Applications of suffix trees

Lecture 4

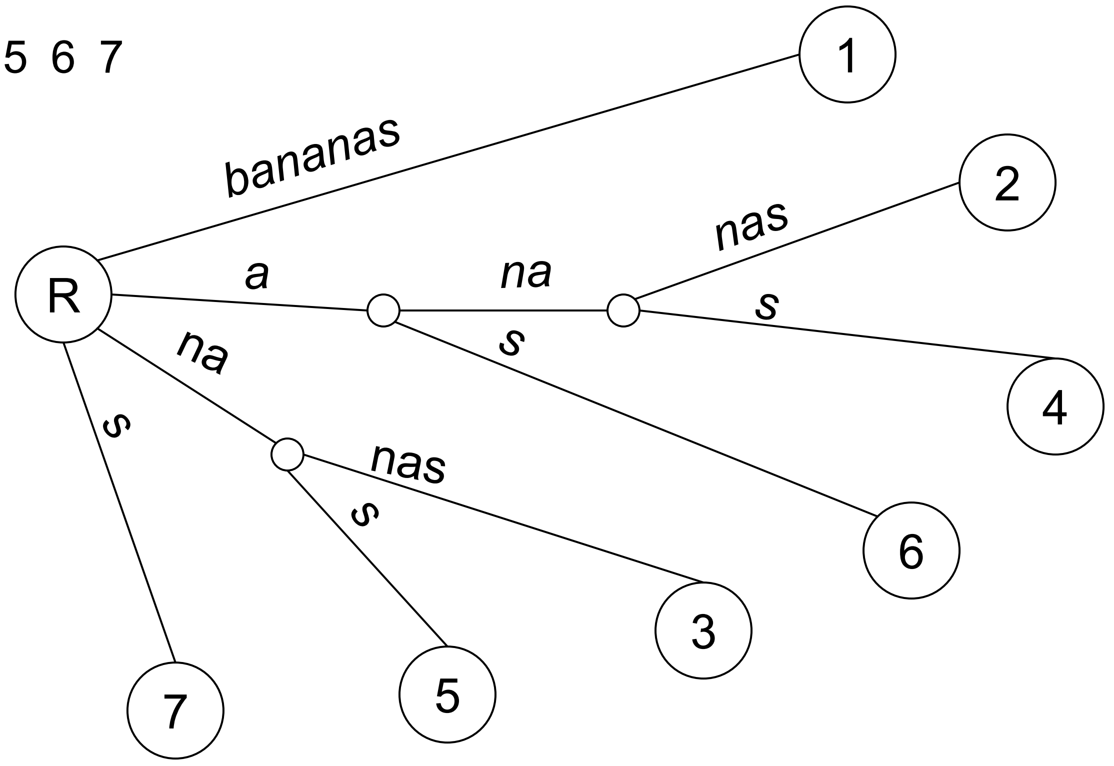
Suffix tree - summary

- Suffix tree is a digital tree of all suffixes of text *T* (of length *N*)
- The suffixes are inserted by following a path of characters from the root, and a new branch of a tree is created only if the next character in a current suffix does not match the existing path
- Suffix tree has *N* leaves (1 for each suffix), where we store the starting position of a suffix in *T*
- In order for each suffix to have its own leaf, we need to have at the end of *T* a special character which can not be found anywhere else in *T* – this ensures that a special branch will be created for each suffix which is also a prefix of another suffix

Example: *bananas*

b a n a n a s

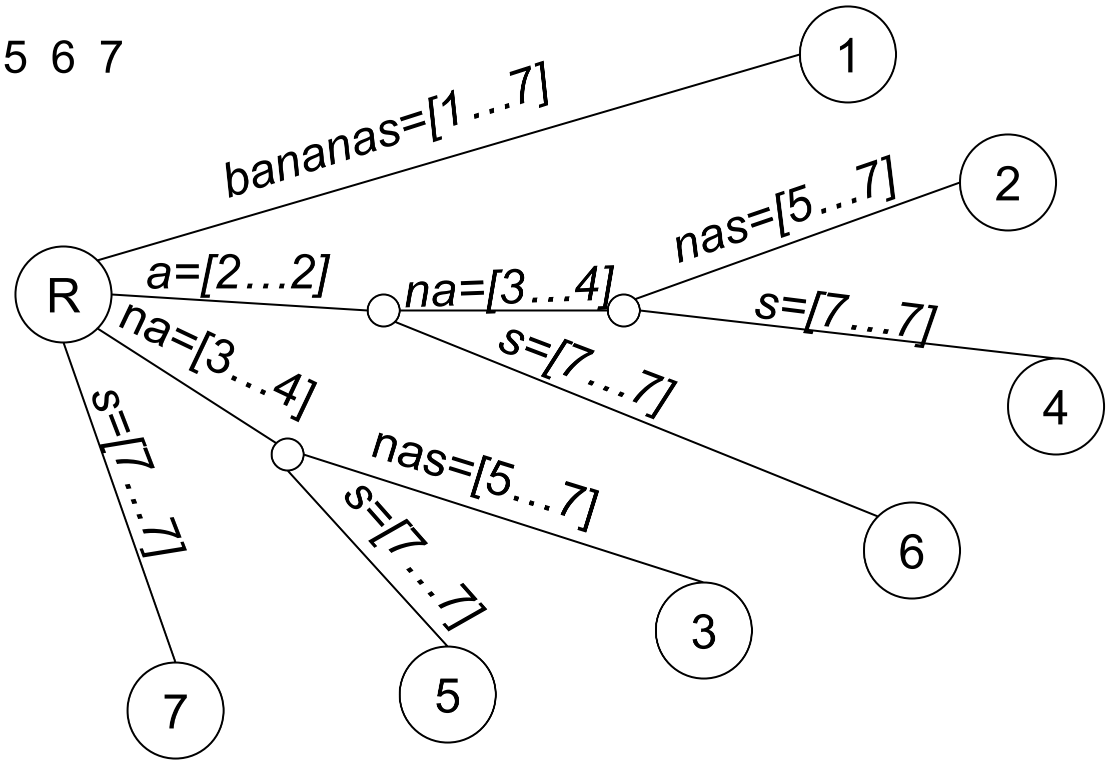
1 2 3 4 5 6 7



Space reduction

b a n a n a s

1 2 3 4 5 6 7



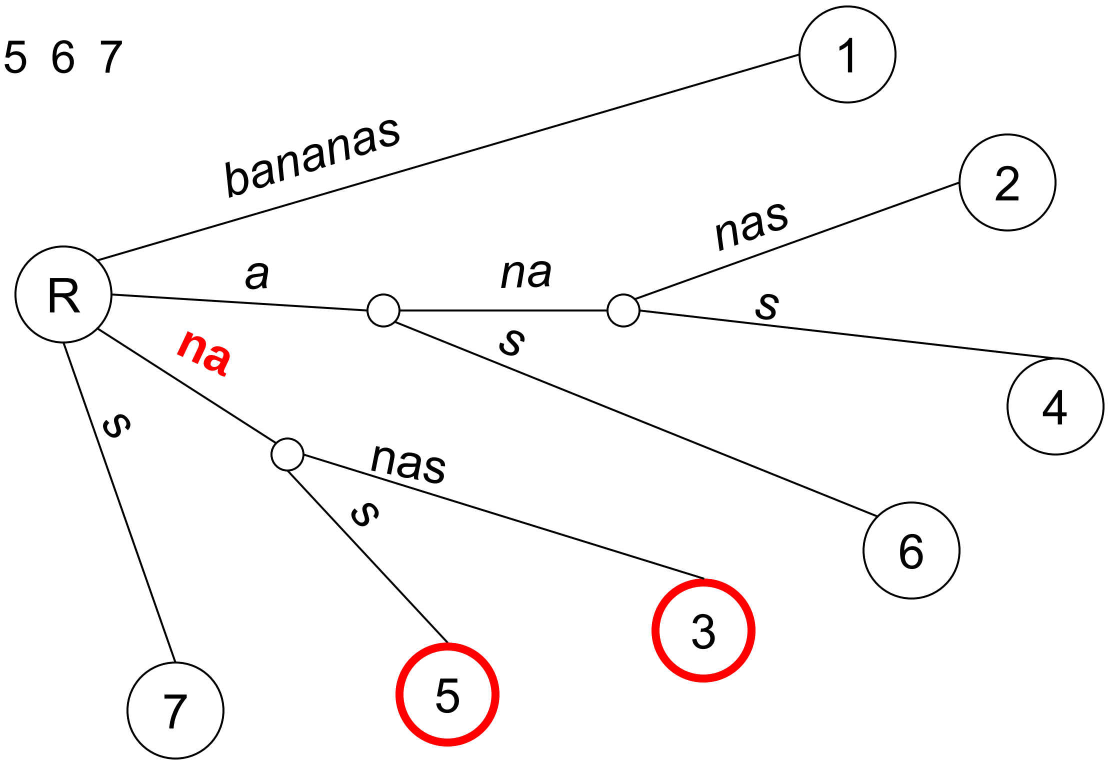
Suffix tree - search

- In order to find all the occurrences of pattern *P* (of length *M*), follow the path of symbols from the root. If there is a path corresponding to all *M* symbols of *P*, the positions where *P* occurs in *T* can be collected by the depth-first traversal of the subtree below the end of this path.
- If there is no path in T for all the characters of *P*, then pattern *P* does not occur in *T*

