CS533 Information Retrieval Lecture 9 - 24/03/2014

CS533-Information Retrieval lecture notes

Lecturer: Fazli Can *Scribe: Abdurrahman Yaşar*

1 PAT-Trees (Patricia Trees)

Traditional information retrieval systems has several problems. In the traditional model there are documents and words and keywords must be extracted from the text which we call indexing. In the other hand queries are restricted to keywords.

PAT tree introduces new indeces to text. It looks a text as a long string. Each position corresponds to a semi infinite string(sistring) and there is no structure or keywords.

1.1 Semi-Infinite Strings

Here, in the above table, you see how to calculate a sistring:

Bit Position	1	2	3	4	5	6	7	8	9	10	11	12
Text(string)	0	1	1	0	0	1	0	0	0	1	0	1
sistring1	0	1	1	0	0	1	0	0	0	1	0	1
sistring1	1	1	0	0	1	0	0	0	1	0	1	
sistring1	1	0	0	1	0	0	0	1	0	1		
sistring1	0	0	1	0	0	0	1	0	1			
sistring1	0	1	0	0	0	1	0	1				
sistring1	1	0	0	0	1	0	1					
sistring1	0	0	0	1	0	1						
sistring1	0	0	1	0	1							

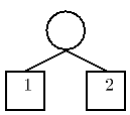
1.2 Construction of a PAT tree

We construct a PAT tree over all possible sistrings of a text.

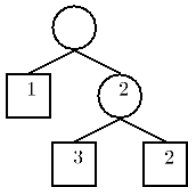
1. □ : external node sistring (integer displacement) total displacement of the bit to be inspected
2. ○ : internal node skip counter & pointer

sis1:

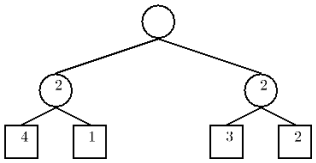
sis2:



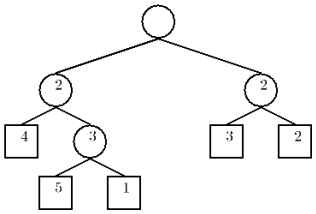
sis3:



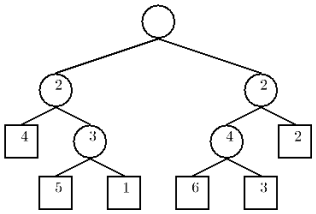
sis4:



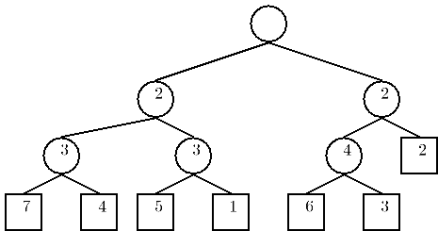
sis5:



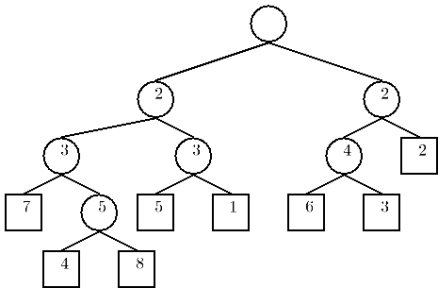
sis6:



sis7:



sis8:



1.3 PAT Array

We can also use pat array for searching purposes. All leaf nodes are in lexicographic order so we can use just binary search on the array. Update could be more complex. PAT array of the previous example:

7	4	8	5	1	6	3	2

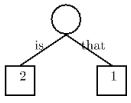
1.4 PAT tree on a non binary vocabulary

Text : "that is that that is not that"

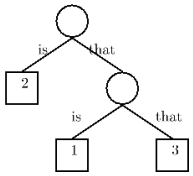
Position	1	2	3	4	5	6	7
sis1	that	is	that	that	is	not	that
sis1	is	that	that	is	not	that	
sis1	that	that	is	not	that		
sis1	that	is	not	that			
sis1	is	not	that				

sis1:

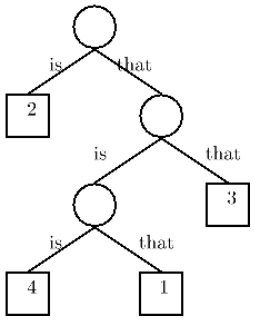
sis2:



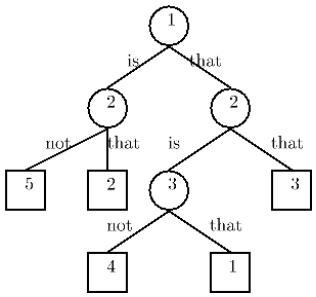
sis3:



