1
b3
y
1 a
7 b
5 b
3 a
d
2 a
8 b
6 b
4 a
d
o
0 $

Step 3: Merge

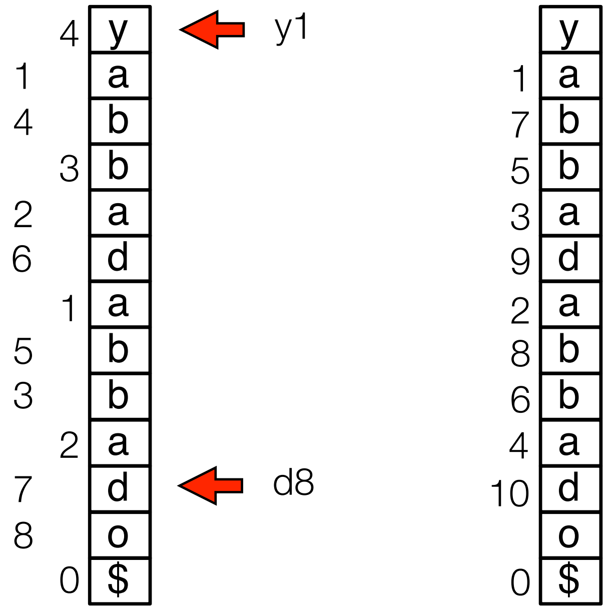


4 y
1 a
4 b
3 b
2 a
6 d
1 a
5 b
3 b
2 a
7 d
8 o
0 $
ya4
da5

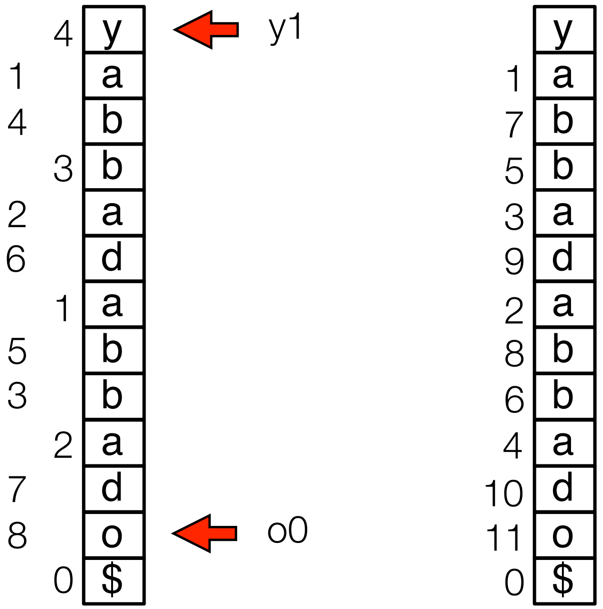


y
1 a
7 b
5 b
3 a
9 d
2 a
8 b
6 b
4 a
d
o
0 $

Step 3: Merge

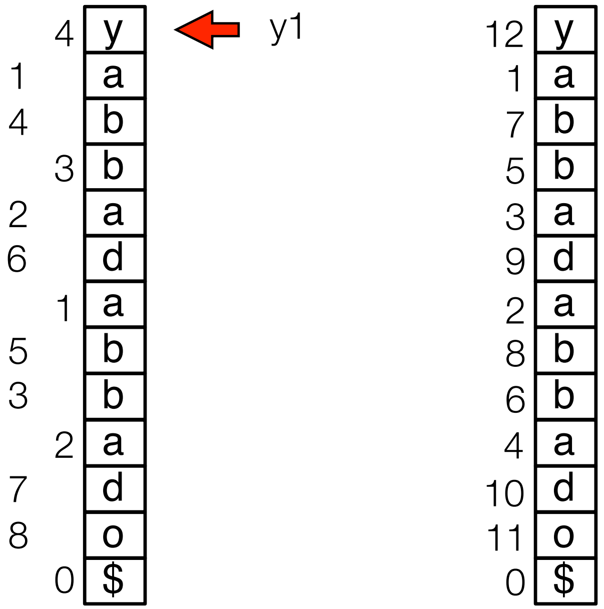

4 y
1 a
4 b
3 b
2 a
6 d
1 a
5 b
3 b
2 a
7 d
8 o
0 $
y1
d8
y
1 a
7 b
5 b
3 a
9 d
2 a
8 b
6 b
4 a
10 d
o
0 $

Step 3: Merge



4 y
1 a
4 b
3 b
2 a
6 d
1 a
5 b
3 b
2 a
7 d
8 o
0 $
y1
o0
y
1 a
7 b
5 b
3 a
9 d
2 a
8 b
6 b
4 a
10 d
11 o
0 $