Example: *P=na*

b a n a n a s

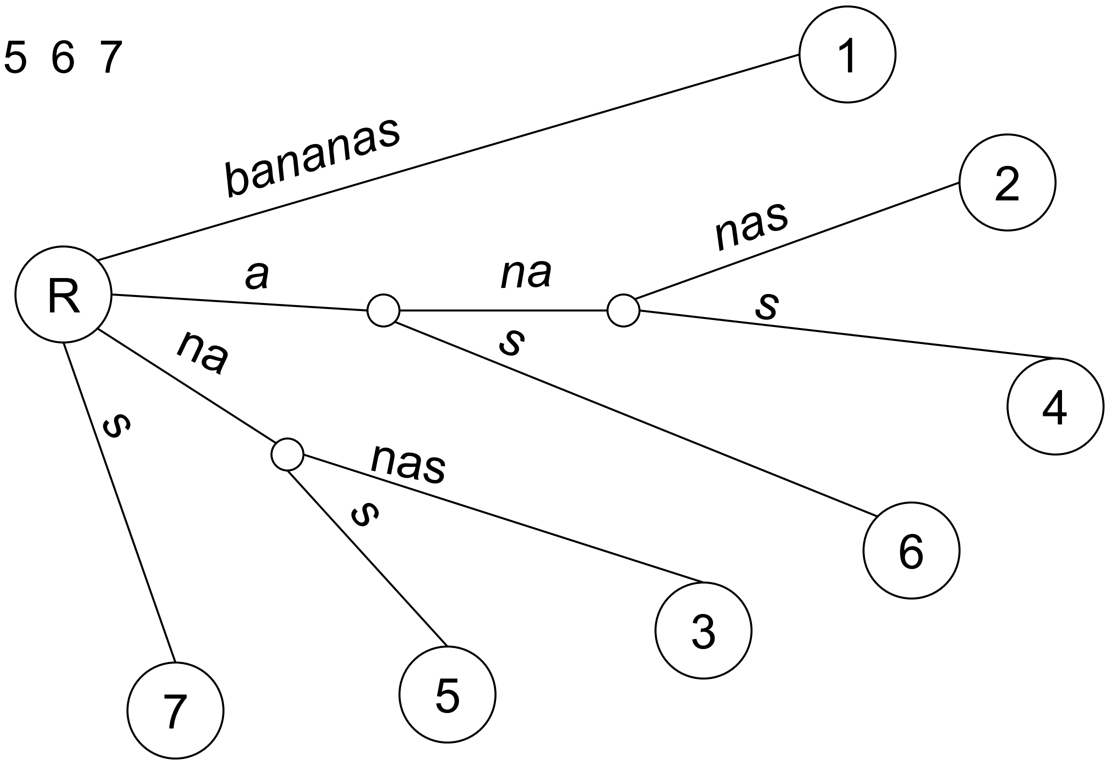
1 2 3 4 5 6 7



Example: *P=an*

b a n a n a s

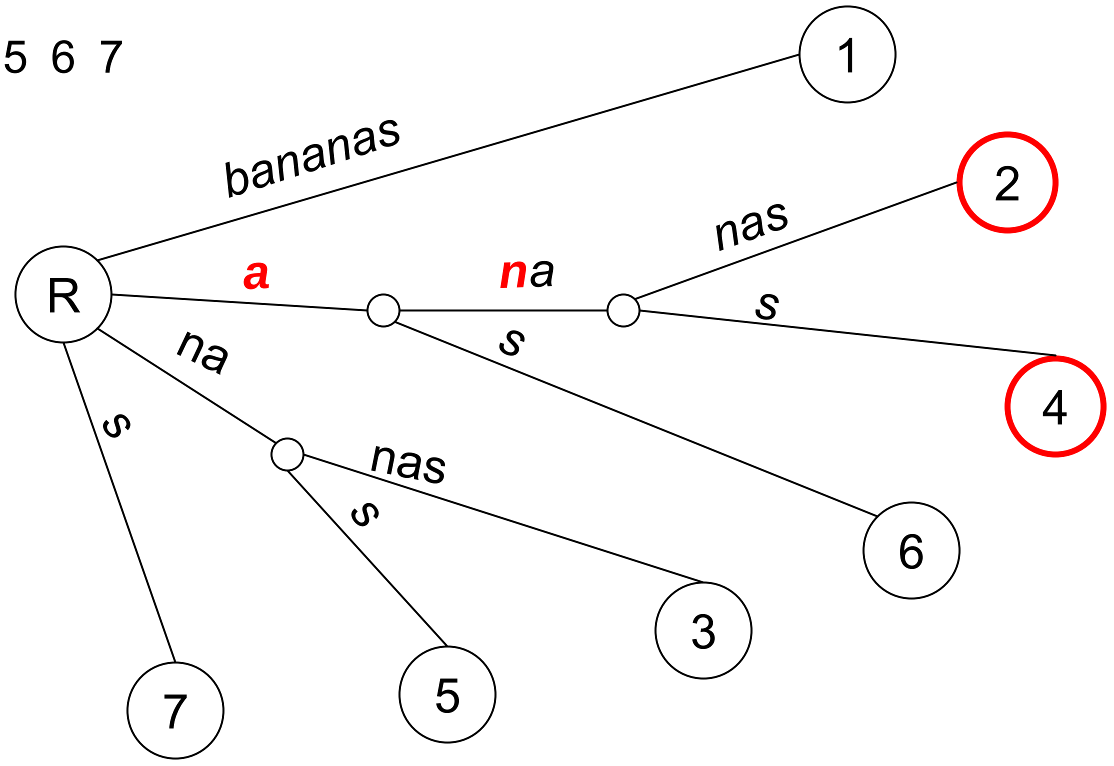
1 2 3 4 5 6 7



Example: *P=an*

b a n a n a s

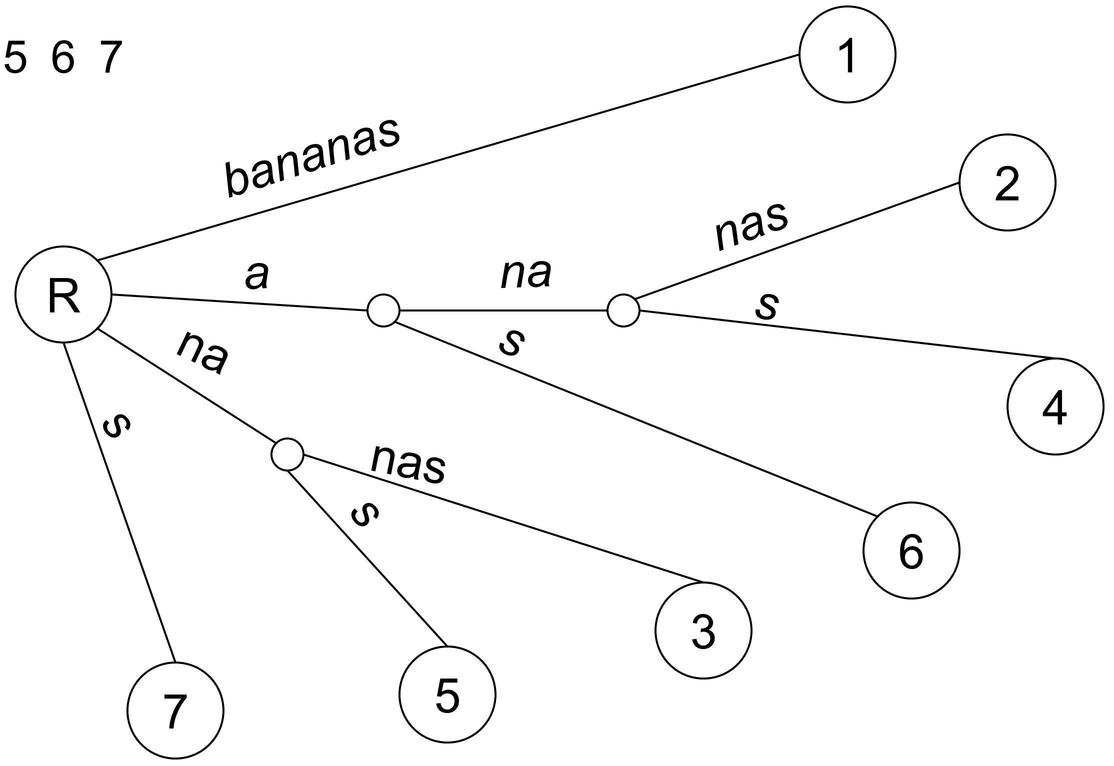
1 2 3 4 5 6 7



Example: *P=naa*

b a n a n a s

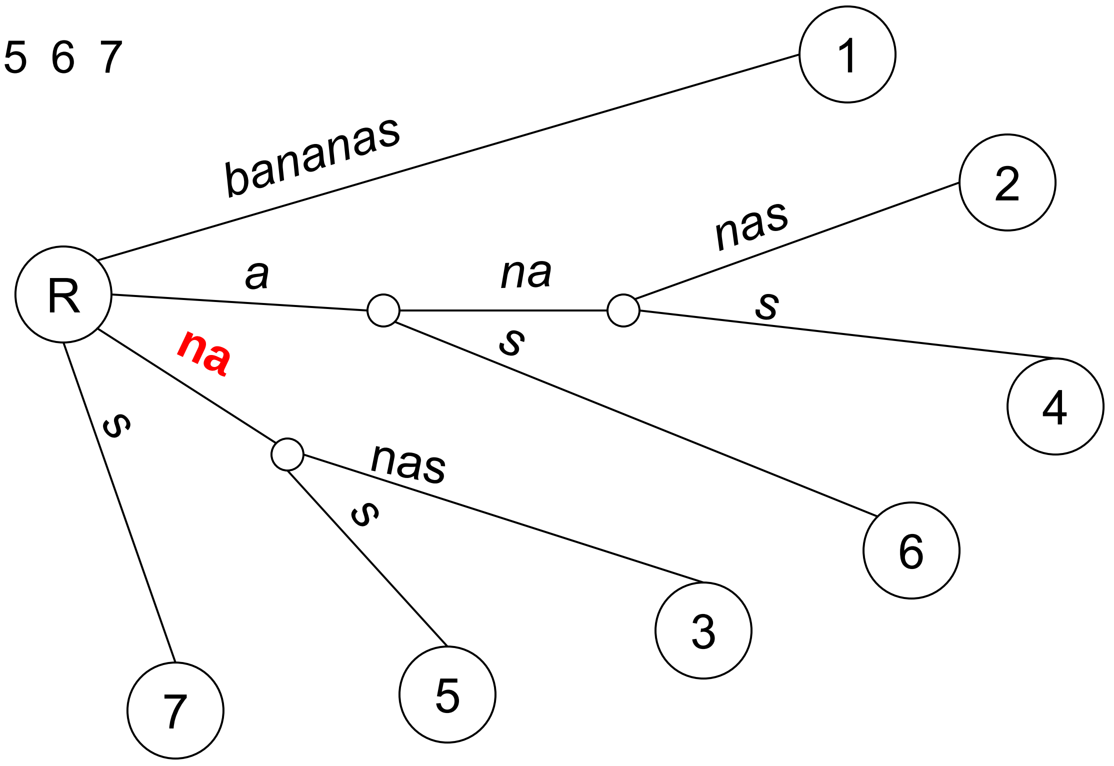
1 2 3 4 5 6 7



Example: *P=naa*

b a n a n a s

1 2 3 4 5 6 7



Not found

Repetitions observed in genome sequences

- Families of reiterated sequences account for about one third of the human genome
- For $3.6*10^6$ nucleotides of the C. Elegans genome, 7000 families of repetitive sequences were discovered
- Prokaryotes[1)] have in total little repetitive DNA
- The mechanism – unequal crossing-over

[1)] Prokaryotes (for example, Bacteria) have a circular DNA not enclosed into a nucleus

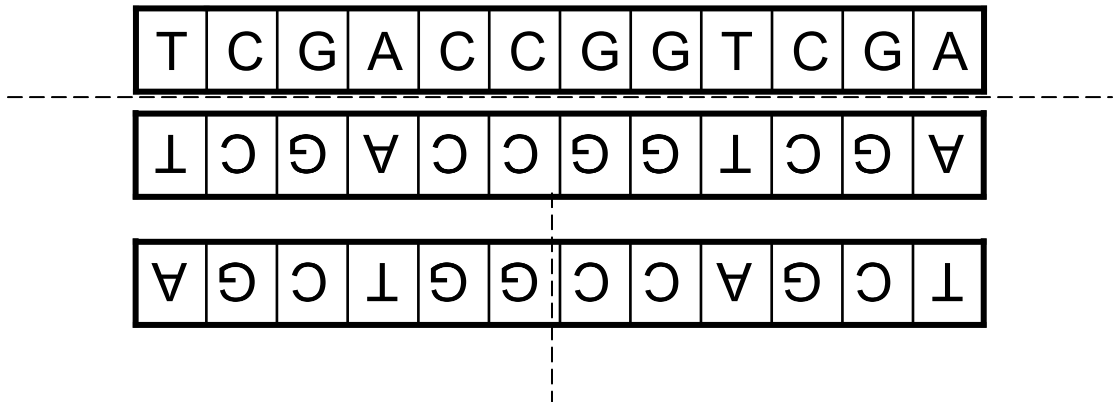
Repetitions in genome sequences I

- Gene families
  - Many genes occur in multiple copies in the genome
  - They may be identical copies (r-RNA genes) or just similar sequences of the same gene, modified by mutations
  - Some contain only a short similar motif – homeobox (~160 Bp)– which defines the shape of the protein-binding site
  - The copies may occur in tandem (one after another) or are dispersed through different areas of the genome
- Some of these multiple copies serve the purpose of an enhanced gene expression (r-RNA), others are redundant and are used interchangeably when one copy is damaged
- The copy of the original gene can mutate and acquire a new function
- This is believed to be a main mechanism of evolution

Repetitions in genome sequences II

- Functional repeats – short repeats encoding the same functional sites (transcription sites and protein-binding sites on DNA)
- They often have a form of a *complemented palindrome*

T	C	G	A	C	C	G	G	T	C	G	A
A	G	C	T	G	G	C	C	A	G	C	T



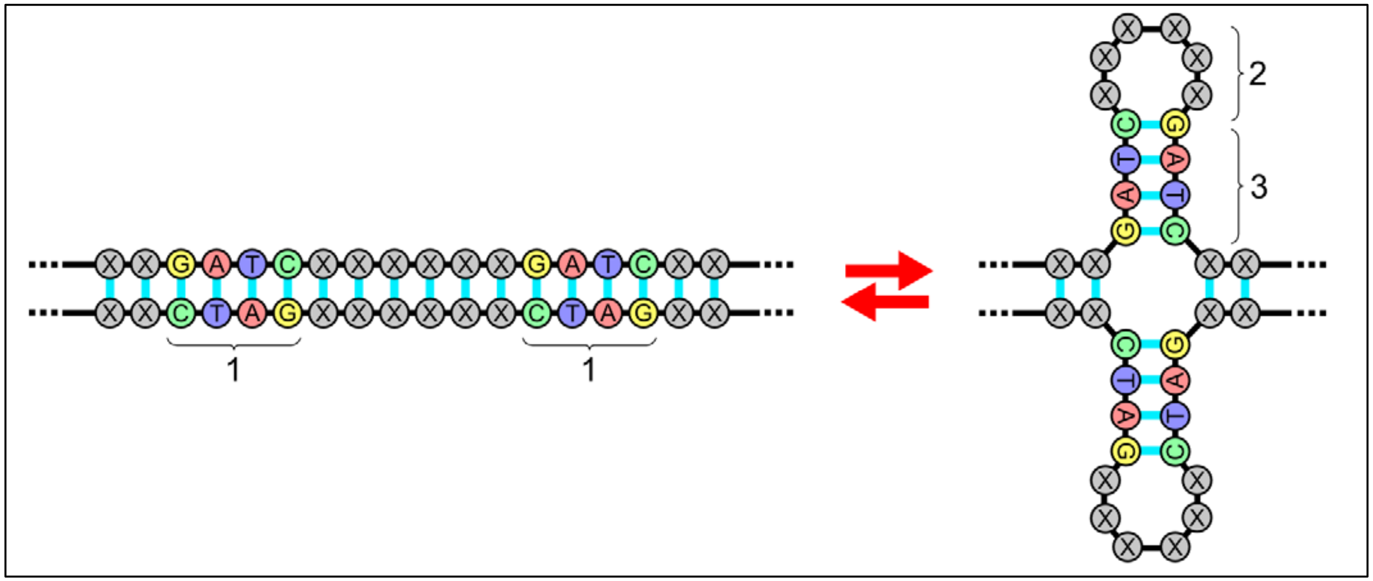
1 rotation around the horizontal

and

1 around the vertical axis

Repetitions in genome sequences II

- These complemented palindromic repeats have a potential to form secondary structures such as hairpins and stem loops reflecting the dimeric nature of proteins. They serve for recognition of the transcription sites by enzymes



Repetitions in genome sequences III

- Transposons – dispersed repetitive elements – the remains of viral DNA which were incorporated into genome and lost their functionality
  - SINE – Short Interspersed Nuclear Elements – *Alu* element (in human but not in mouse) 300 bp flanked by direct repeats – 10^6 copies, 1 such element per each 4 kBp sequence
  - LINE – Long Interspersed Nuclear Element – L1 element – 6 kbp long and 10^5 copies They are not in the protein-coding regions, but often in introns, or at the ends of the transcribed region, so they are transcribed as a part of a gene

Repetitions in genome sequences IV

- <u>Satellite sequences</u>

<u>Microsatellites </u>– distributed through the entire genome 1-4 bp repeats in clusters of ~200 Bp. They are highly polymorphic (in the number of copies) and make an ideal genetic marker. *VNTR*, variable number of tandem repeats, is used for personal identification

- When these repeats are inside a protein-coding region, they cause severe diseases (for example, Huntington's disease, if more than 20 *CAG* repeats are present inside the coding region for the *huntingtin* protein)

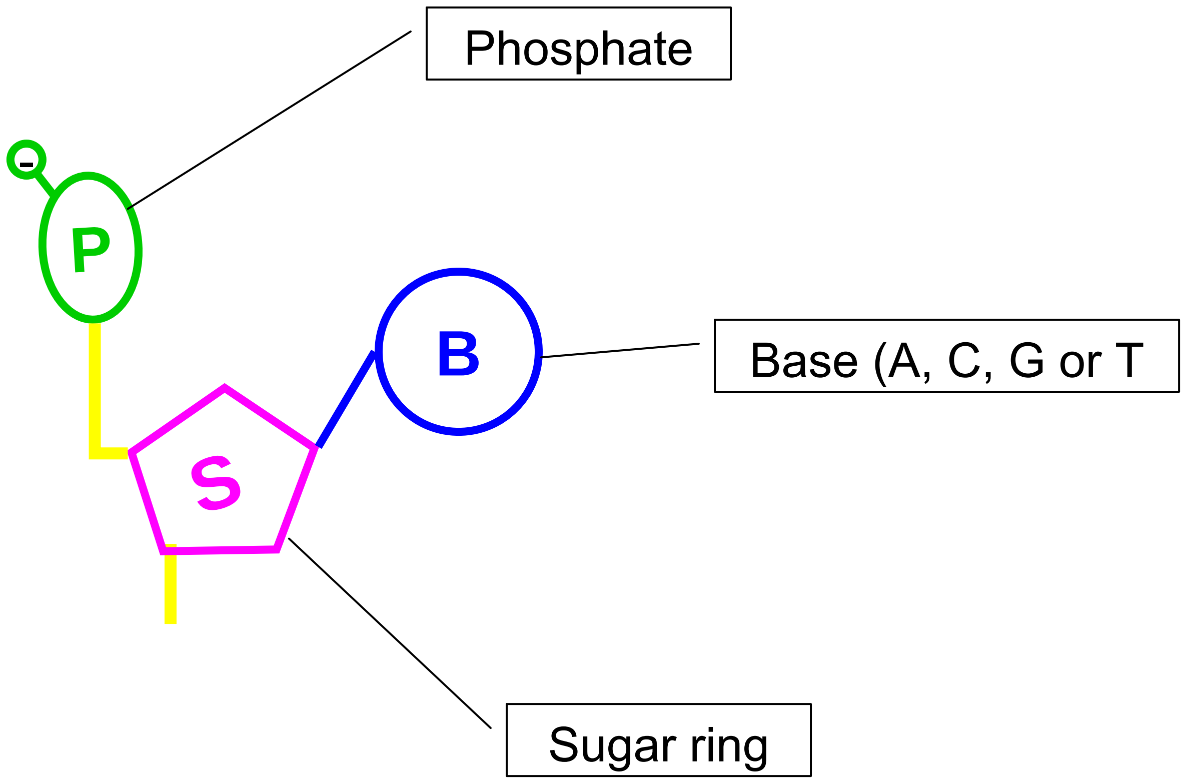
Repetitions in genome sequences IV

- <u>Satellite sequences</u>

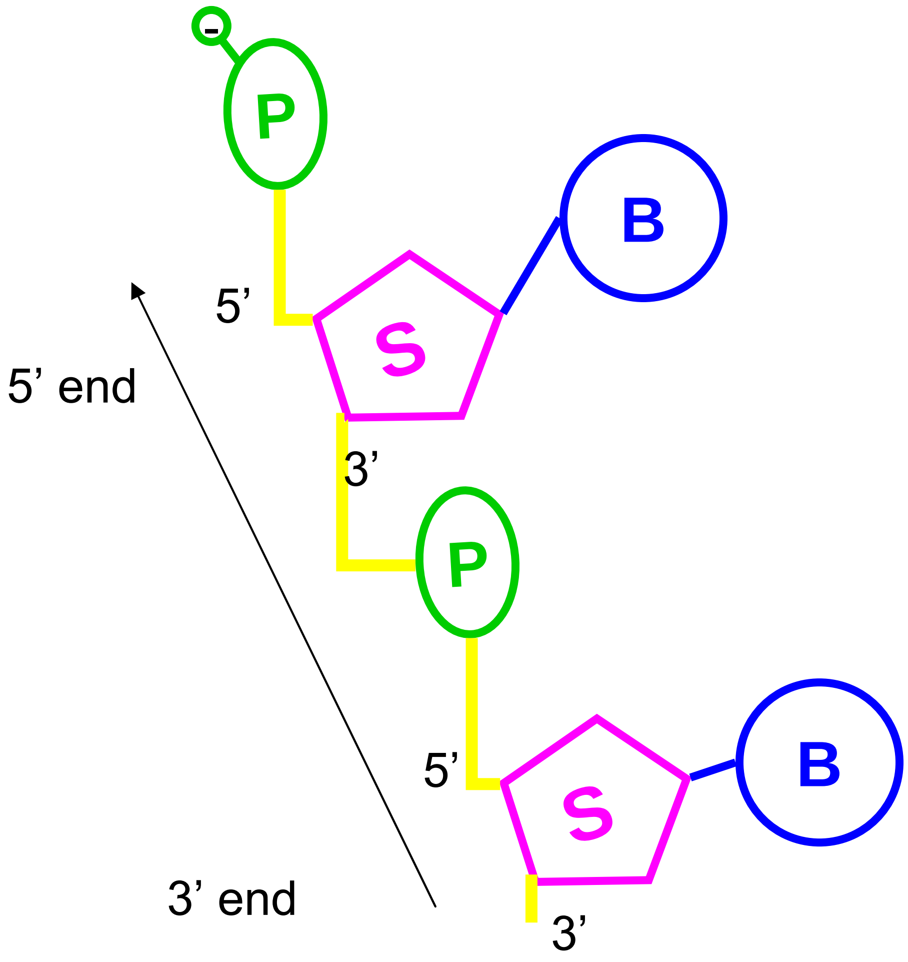
<u>Minisatellites </u>– occur as tandem repeats at the end of chromosomes

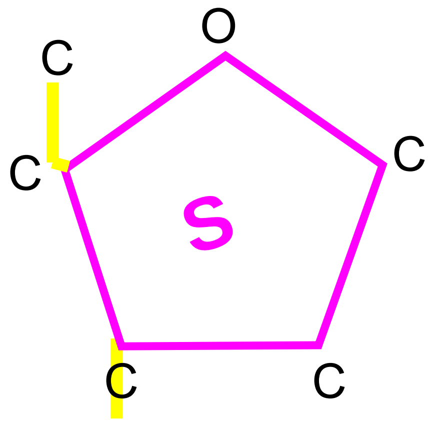
They have an important function discussed below

