Suffix sorting

Lecture 5.1

Algorithm based on Larsson fast suffix sorting

Reading:

http://www.larsson.dogma.net/ssrev-tr.pdf

How do we construct the suffix array

- The suffix array can be constructed from the suffix tree
- Why NOT to do it:
  - The suffix tree construction algorithms are complex
  - We need an intermediate space to store the suffix tree – which may be too big!

Larsson algorithm: intuition

- Sort suffixes by prefix of length 1 character
- Now, in order to sort suffixes by prefix of length 2, we can look at the results of the previous sorting at position i+1
- Once the suffixes are sorted by prefix of length 2, we can now produce a suffix order for prefixes of length 4, by looking at the results of the previous step at position i+2
- Once suffixes are sorted by prefix of length 4, we can immediately produce sorting of 8-character prefixes by looking at the results at position i+4

- At each iteration *h*, we produce total suffix sorting for prefixes of length 2^h, and in at most **log N** iterations we produce the final ranks for each suffix in the suffix array

Larsson suffix sorting

- Complexity: O(N log N)
- Assumption: the entire input string is in memory and all the intermediate ranks are in memory to be read at random position in a constant time

SAMPLE RUN OF THE LARSSON ALGORITHM

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

Sort (bucket or merge sort) by the first character of each suffix:

h-order with h=1

	$	a	a	c	h	h	h	i	u	u
SA (Start pos of sorted suffixes)	9	5	8	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	1	3	4	4	4	7	8	8
Group length	1	-2		1	-3			1	-2	

For the next step we need rank (SA[X]+1)

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+1

h-order with h=2

	$	a	a	c	h	h	h	i	u	u
Start pos	9	5	8	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	1	3	4	4	4	7	8	8
Group length	1	-2		1	-3			1	-2	

Rank 1 for a at position 5 is followed by rank 4, while rank 1 for a at position 8 is followed by rank 0, so we can resolve ranks for two a's

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+1

h-order with h=2

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	4	4	7	8	8
Group length	1	1	1	1	-3			1	-2	

Rank 1 for *a* at position 5 is followed by rank 4, while rank 1 for *a* at position 8 is followed by rank 0, so we can resolve ranks for two *a*'s

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+1

h-order with h=2

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	5	5	7	8	8
Group length	1	1	1	1	1	-2	-2	1	-2	

Similarly, we resolve ranks for h1, h3 and h6:
h1 – (4,7), h3 – (4,8), h6 – (4,8)

and for u4 and u7:
u4 – (8,1), u7 – (8,1)

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+1

h-order with h=2

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	5	5	7	8	8
Group length	1	1	1	1	1	-2	-2	1	-2	

Because prefixes of length 2 are already sorted, next we look at ranks at position SA[X] + 2

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+2

h-order with h=4

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	3	6	2	4	7
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	5	5	7	8	8
Group length	1	1	1	1	1	-2	-2	1	-2	

To resolve ranks for h3 and h6:
h3 – (5,2), h6 – (5,1)

To resolve ranks for u4 and u7:
u4 – (8,5), u7 – (8,0)

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

To resolve equal ranks we look at ranks at position i+2

h-order with h=4

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	6	3	2	7	4
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	5	6	7	8	9
Group length	1	1	1	1	1	1	1	1	1	1

To resolve ranks for h3 and h6:
h3 – (5,2), h6 – (5,1)

To resolve ranks for u4 and u7:
u4 – (8,5), u7 – (8,0)

pos	c	h	i	h	u	a	h	u	a	$
i	0	1	2	3	4	5	6	7	8	9

All suffixes now have their unique distinct rank: all are sorted

	$	a	a	c	h	h	h	i	u	u
Start pos	9	8	5	0	1	6	3	2	7	4
Pos in SA: X	0	1	2	3	4	5	6	7	8	9
rank	0	1	2	3	4	5	6	7	8	9
Group length	1	1	1	1	1	1	1	1	1	1

Final suffix array

SA	9	8	5	0	1	6	3	2	7	4

c	h	i	h	u	a	h	u	a	$
0	1	2	3	4	5	6	7	8	9

Checking suffix order

SA2	9	8	5	0	1	6	3	2	7	4
	$	a	a	c	h	h	h	i	u	u
		$	h	h	i	u	u	h	a	a
			u	…	h	a	a	…	$	h
			a		…	$	h			u
			$				u			…
							…			

SA	9	8	5	0	1	6	3	2	7	4

It works!