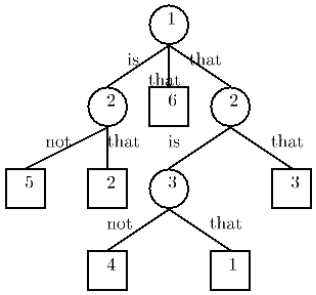
sis4:



sis5:



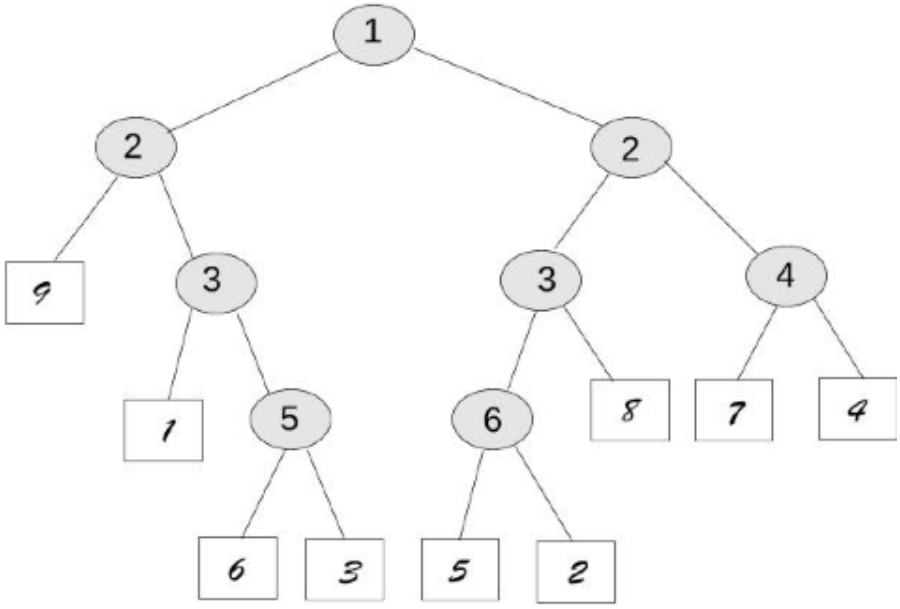
sis6:



2 Questions

2.1 Question 1:

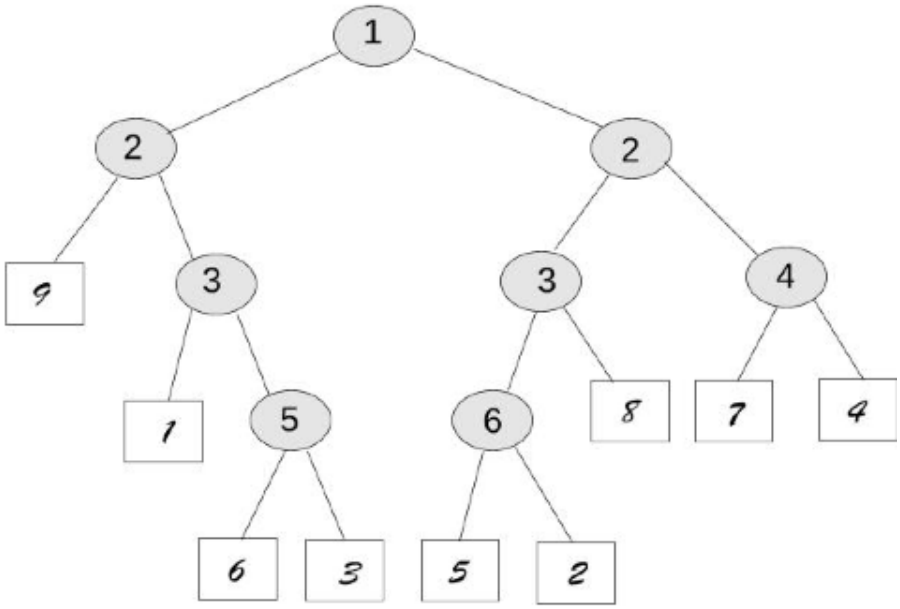
Write the text of given tree:



Answer : 01011011000111

2.2 Question 2:

Find the most frequent substring of length 2?



Answer : 01 and 10 (3 times)

2.3 Question 3

Construction of a PAT array for small strings can be easily done in memory. How can you do this for very large texts? Give an algorithm.

Hint:

- Split the text into small partitions
- For each partition construct a PAT array
- Merge these PAT arrays (How can you minimize random io while merging? $merge(small\_partition, large\_partition)$ or $merge(large\_partition, large\_partition)$)