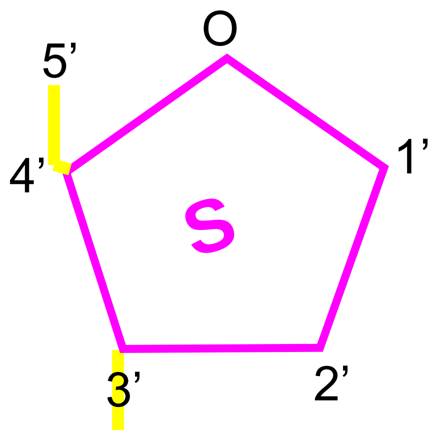
Nucleotide – DNA building block



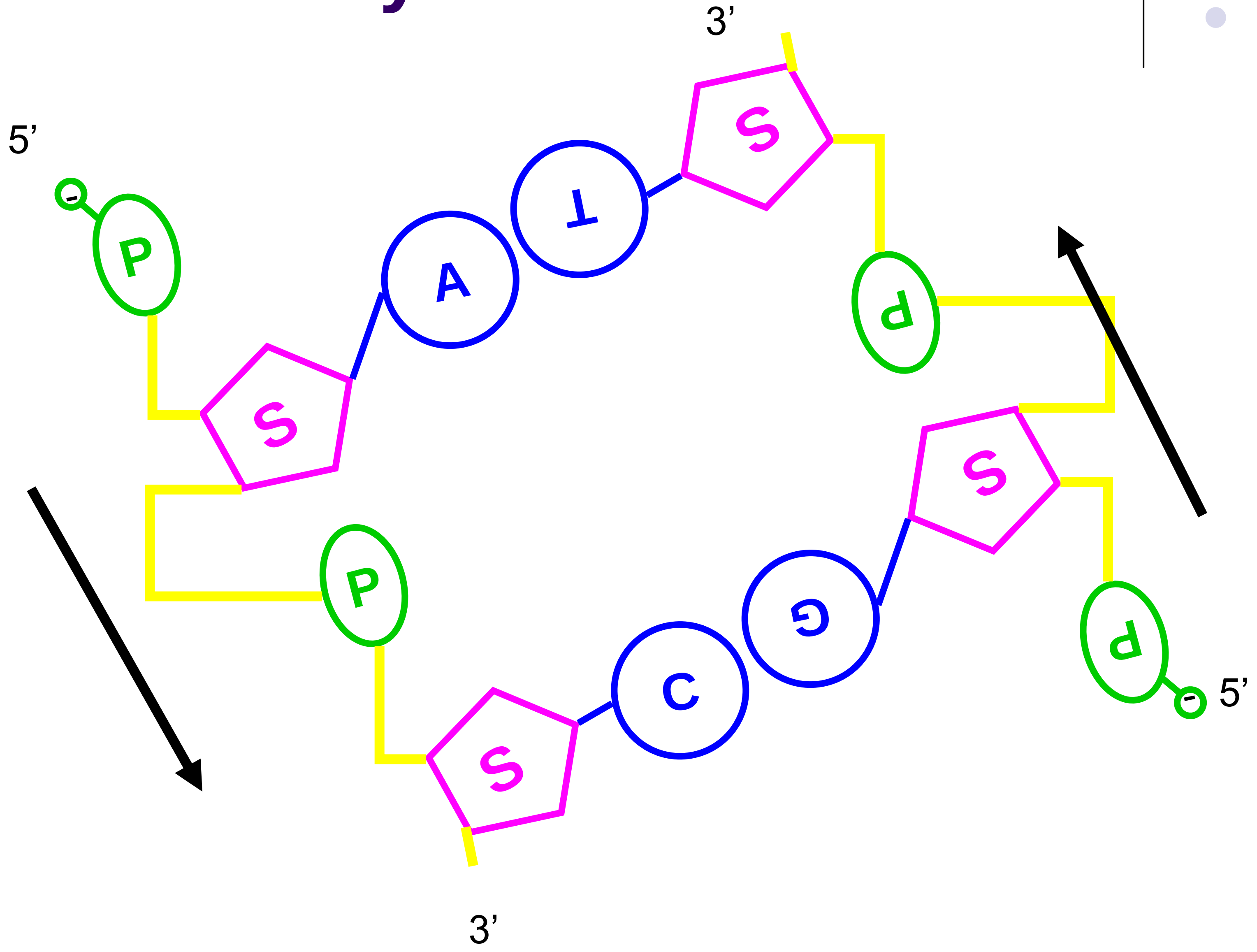
Chaining nucleotides



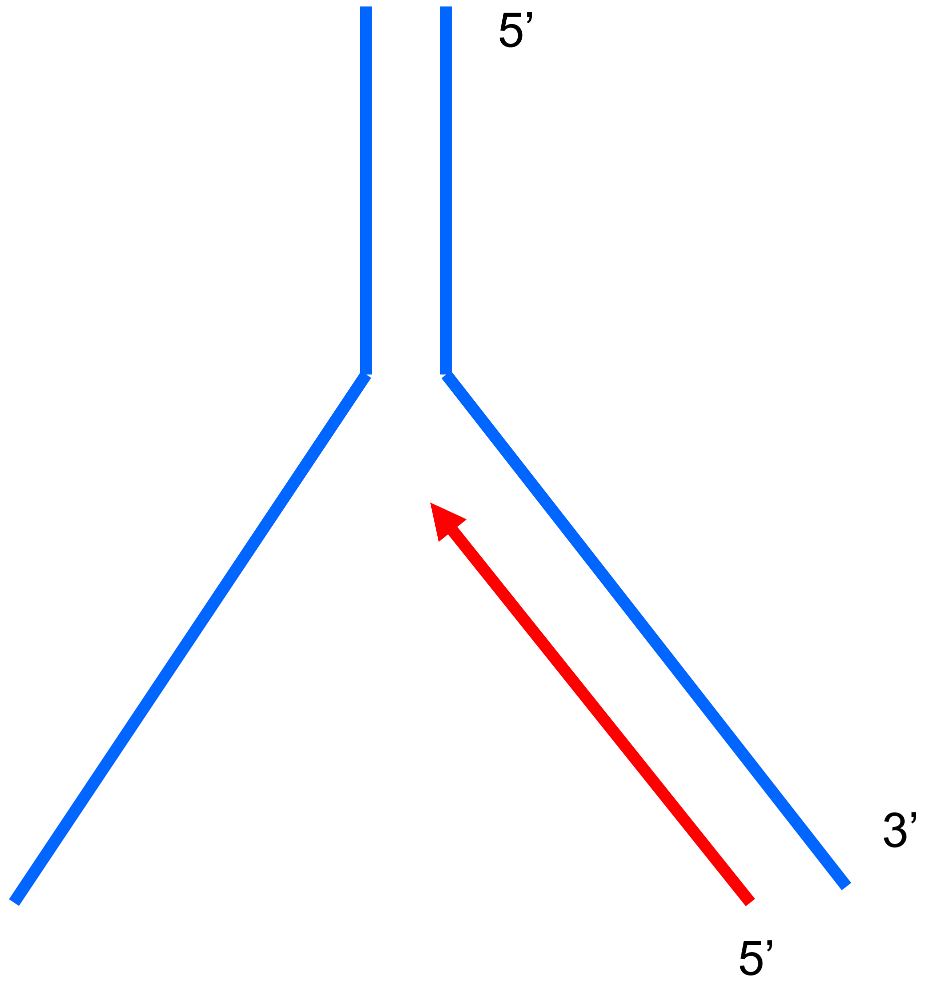




Synthesis of the chain can be performed only in one direction

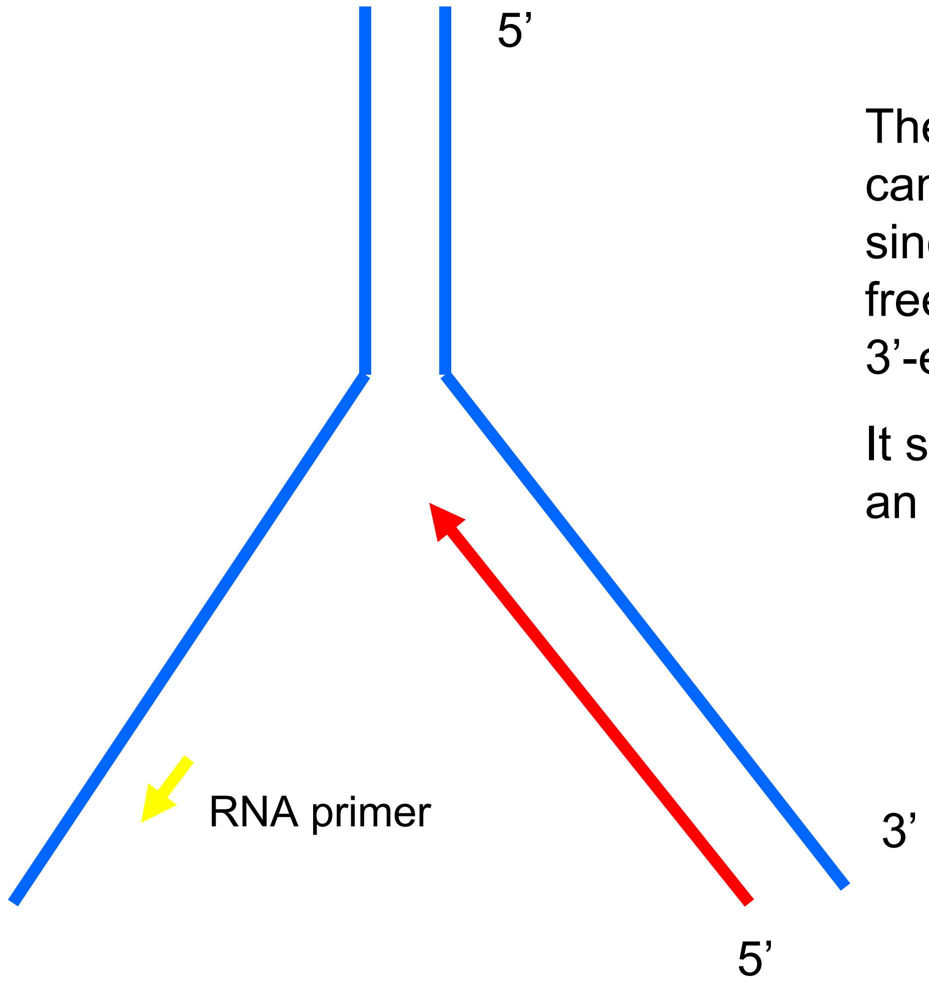


DNA Replication



The leading strand is replicating without problems

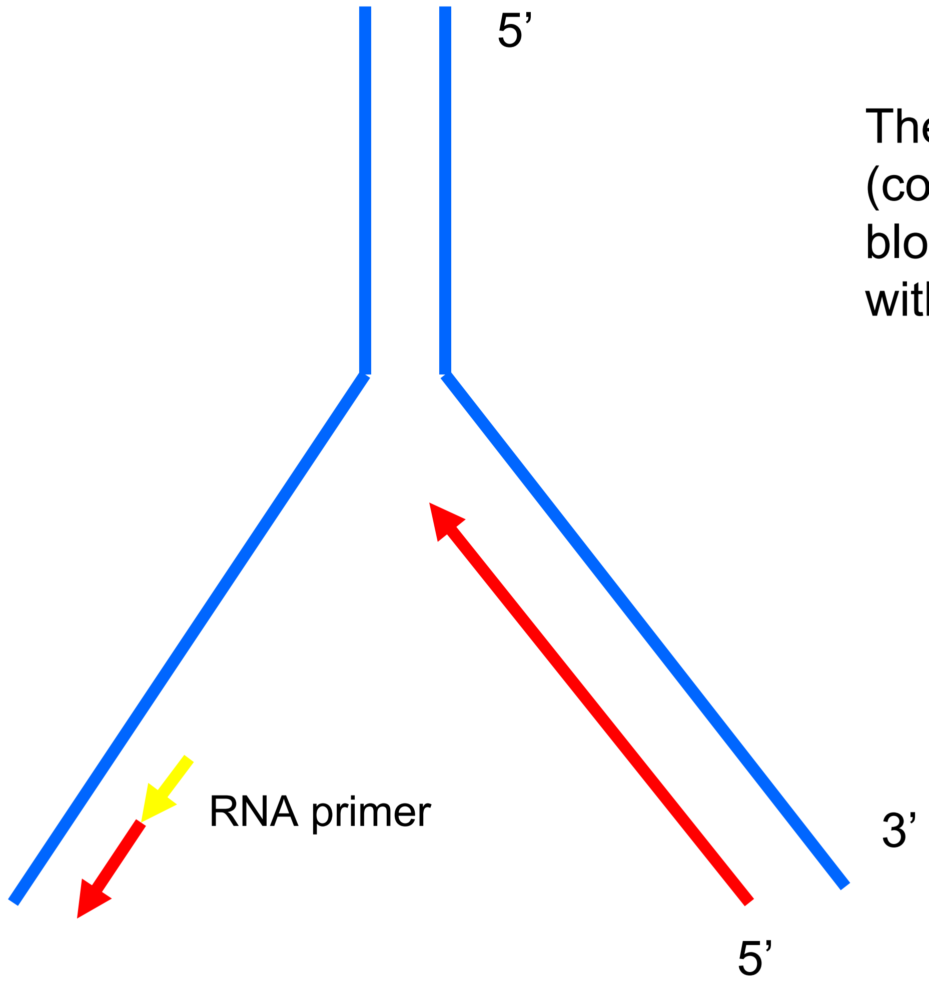
DNA Replication



The lagging strand cannot even start, since there is no free complementary 3’-end

