Radix and Suffix Sorting

- Radix Sort
- Suffix Sort

Philip Bille

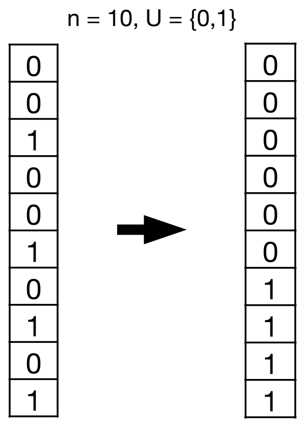
Radix and Suffix Sorting

- 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?

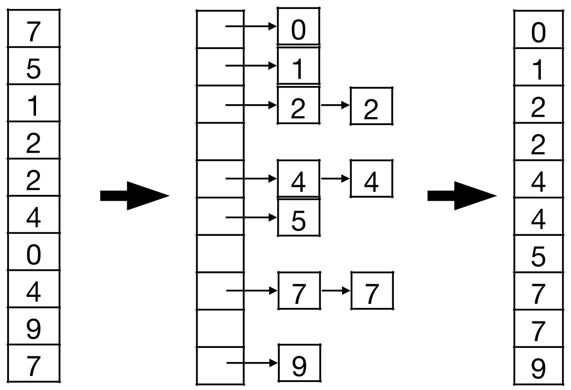
Radix Sort



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

Radix Sort

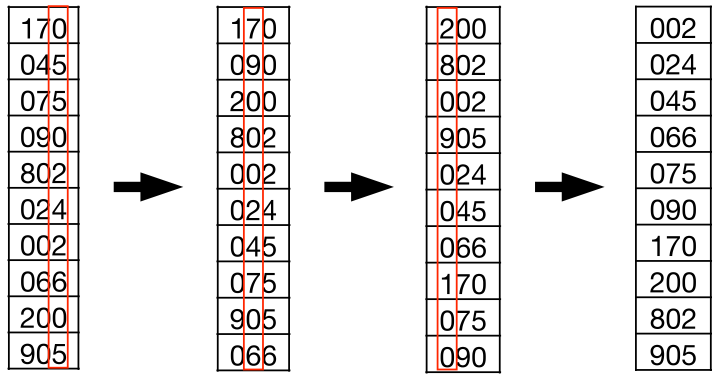
n = 10, U = {0,1, ..., n-1 = 9}



- 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}



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

Radix Sort

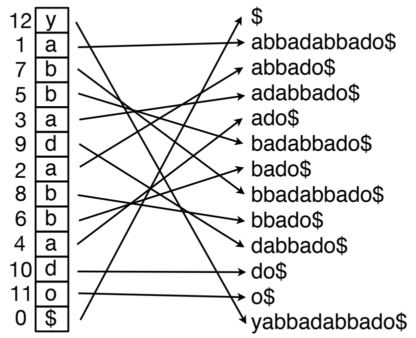
- Radix Sort [Hollerith 1887]. Sort sequence of n integers from U = {0, ..., n^k-1}.
  - Write each element in sequence as a base n integer x = (x_1, x_2, .., x_k)
  - Sort sequence according to each digit from right to left. Sorting should be stable.
- Time. O(nk)

Radix and Suffix Sorting

- Radix Sort
- Suffix Sort

Suffix Sort

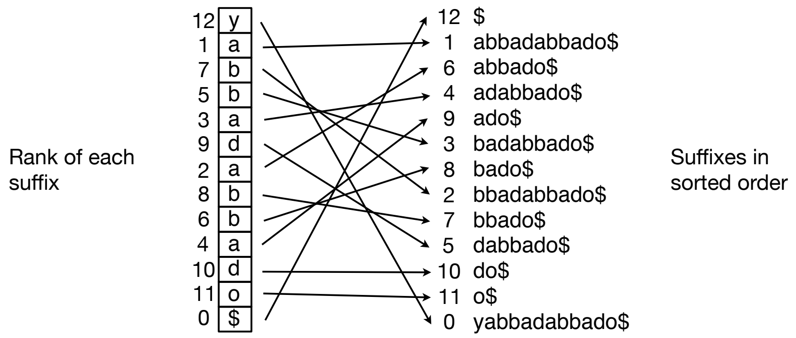
- Suffix sorting. Given string S of length n over alphabet Σ, compute the sorted lexicographic order of all suffixes of S.



