L6: Suffix Arrays

The Problem: input: a string $s \in \Sigma^*$ ← alphabet set (finite) — $\{a, b, c, \ldots, z\}$, think ascii etc.

output: suffix array a of length $|s|+1$ where:

$$s[a[i]:] < s[a[i+1]:]$$

ex: banana$

all suffixes

Suffix	i
banana$	1
anana$	2
nana$	3
ana$	4
na$	5
a$	6
$	7

Sort ⟹ ($ is smaller that everything)

$	7
a$	6
ana$	4
anana$	2
banana$	1
na$	5
nana$	3

⟹ $a = [7, 6, 4, 2, 1, 5, 3]$

alt: $[6, 5, 3, 1, 0, 4, 2]$

Why? Lots of applications — string search, smallest cyclic shift, longest common prefix, number of unique substrings, bzip2.

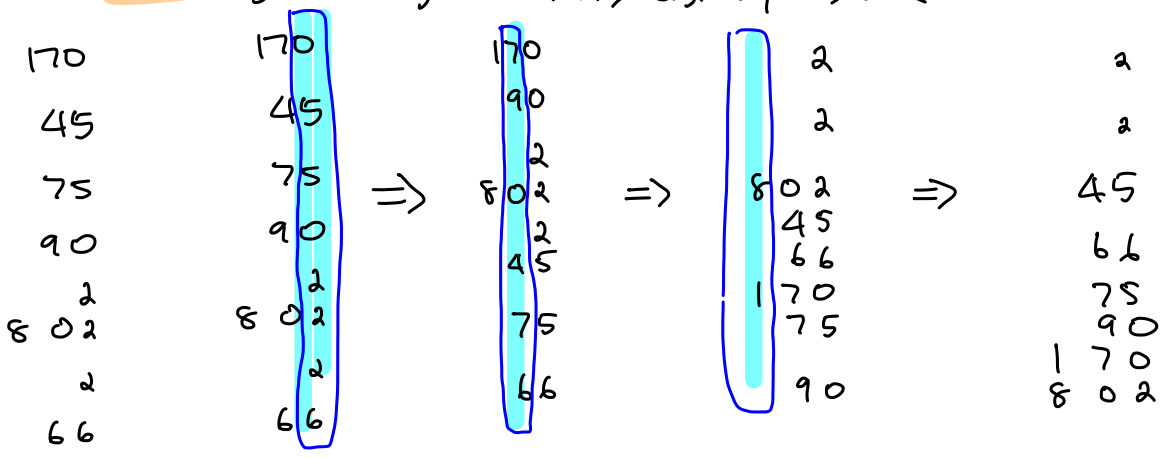
Naive solution: - List out all suffixes (time: $1+2+3+\ldots+n = \Theta(n^2)$)

- Sort: n keys but each compare looks at strings of len $\leq n \Rightarrow O(n)$

$\Rightarrow O(n^2 \log n)$

Goal: Do much better, best known: $O(n)$ time

Digression - Radix Sorting — right to left, digit by digit (each digit: stable sorting)



- # passes = # digits = $\log_b W$ ← max #
- Each pass: $O(n)$ via stable counting sort.

(Avoid comparing the whole strings)

Idea: Sort by piecing together smaller sorted keys

"God" given rank	
ba	1
an	0
na	2

To sort $\Leftrightarrow$ sort

banana = (1,2)

anan = (0,0)

nana = (2,2)

anan
bana
nana

for $k = 0$ to $\lceil \log_2 n \rceil$:

① Sort all substrings of len 2^k (hopefully with help from prev ranks)

② Generate their ranks

Example:

bananas → b a n a n a $

pos: 0 1 2 3 4 5 6

$	a	a	a	b	n	n
6	1	3	5	0	2	4
rank: 0	1	2	3	4	5	6
revised rank: 0	1	1	1	2	3	3

ba	an	na	an	na	a$
0	1	2	3	4	5
(4,1)	(1,6)	(5,2)	(2,6)	(6,3)	(3,0)
(2,1)	(1,3)	(3,1)	(1,3)	(3,1)	(1,0)

→

an	an	a$
1	3	5
(1,6)	(2,6)	(3,0)

a$	an	an	ba	na	na
5	1	3	0	2	4
(1,0)	(1,3)	(1,3)	(2,1)	(3,1)	(3,1)
rank: 0	1	1	2	3	3

bana	anan	nana	ana$	na$	a$
0	1	2	3	4	5
(2,3)	(1,1)	(3,3)	(1,0)	(3,$)	(0,$)

⇒

a$	ana$	anan	bana	na$	nana
5	3	1	0	4	2
(0,$)	(1,0)	(1,1)	(2,3)	(3,$)	(3,3)
rank: 0	1	2	3	4	5

Running time: - $\log_2 n$ iterations

- each involves < poor man's hash table O(n) / radix sort (2 digits)

$\Rightarrow O(n \log n)$

Suffix Arrays < in parallel / O(n) work