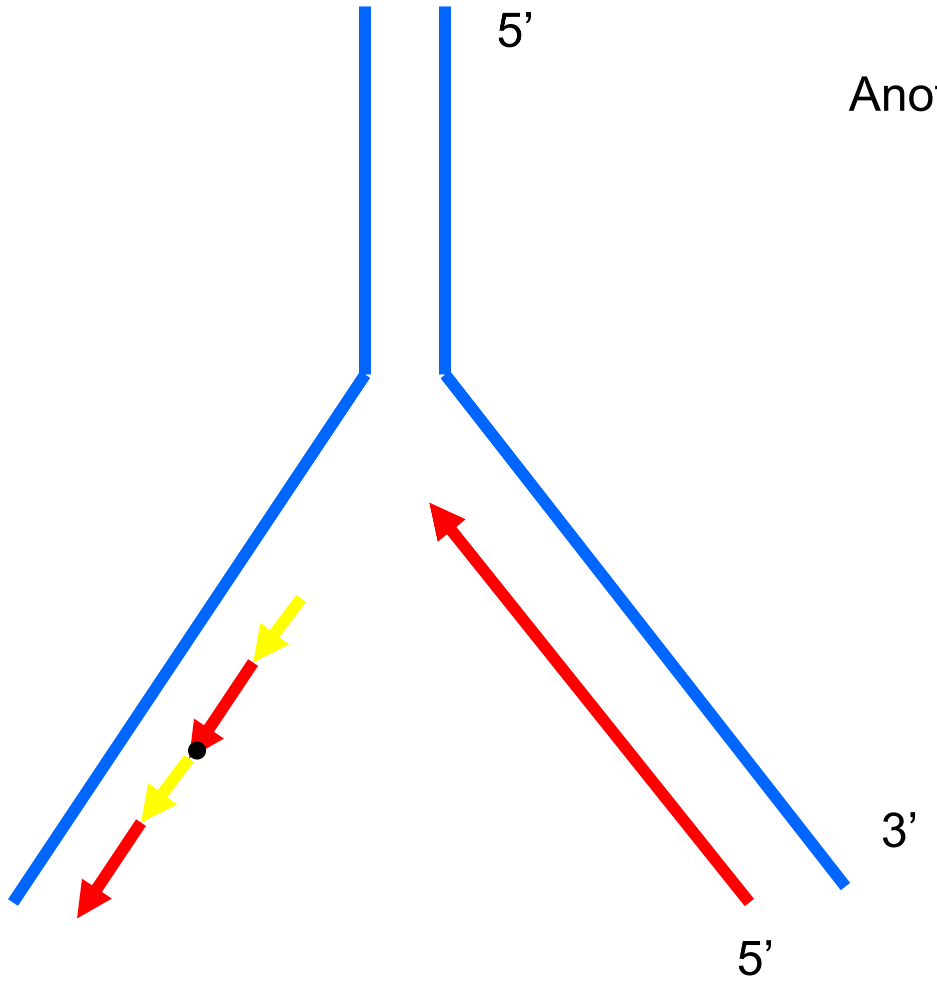
It starts by creating an RNA primer

DNA Replication



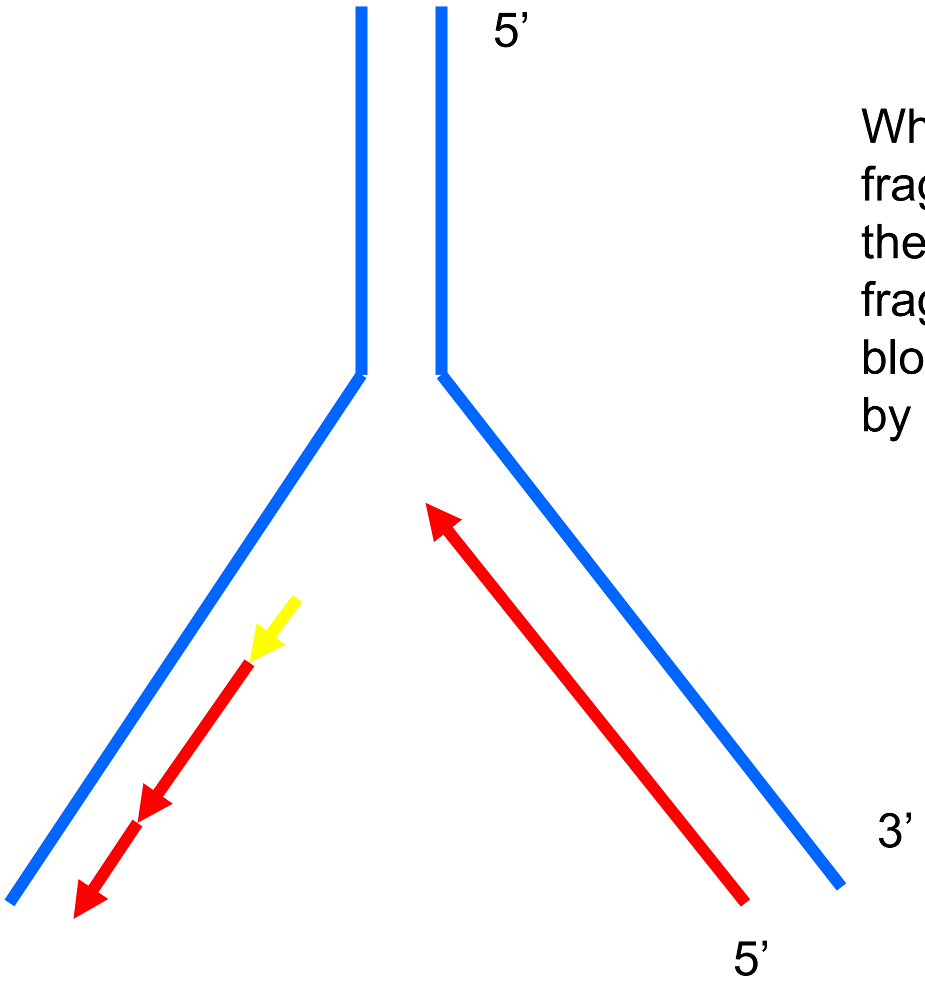
The primer (consisting of RNA blocks) is extended with DNA blocks

DNA Replication



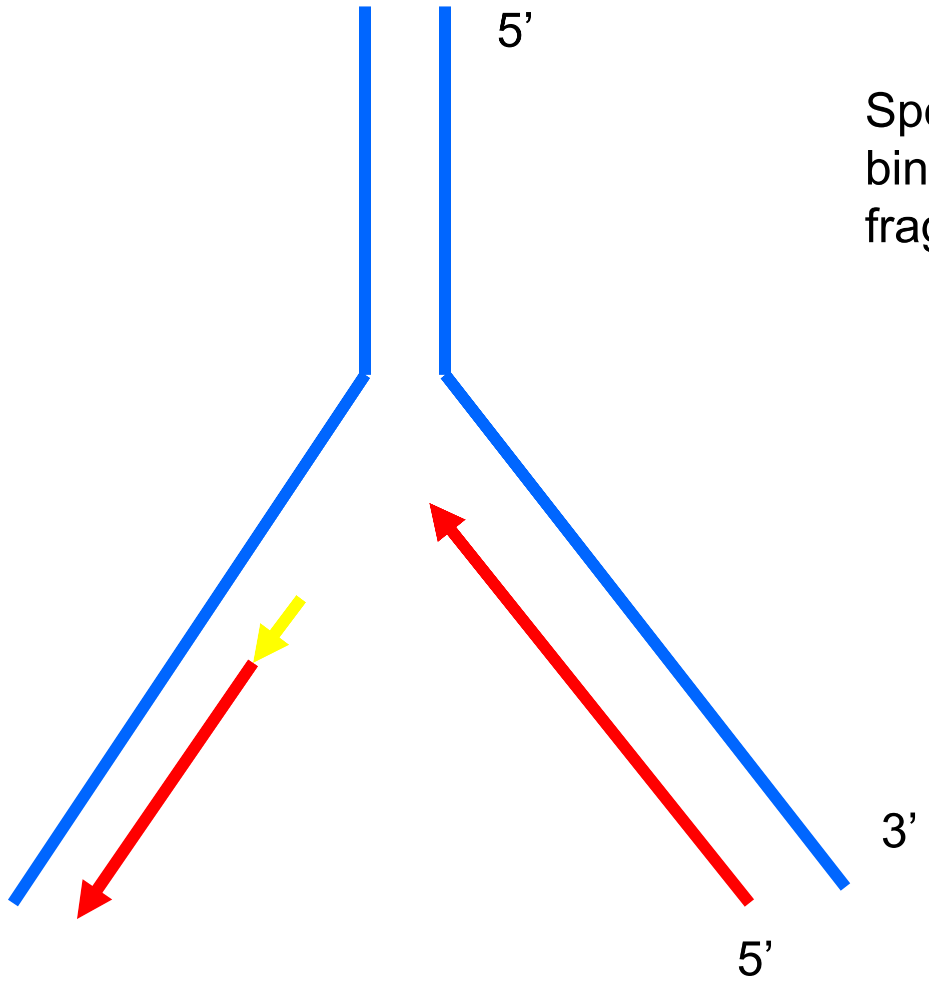
Another fragment

DNA Replication



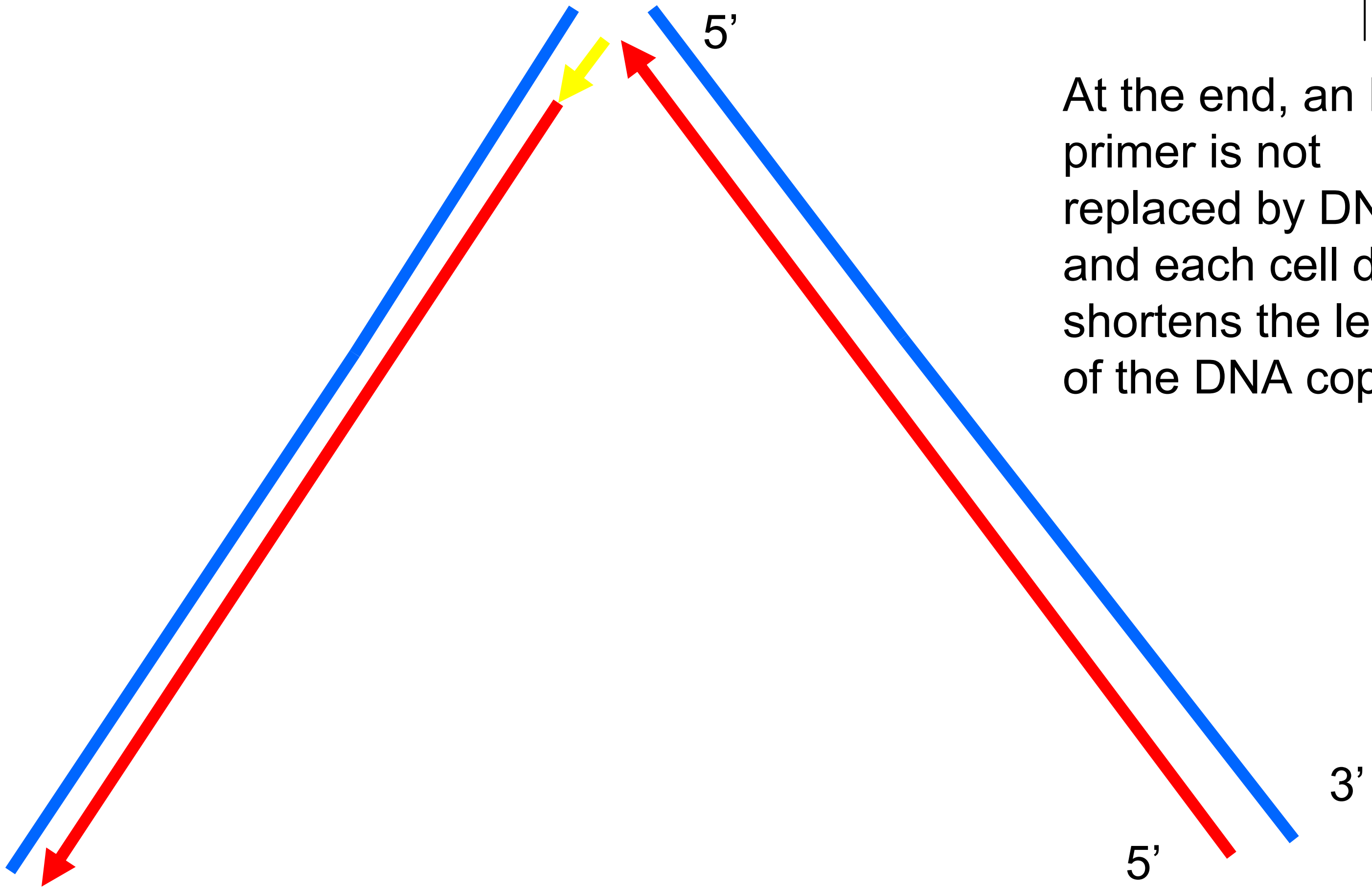
When the second fragment reaches the start of the first fragment, RNA blocks are replaced by DNA blocks

DNA Replication



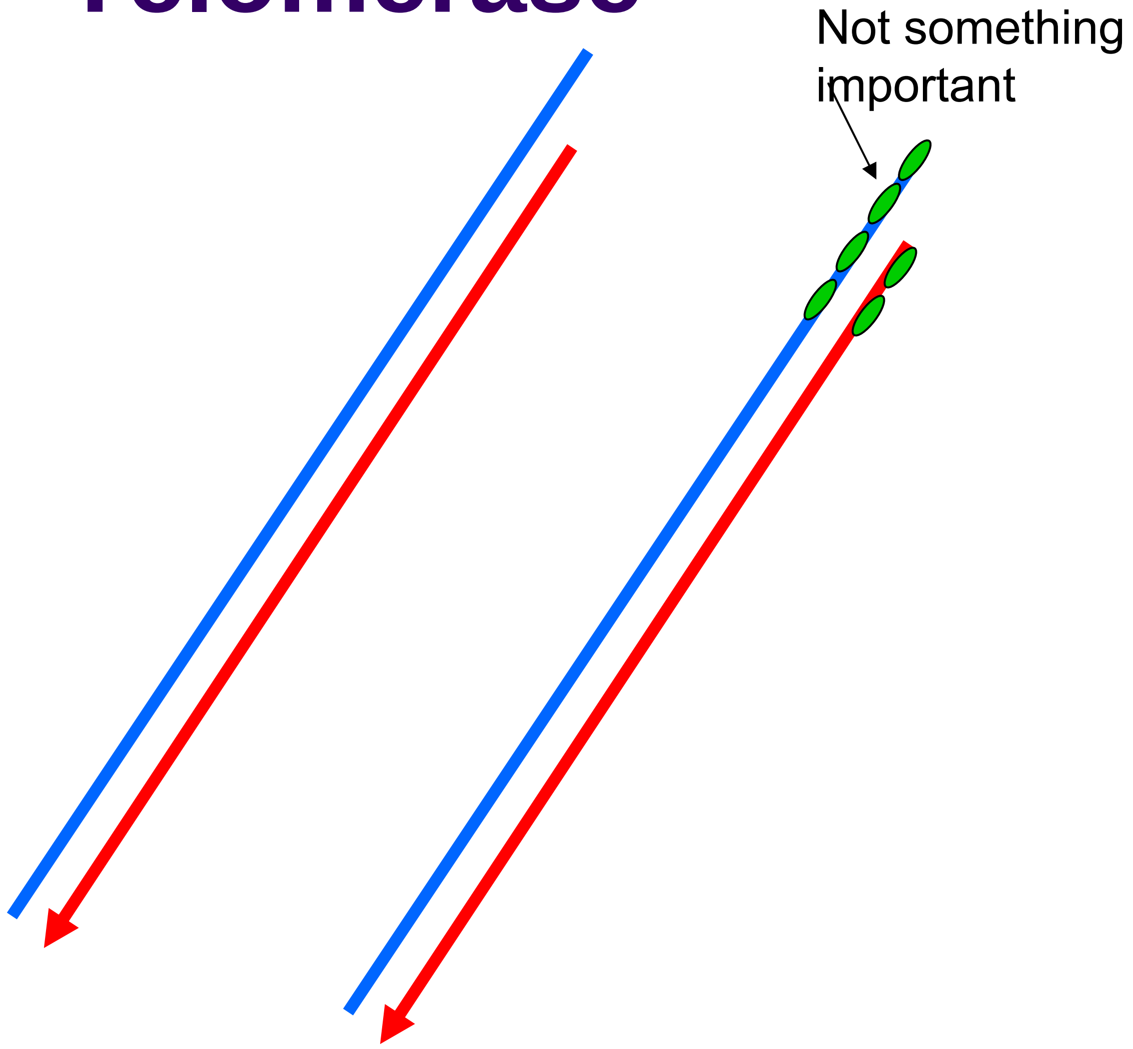
Special enzyme binds the DNA fragments together

DNA Replication



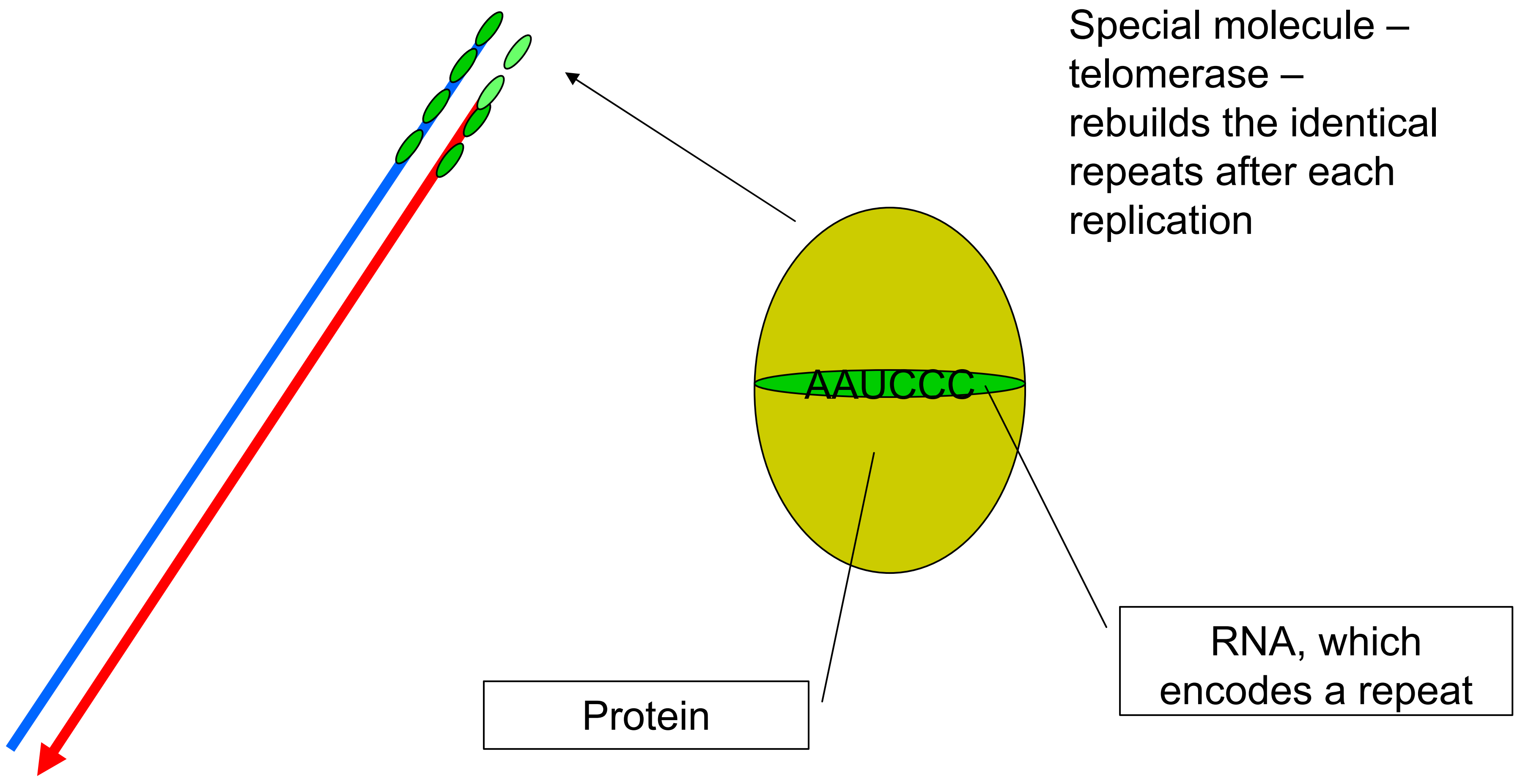
At the end, an RNA primer is not replaced by DNA, and each cell division shortens the length of the DNA copy