Step 3: Merge



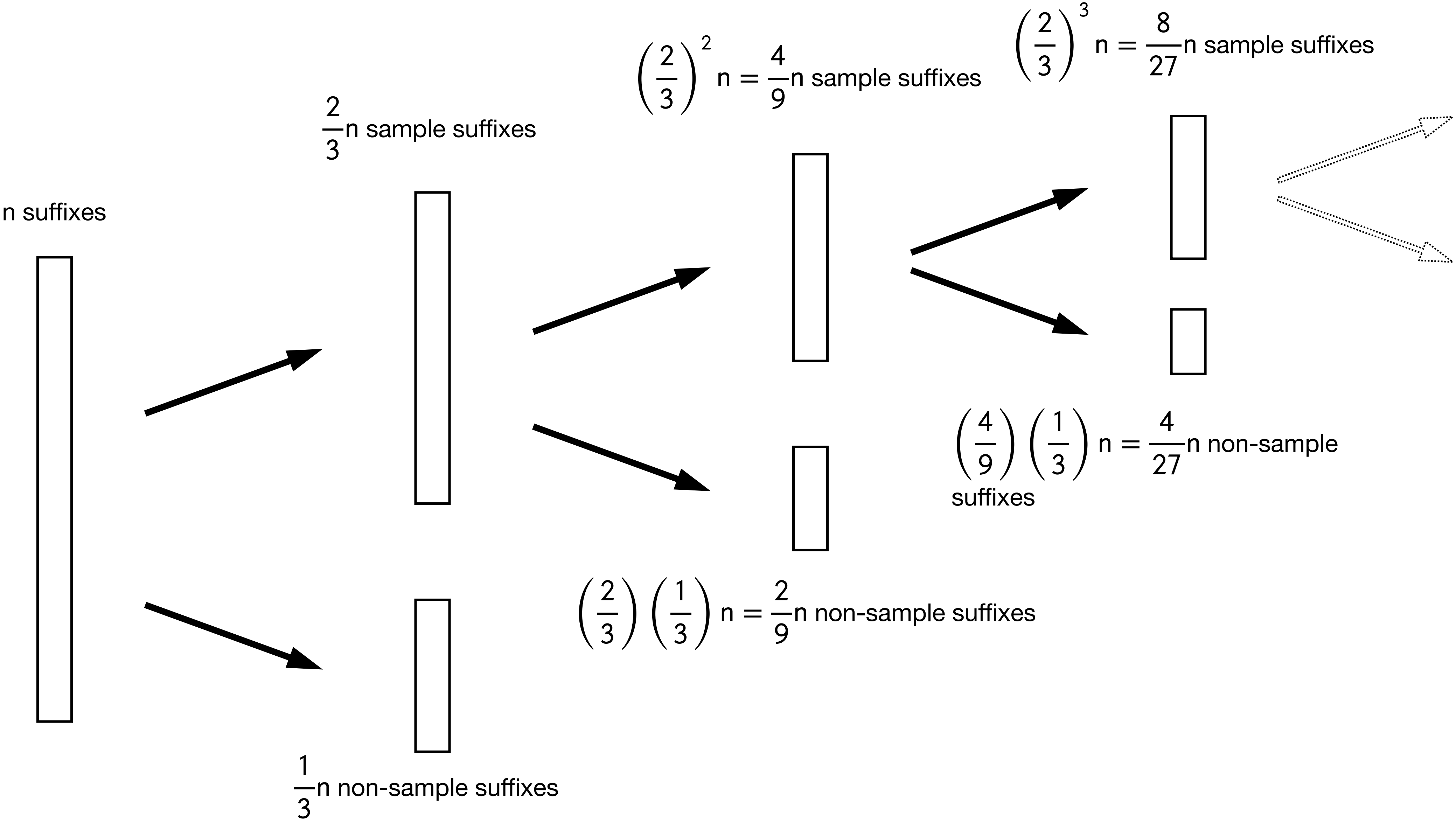
	4	y	y1
1		a	
4		b	
	3	b	
2		a	
6		d	
	1	a	
5		b	
3		b	
	2	a	
7		d	
8		o	
	0	$	

12	y
1	a
7	b
5	b
3	a
9	d
2	a
8	b
6	b
4	a
10	d
11	o
0	$

Solution 3: Difference Cover Sampling

- DC3 Algorithm. Sort suffixes in three steps:
  - Step 1. Sort sample suffixes.
    - Sample all suffixes starting at positions $i = 1 \bmod 3$ and $i = 2 \bmod 3$. $O(n)$
    - Recursively sort sample suffixes. $T(2n/3)$
  - Step 2. Sort non-sample suffixes.
    - Sort the remaining suffixes (starting at positions $i = 0 \bmod 3$). $O(n)$
  - Step 3. Merge.
    - Merge sample and non-sample suffixes. $O(n)$
- $T(n)$ = time to suffix sort a string of length n over alphabet of size n

- Time. $T(n) = T(2n/3) + O(n) = O(n)$


n suffixes
$\frac{2}{3}$n sample suffixes
$\frac{1}{3}$n non-sample suffixes
$\left(\frac{2}{3}\right)^2 n = \frac{4}{9}$n sample suffixes
$\left(\frac{2}{3}\right)\left(\frac{1}{3}\right) n = \frac{2}{9}$n non-sample suffixes
$\left(\frac{2}{3}\right)^3 n = \frac{8}{27}$n sample suffixes
$\left(\frac{4}{9}\right)\left(\frac{1}{3}\right) n = \frac{4}{27}$n non-sample suffixes

Solution 3: Difference Cover Sampling

- Theorem. We can suffix sort a string of length n over alphabet Σ of size n in time O(n).
- Theorem. We can suffix sort a string of length n over alphabet Σ O(sort(n, |Σ|)) time.

Radix and Suffix Sorting

- Radix Sort
- Suffix Sort