- 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

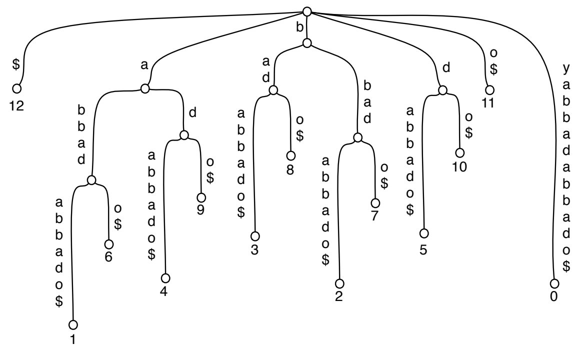
- Suffix sorting. Given string S of length n over alphabet Σ, compute the sorted lexicographic order of all suffixes of S.



- 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 |Σ| = 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. $O(n \log n)$

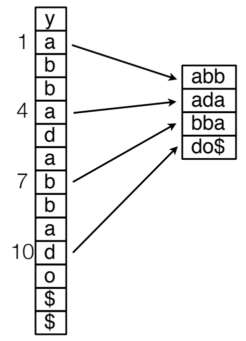
Solution 3: Difference Cover Sampling

- 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.

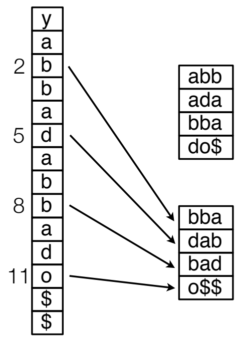
Step 1: Sort Sample Suffixes

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

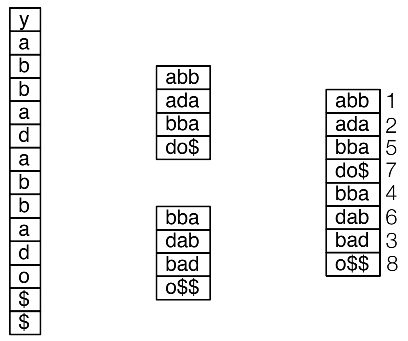
Step 1: Sort Sample Suffixes



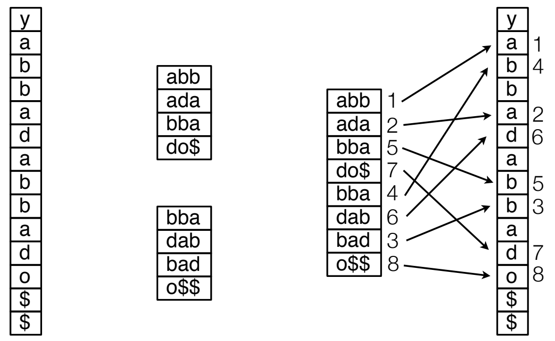
Step 1: Sort Sample Suffixes



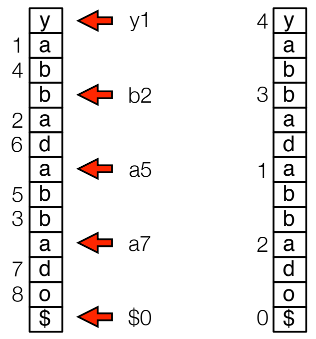
Step 1: Sort Sample Suffixes



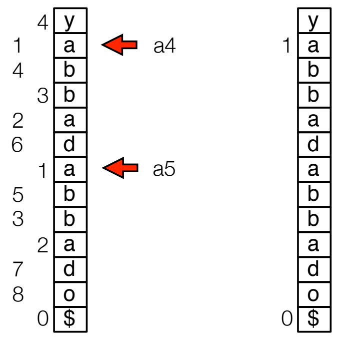
Step 1: Sort Sample Suffixes



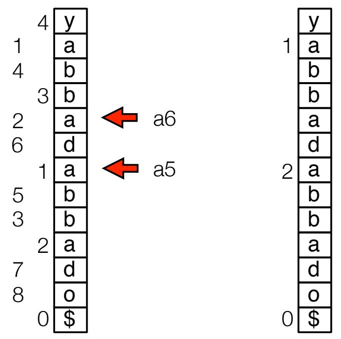