Telomerase



At the end of each chromosome there are no genes, but tandem repeats. For example, *TTAGGG* (Mammals)

Telomerase



Special molecule – telomerase – rebuilds the identical repeats after each replication

Telomerase

- Hypothesis: the activity of telomerase is decreasing with aging, and the chromosomes start shortening
- When the shortage reaches the encoding zone, organism dies
- To prevent aging we cannot just add telomerase, since it also promotes cancer
- The knock-out mice without telomerase within several generation become early-aging

Info

- Assignments – in pairs, preferable mixed pairs
- Projects list so far
- Suffix tree in linear time
- Book

Finding repeats

An efficient algorithm

Finding repeating substrings

- The path from the root to any internal node of the suffix tree represents a substring of *T* which occurs at least twice in *T*, since it corresponds to a common prefix of at least 2 different suffixes
- Thus, all repeating substrings can be found by collecting the internal nodes of the suffix tree during the depth-first traversal

DFS Pseudocode

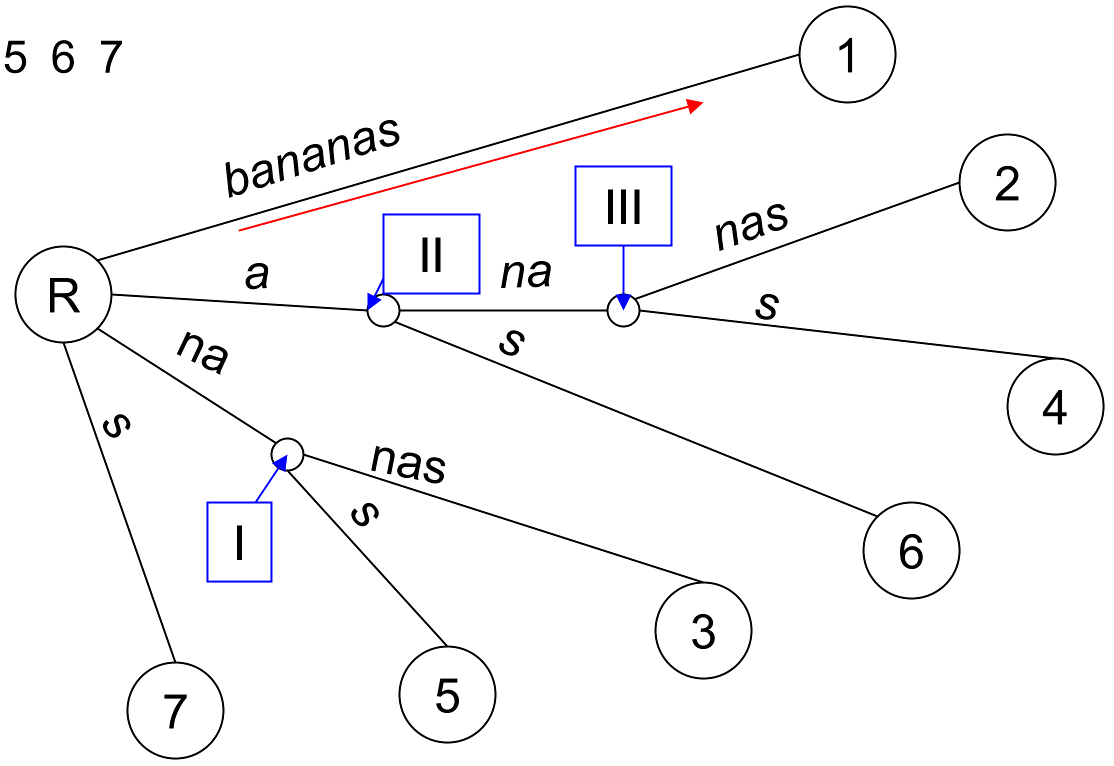
```
algorithm DFS (Node root)
        traverse (root)

algorithm traverse (Node parent)
        for each node in children of parent
                traverse (node)
```

In other words: for each child in order traverse tree in depth, until the node without children is reached. Then proceed to the next child.

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

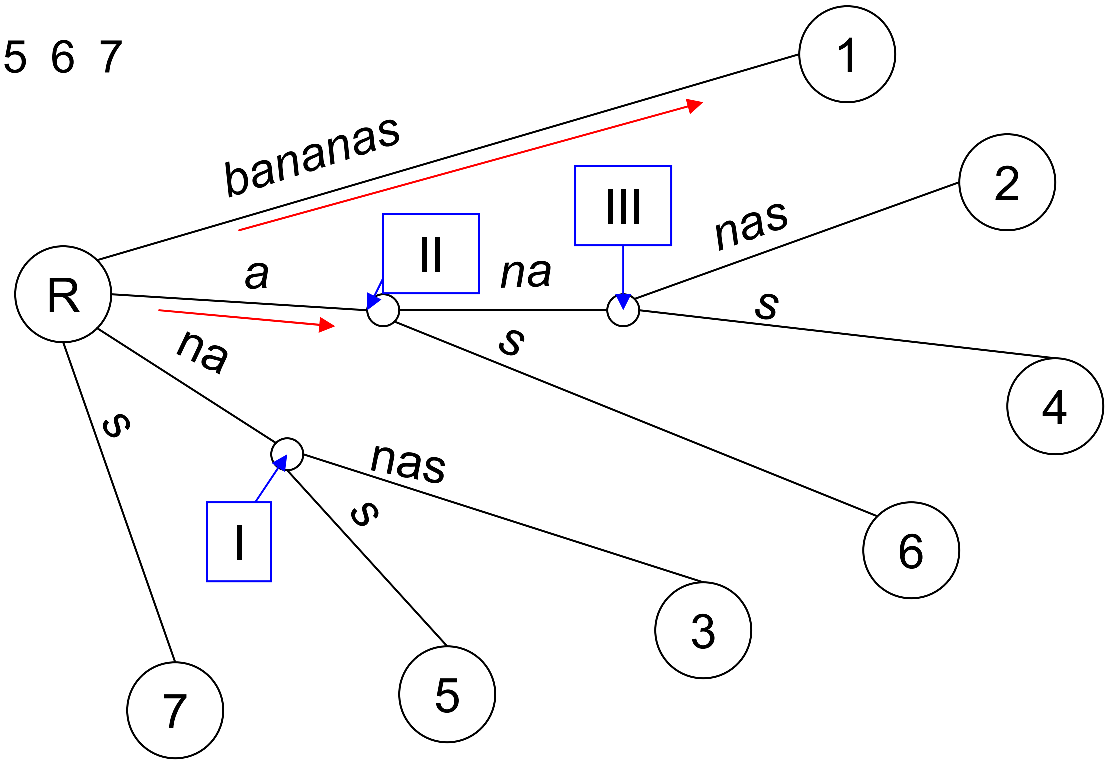


Sequence: R 1 R

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

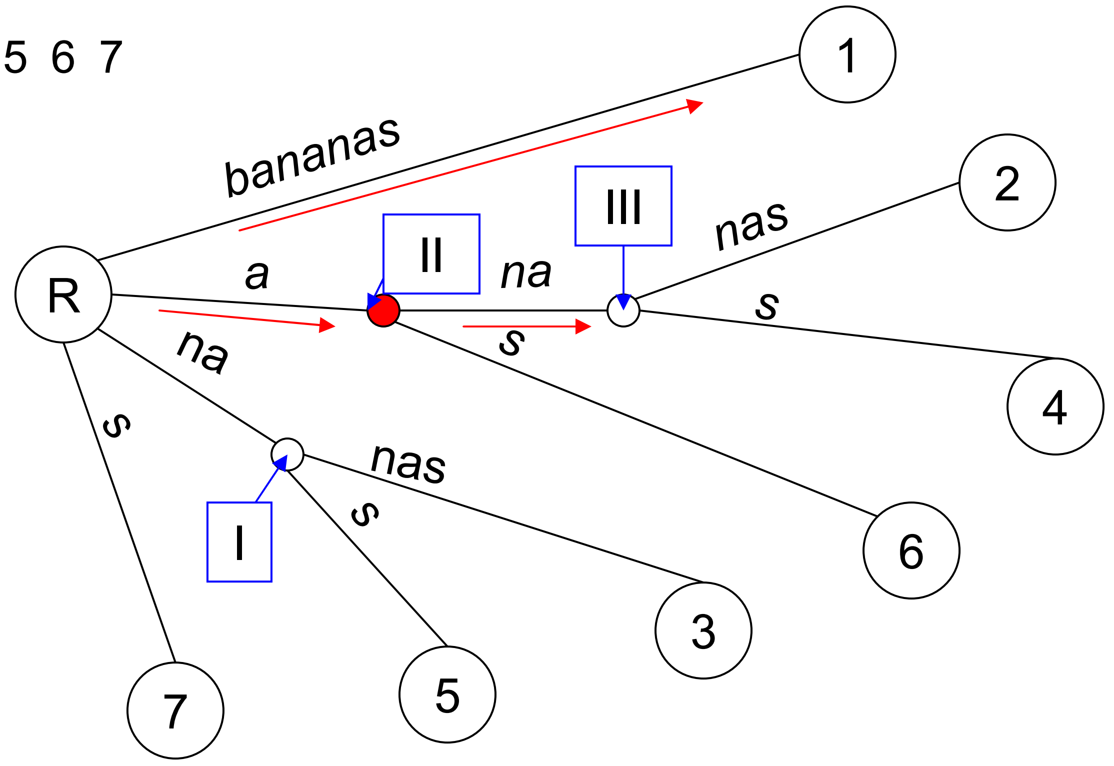


Sequence: R 1 R II

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

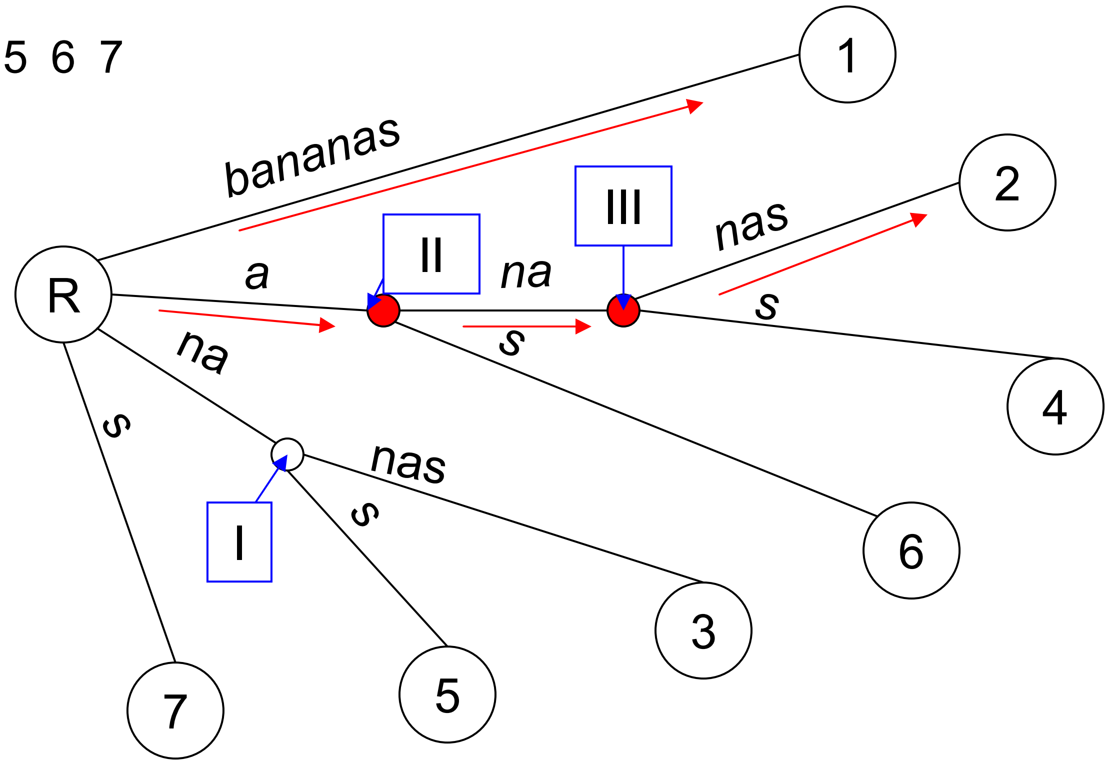


Sequence: R 1 R II III

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

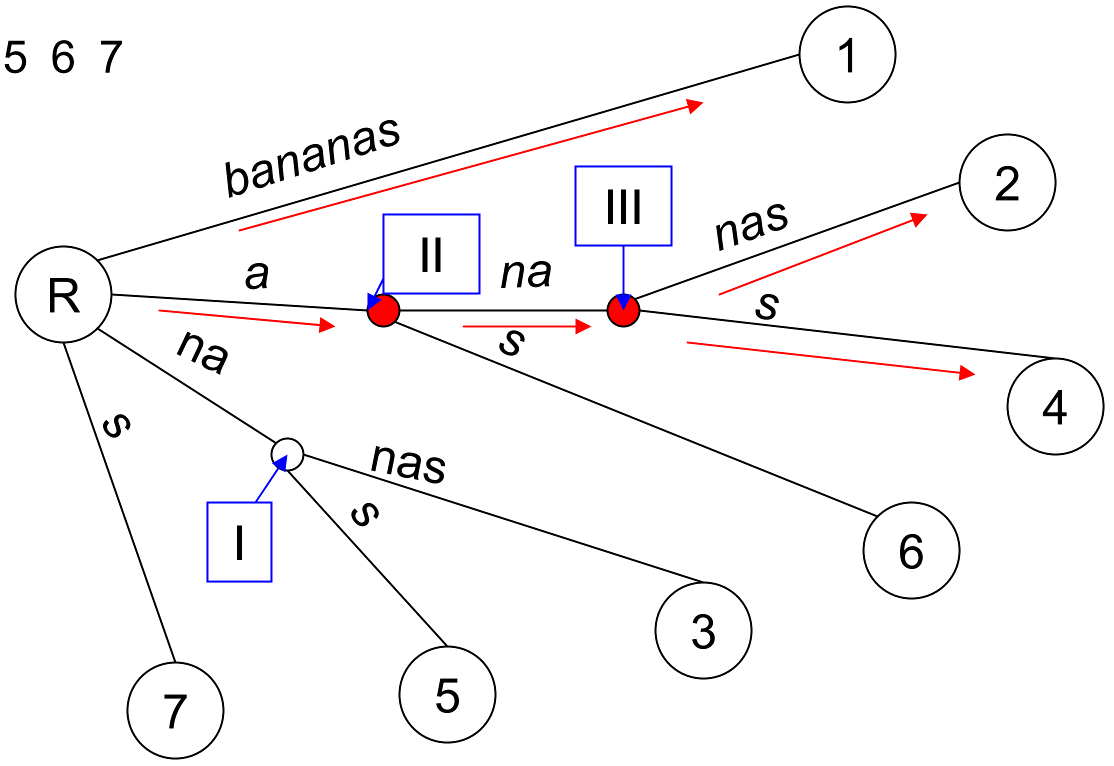


Sequence: R 1 R II III 2 III

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

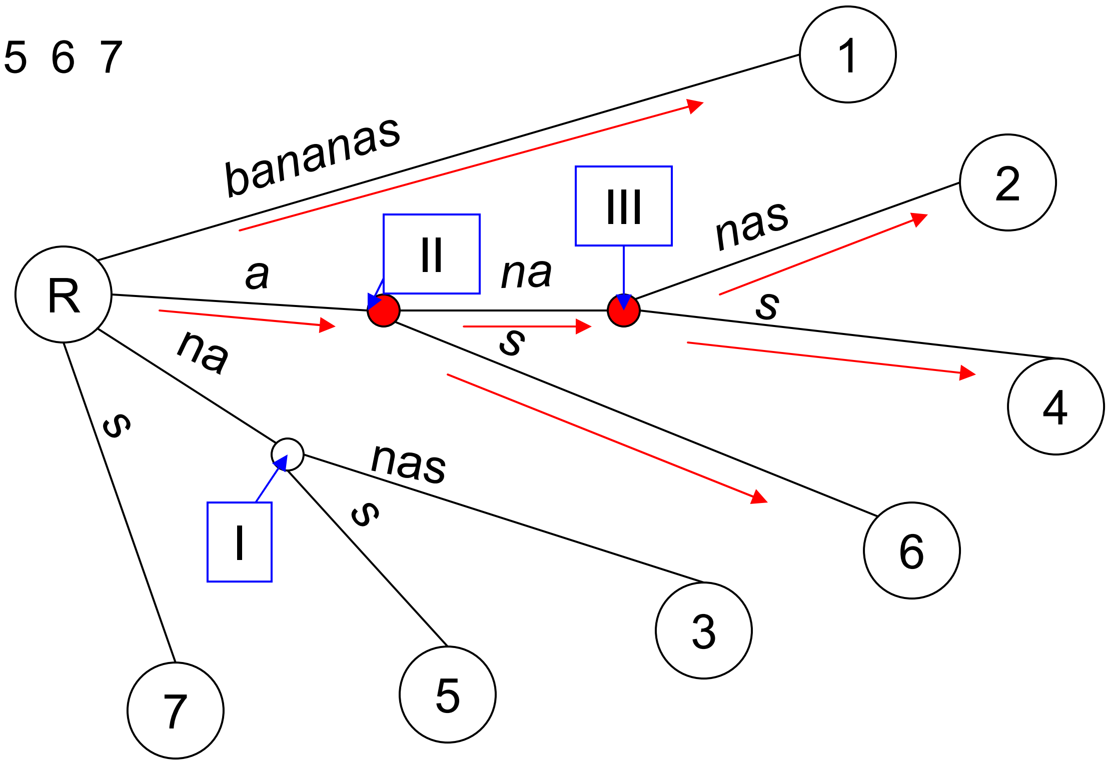


Sequence: R 1 R II III 2 III 4 III

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

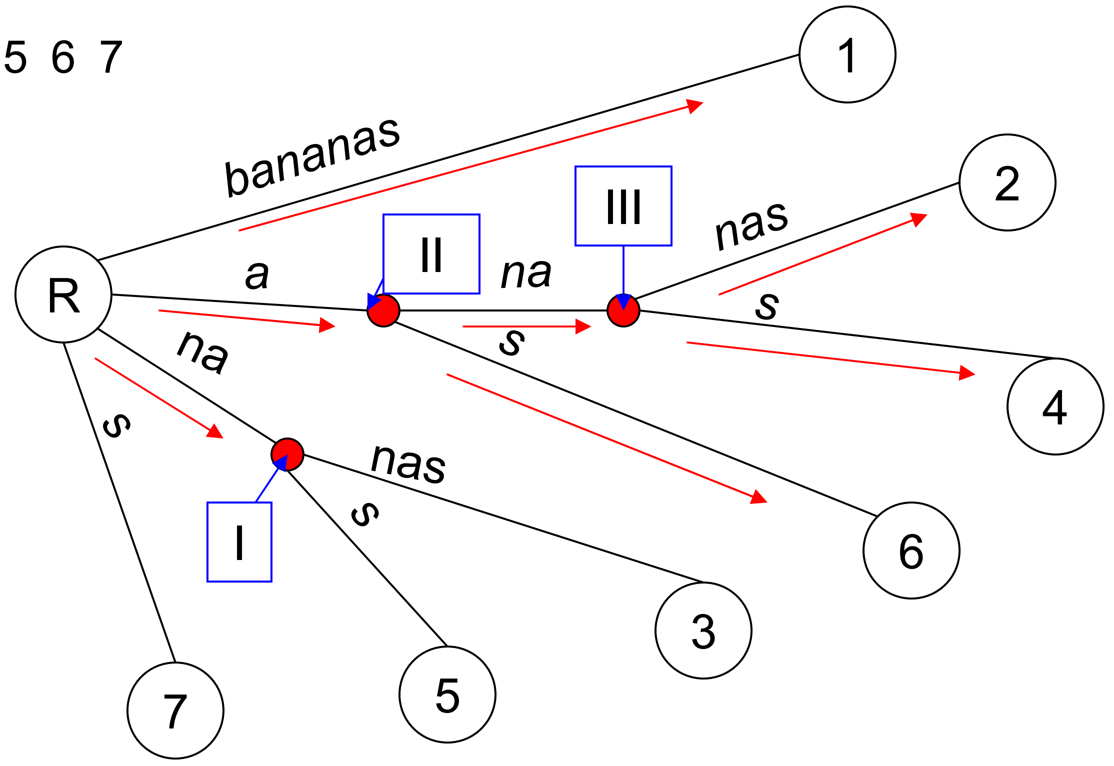


Sequence: R 1 R II III 2 III 4 III II 8 II R

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

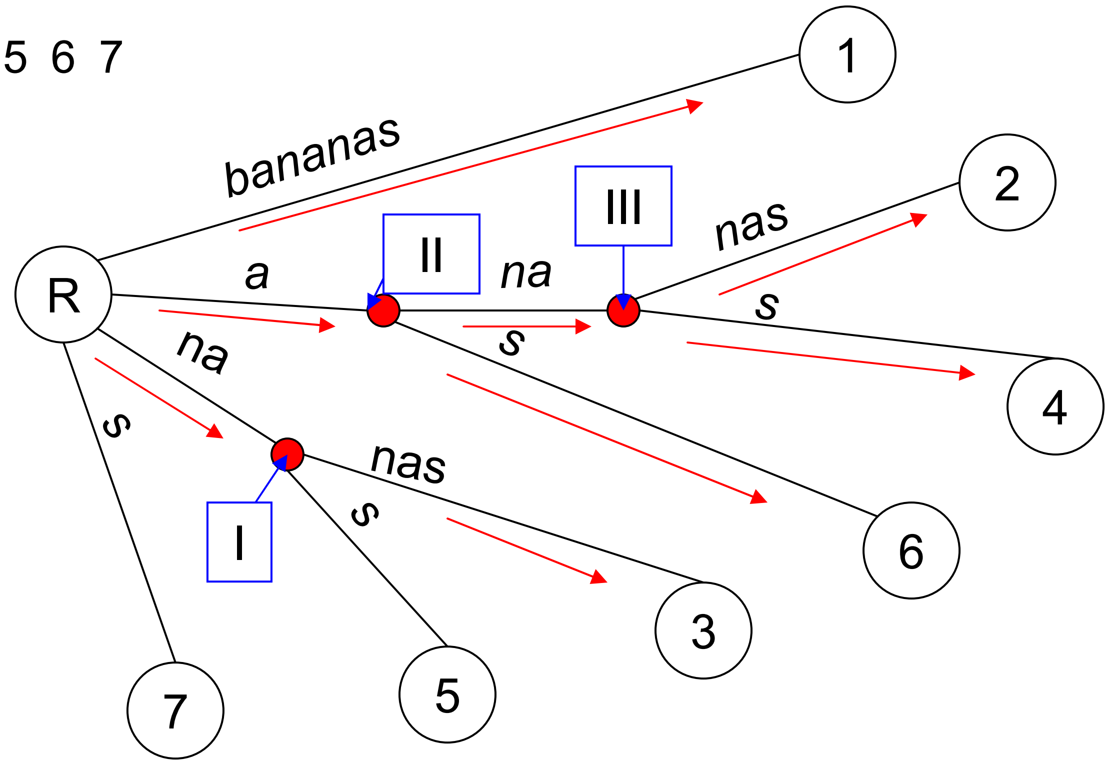


Sequence: R 1 R II III 2 III 4 III II 8 II R I

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

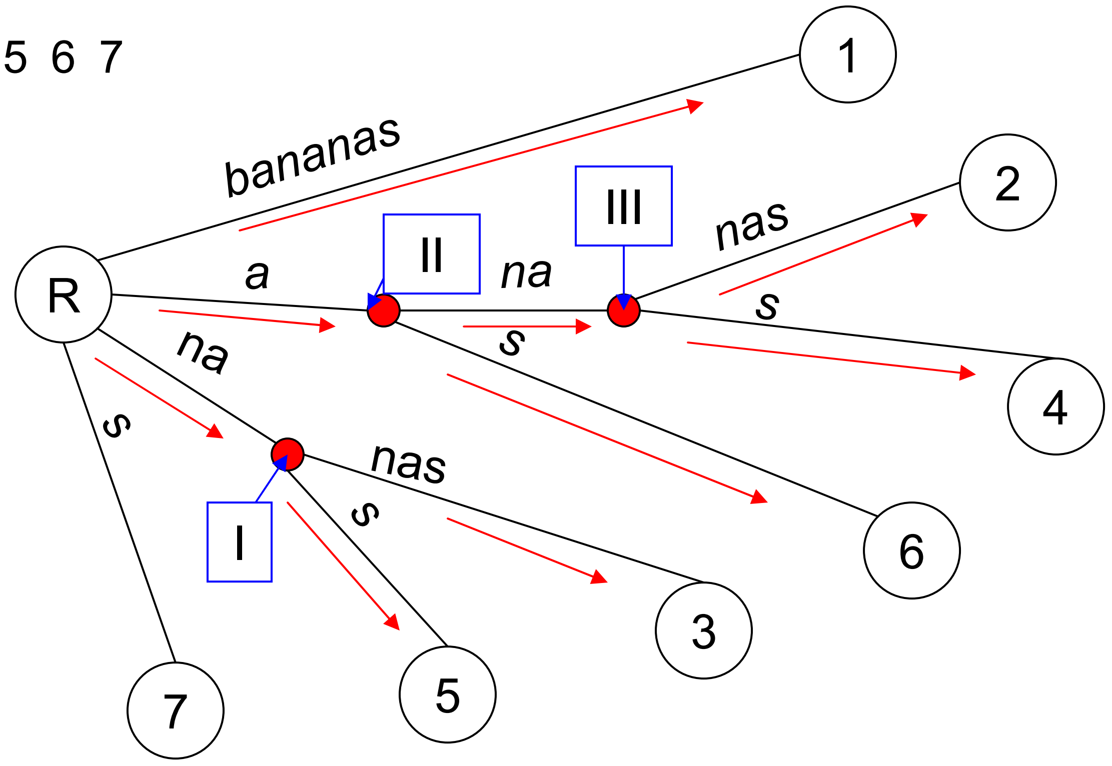


Sequence: R 1 R II III 2 III 4 III II 8 II R I 3 I

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

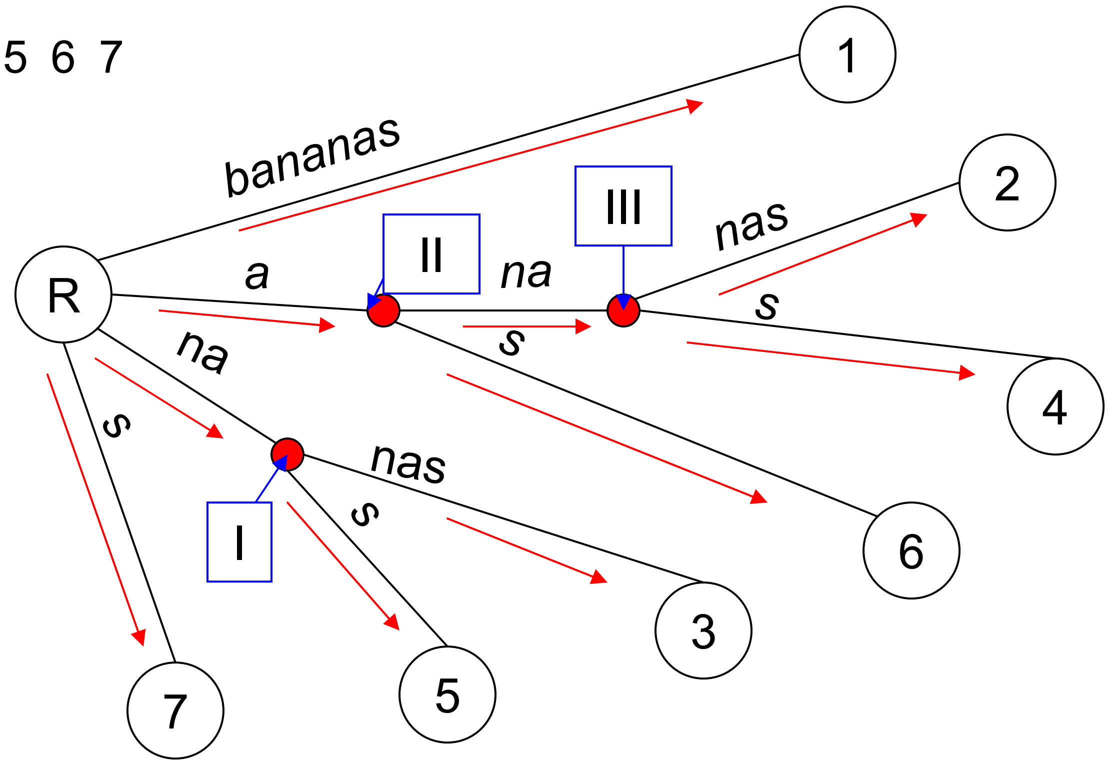


Sequence: R 1 R II III 2 III 4 III II 8 II R I 3 I 5 I R

The depth-first traversal

b a n a n a s

1 2 3 4 5 6 7

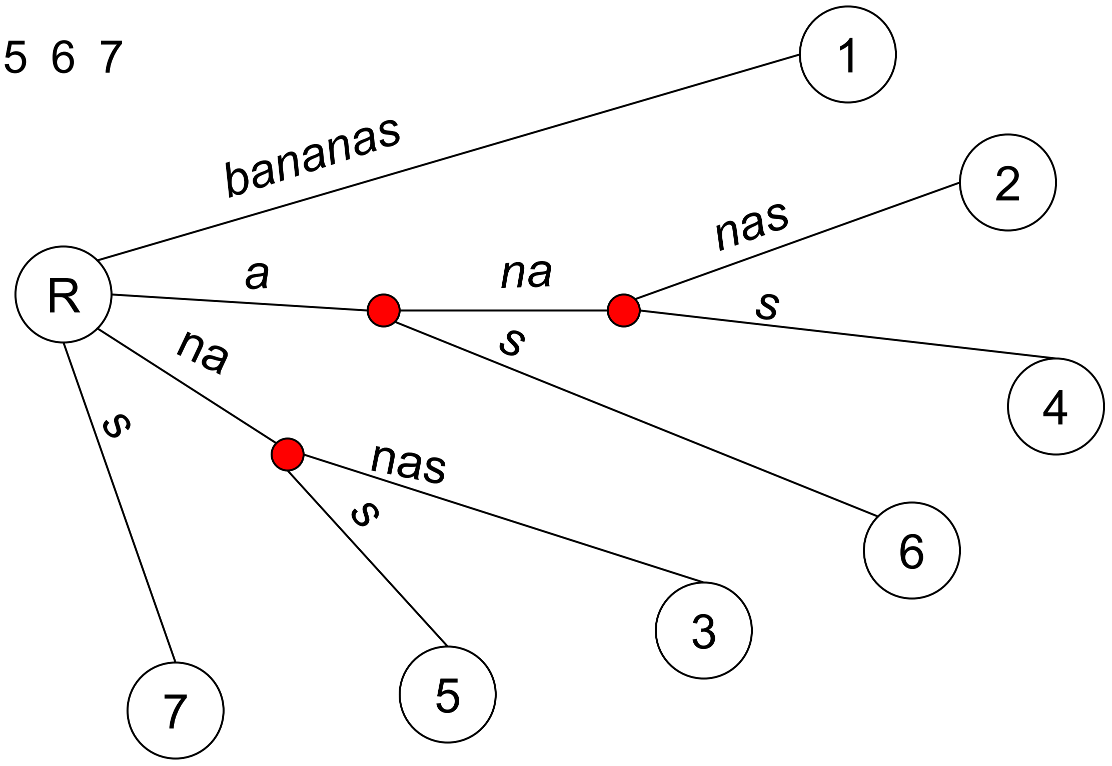


Sequence: R 1 R II III 2 III 4 III II 8 II R I 3 I 5 I R 7 R

All repetitions

b a n a n a s

1 2 3 4 5 6 7

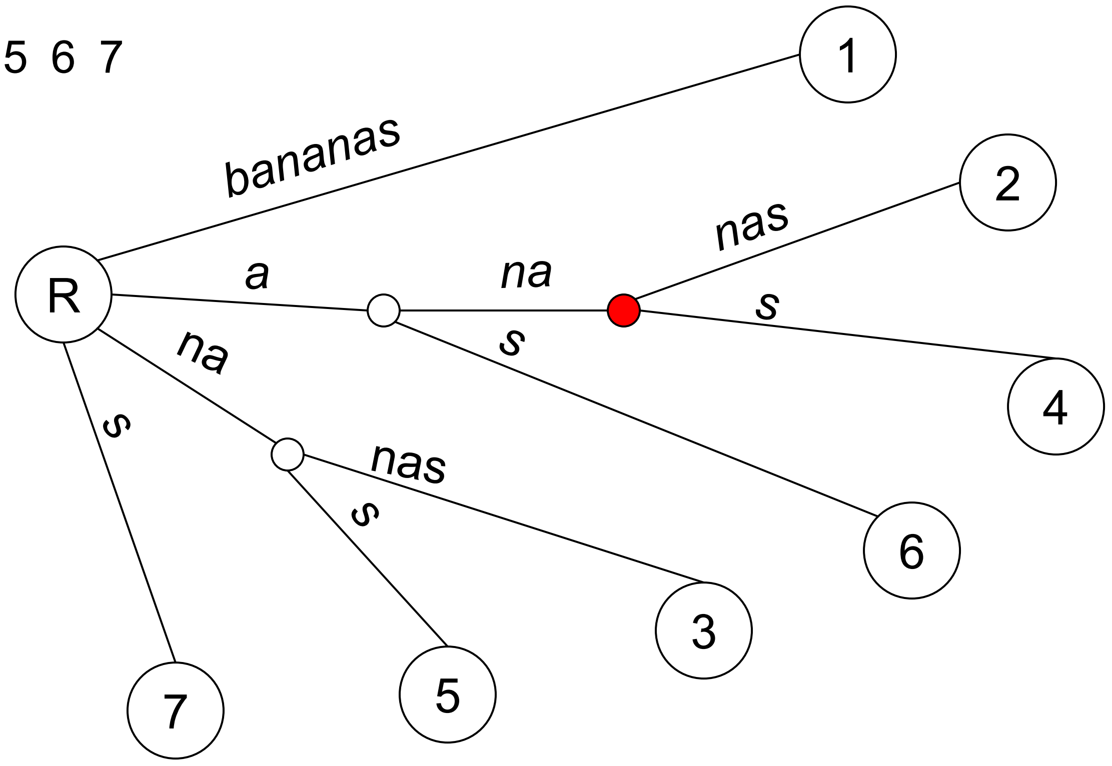


n, na; a, an, ana;