Step 2: Sort Non-Sample Suffixes



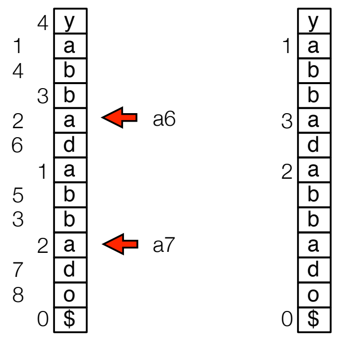
Step 3: Merge



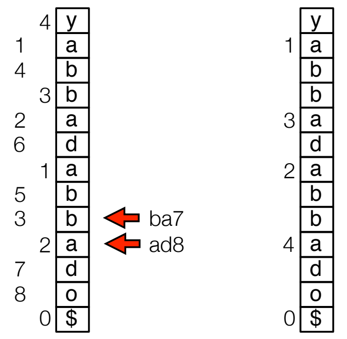
Step 3: Merge



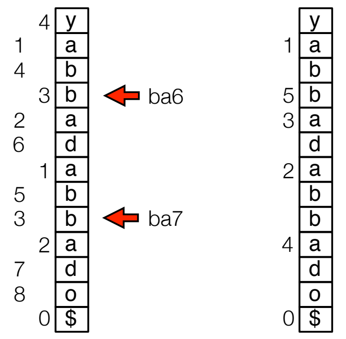
Step 3: Merge



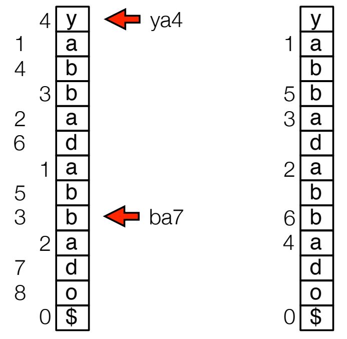
Step 3: Merge



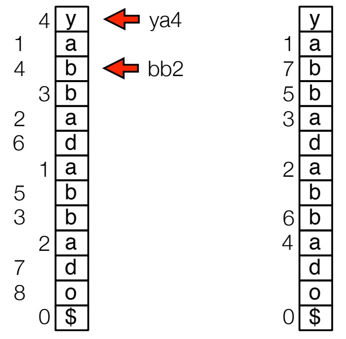
Step 3: Merge



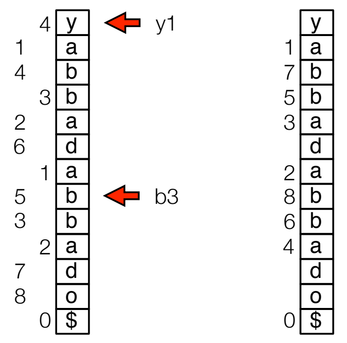
Step 3: Merge



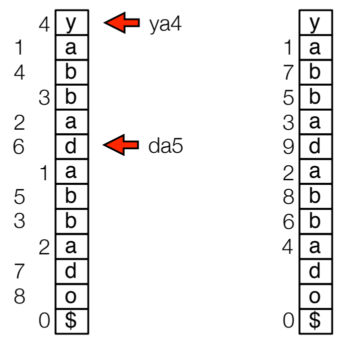
Step 3: Merge



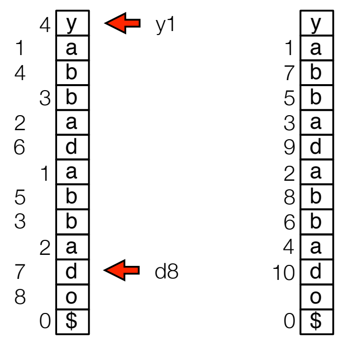
Step 3: Merge



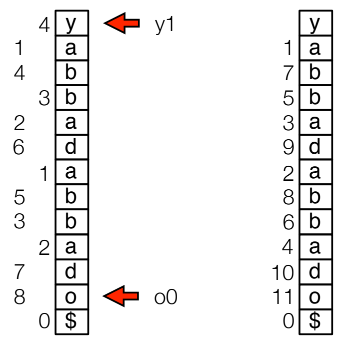
Step 3: Merge



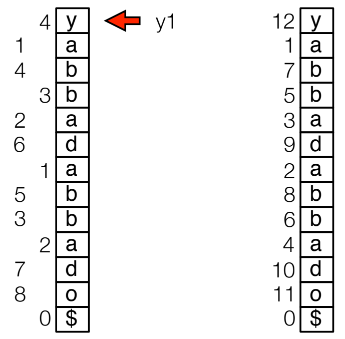
Step 3: Merge



Step 3: Merge



Step 3: Merge



y1
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 $
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 mod 3 and i = 2 mod 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 mod 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)

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