The longest repeating substring in linear time

b a n a n a s

1 2 3 4 5 6 7



ana

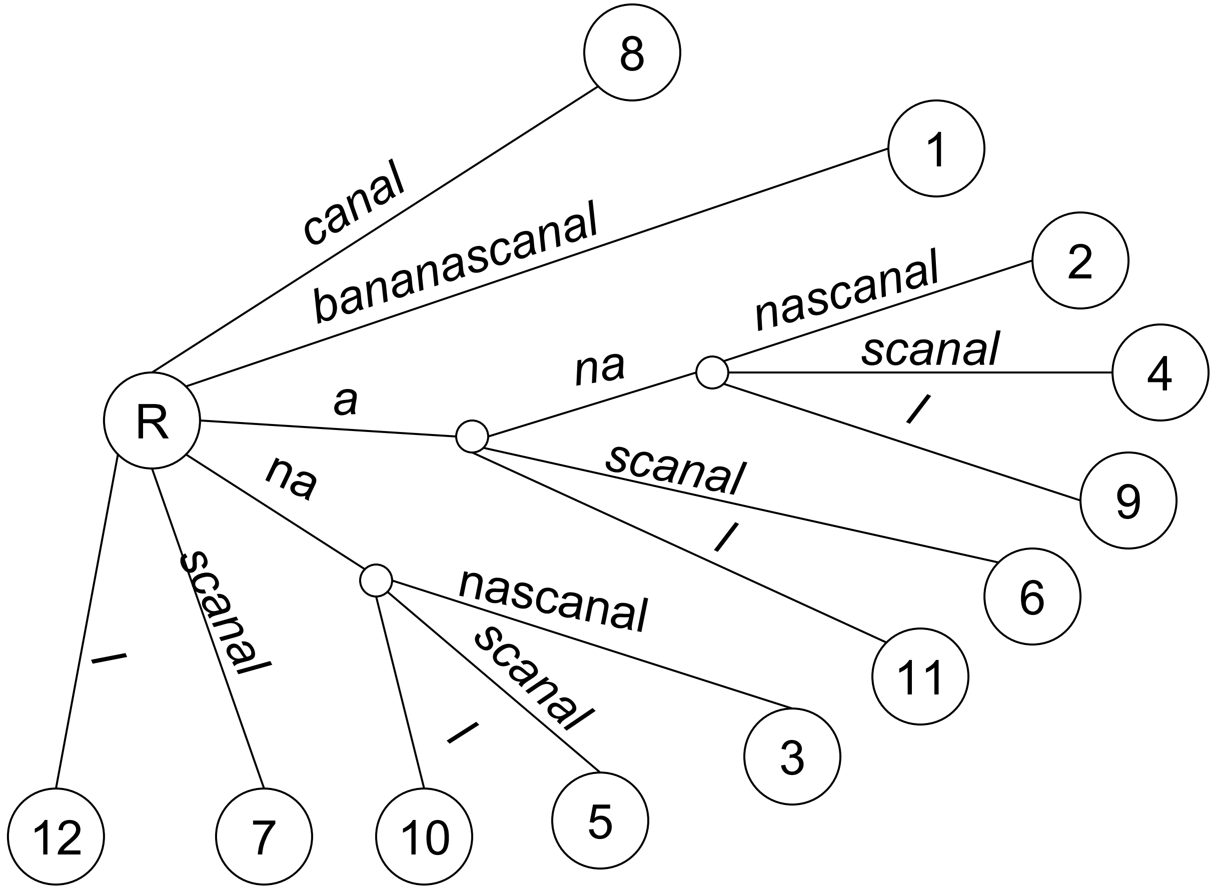
The longest common substring of several strings

- The problem: find the longest substring common to two given strings I and II.
  - For example, if I=*superiorcalifornialives* and II=*sealiver*, then the longest common substring of I and II is *alive*.
- Used in the identification of the remains of US military personnel
  - Mitochondrial DNA from live person is collected, sequenced and the sequences are stored in the database (I)
  - Later, the DNA is extracted from the remains (II), and the longest common substring of I and II helps to narrow down the search
- 1970 – Knuth conjectured that the linear-time solution to the longest common substring problem would be impossible

The longest common substring for 2 strings in linear time

- Concatenate 2 strings and build the suffix tree for the concatenated string
- Label each leaf with the corresponding suffix start position, plus the ID of the string (I or II)
- Perform the depth-first traversal and mark each internal node by I, II or both, depending what suffixes are found in the subtree for this node
- Find the deepest internal node which is marked by both I and II

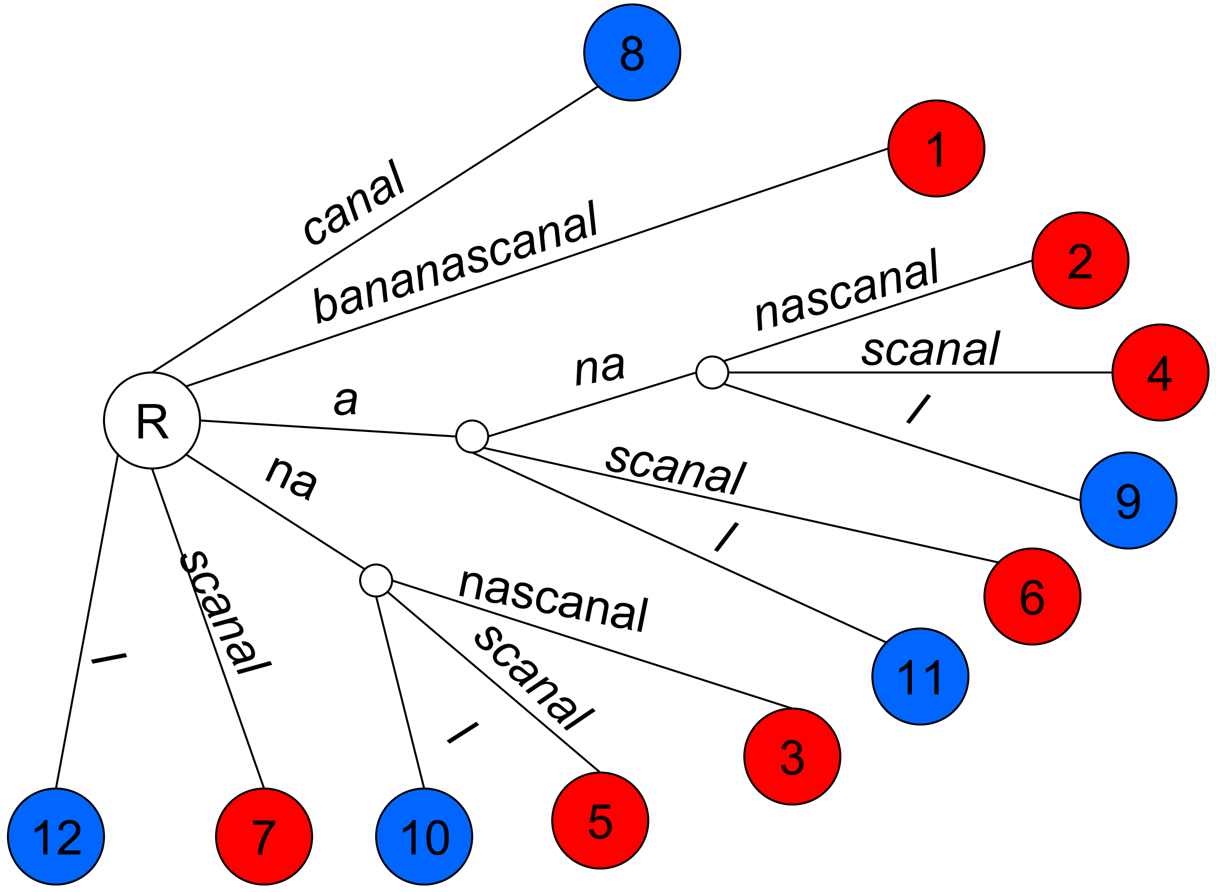
Example: *I=bananas II=canal*



b a n a n a s c a n a l

b	*a*	*n*	*a*	*n*	*a*	*s*	*c*	*a*	*n*	*a*	*l*
1	2	3	4	5	6	7	8	9	10	11	12

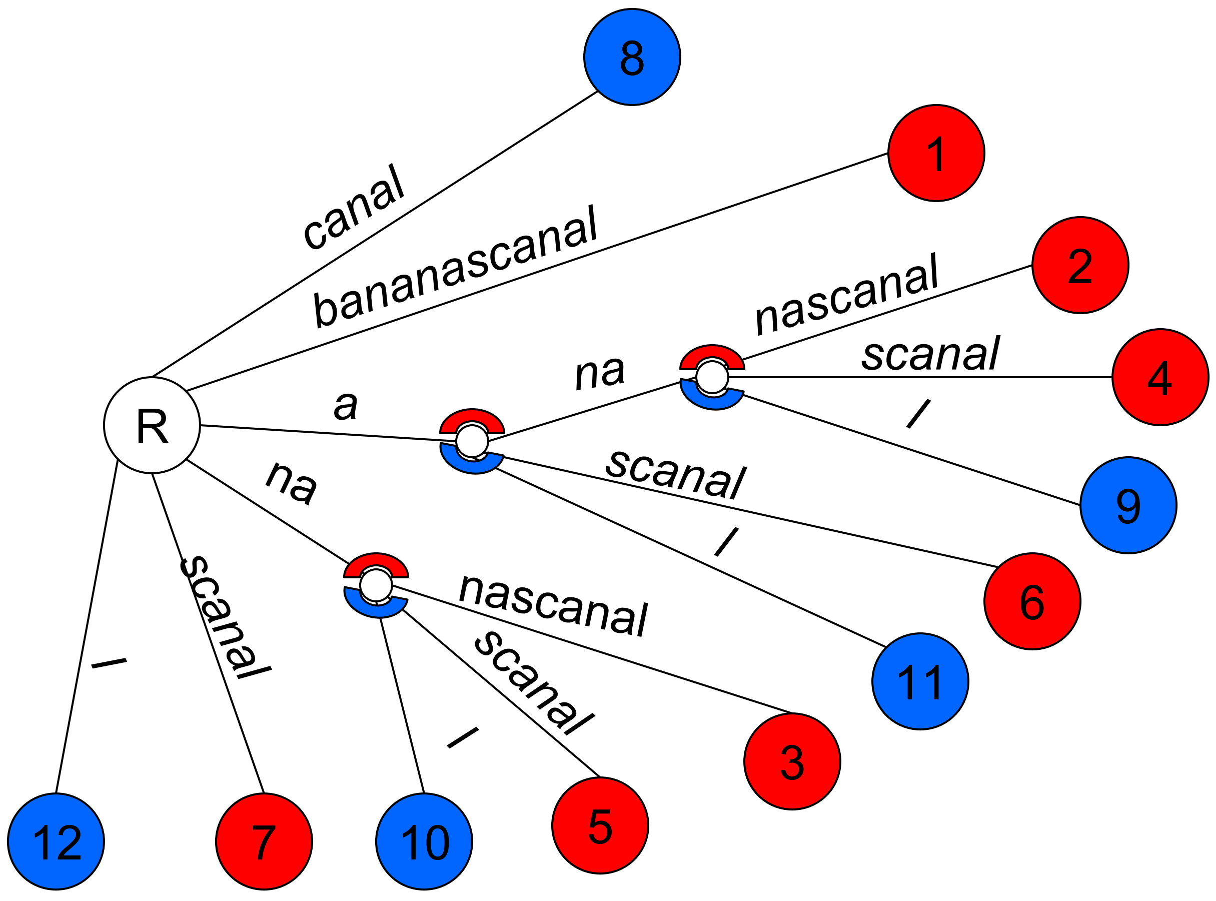
Example: *I=bananas II=canal*



b a n a n a s c a n a l

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

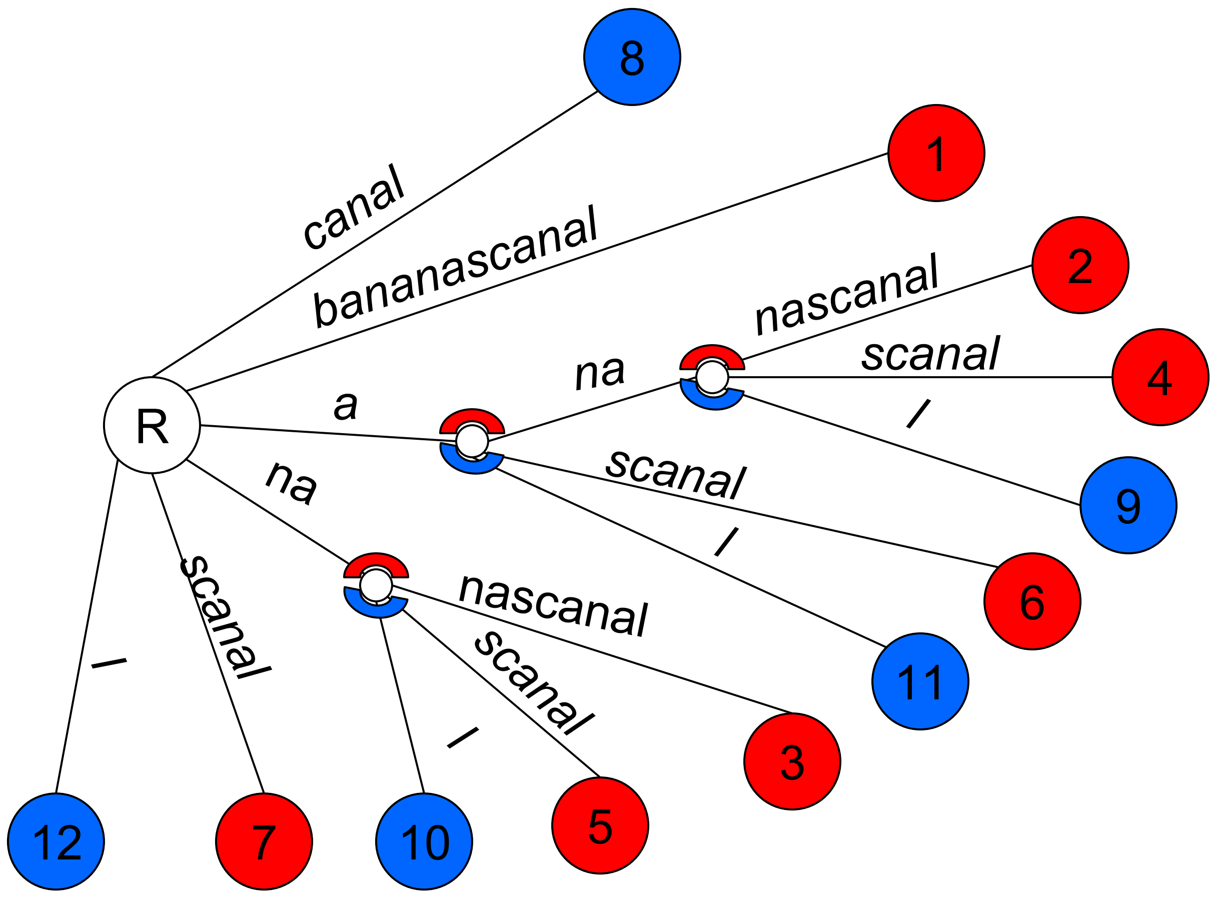
Example: Marking internal nodes



b a n a n a s c a n a l

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

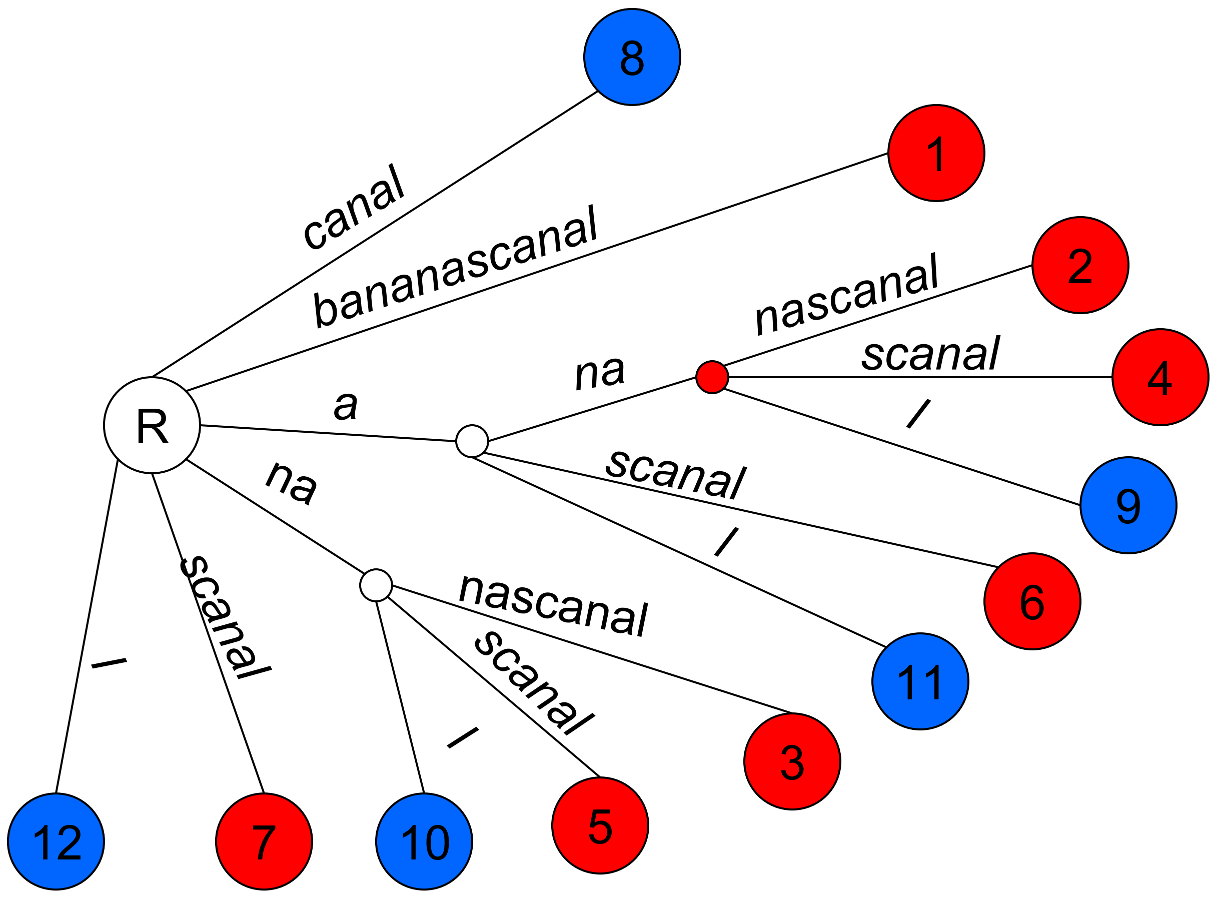
Example: What is the longest common substring ?



b a n a n a s c a n a l

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

Example: LCS=*ana*



b a n a n a s c a n a l

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

All maximal repeats

- Definition: *A maximal repeated pair* (MR) in a string T is a pair of identical substrings t_1 and t_2 such that the character to the immediate right (left) of t_1 is different from the character to the immediate right (left) of t_2. Each MR pair can be represented by a tuple (i,j,k), where i and j are start positions of the corresponding substrings, and k is the substring length
- If the characters to the right of t_1 and t_2 are different, we will call such repeat *right- maximal* (cannot be extended to the right).
- If the characters to the left of t_1 and t_2 are different, we will call such repeat *left- maximal* (cannot be extended to the left).

Maximal repeats example

- $T=x\overset{\overset{2}{\downarrow}}{a}bcyiiiz\overset{\overset{10}{\downarrow}}{a}bcq\overset{\overset{14}{\downarrow}}{a}bcyrxar$
- Which of the following repeated pairs of length 3 are maximal repeats?
  - *(2,10)*
  - *(2,14)*
  - *(10,14)*

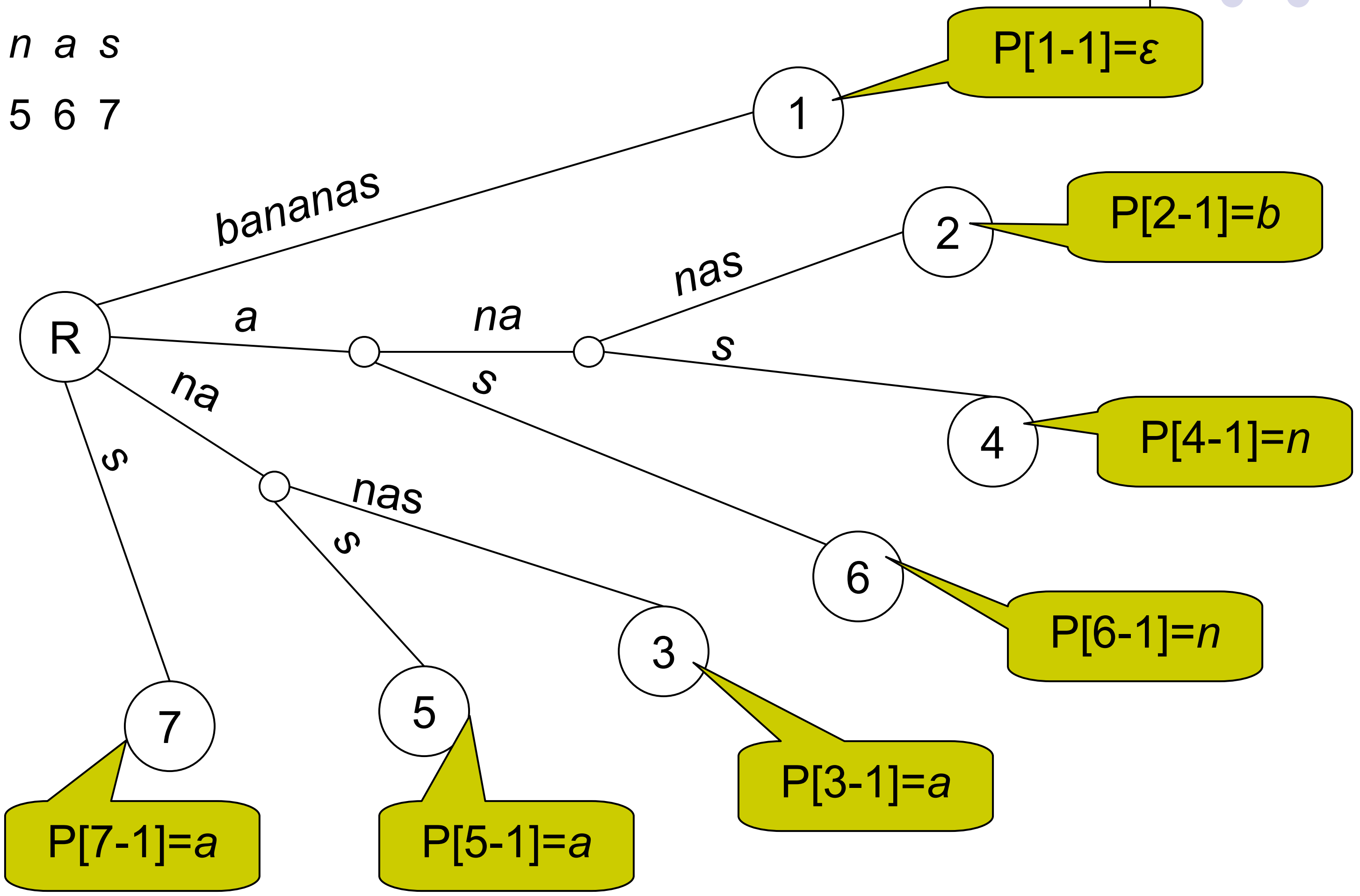
An efficient algorithm for finding all maximal repeats

- The substring labeling the path to any internal node of the suffix tree always represents a right-maximal pair (Why?)
- Each such substring represents a prefix of some pair of suffixes *T[i…N]* and *T[j…N]*. In order to check for each such substring if it is also a left-maximal repeat, we need only to check if the characters at positions *T[i-1]* and *T[j-1]* are different.
- This can be done in a linear time.

Step 1. Mark leaves with the left character

b a n a n a s

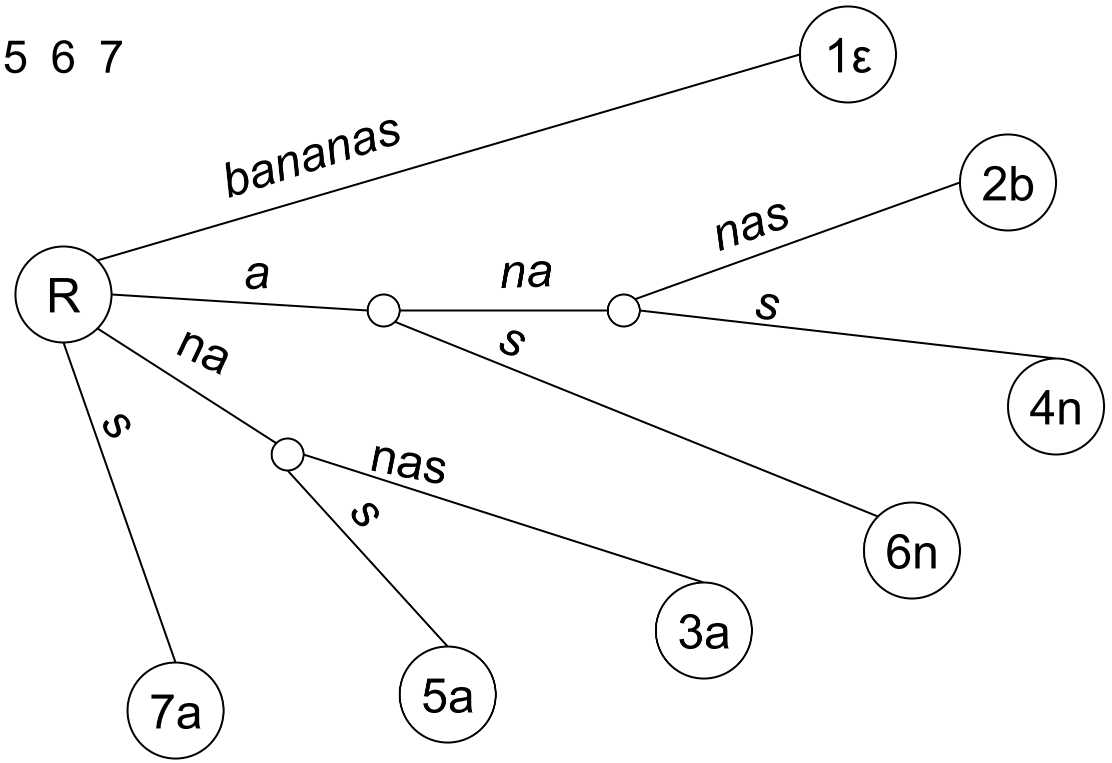
1 2 3 4 5 6 7



Step 2. Traverse

b a n a n a s

1 2 3 4 5 6 7



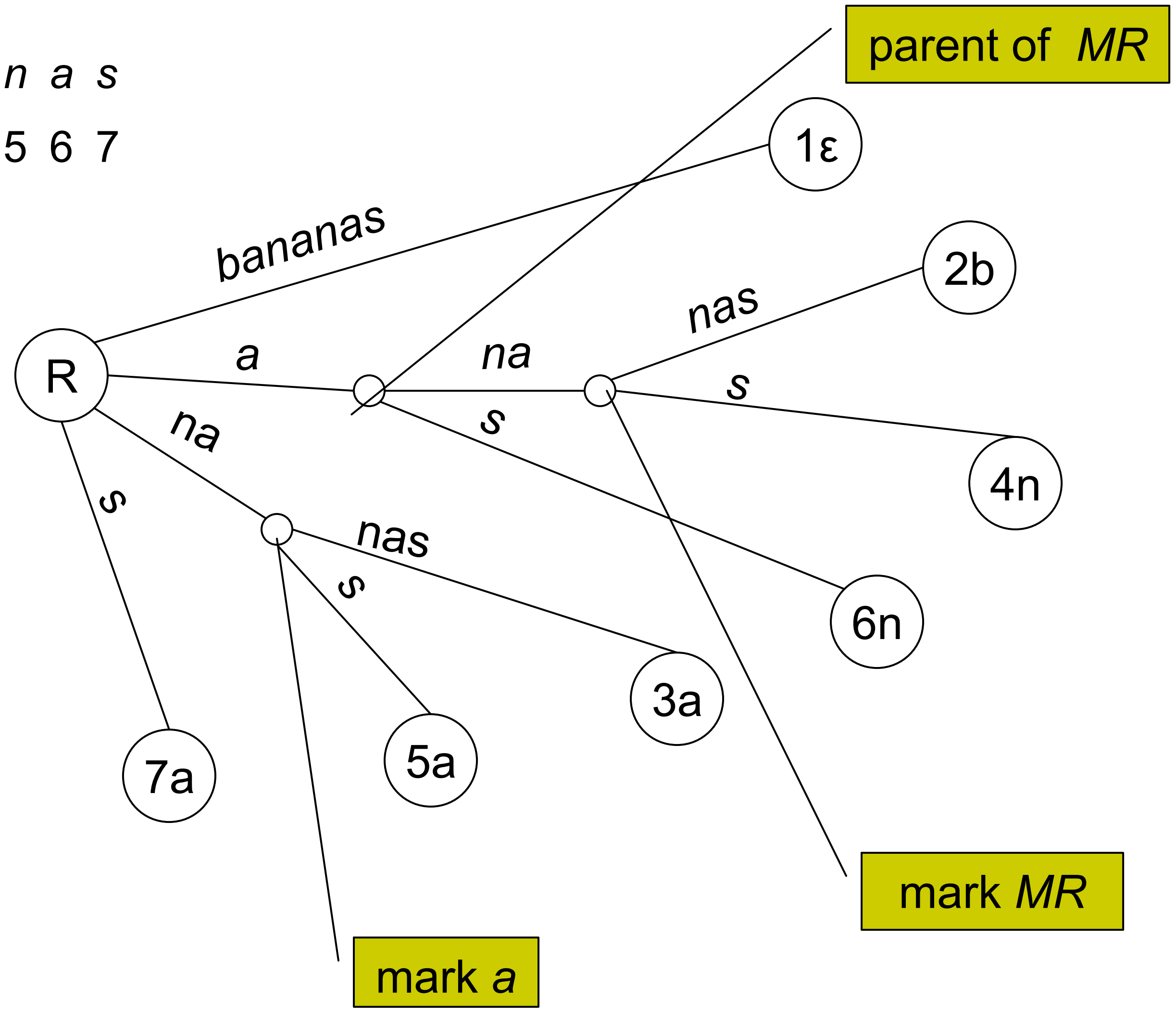
If both children of an internal node have the same character to the left of the suffix, then mark this internal node with this character.

If the left characters are different, then the path from the root to this node represents a maximal repeat, so mark the node as maximal repeat

Step 2. Mark internal nodes

b a n a n a s

1 2 3 4 5 6 7

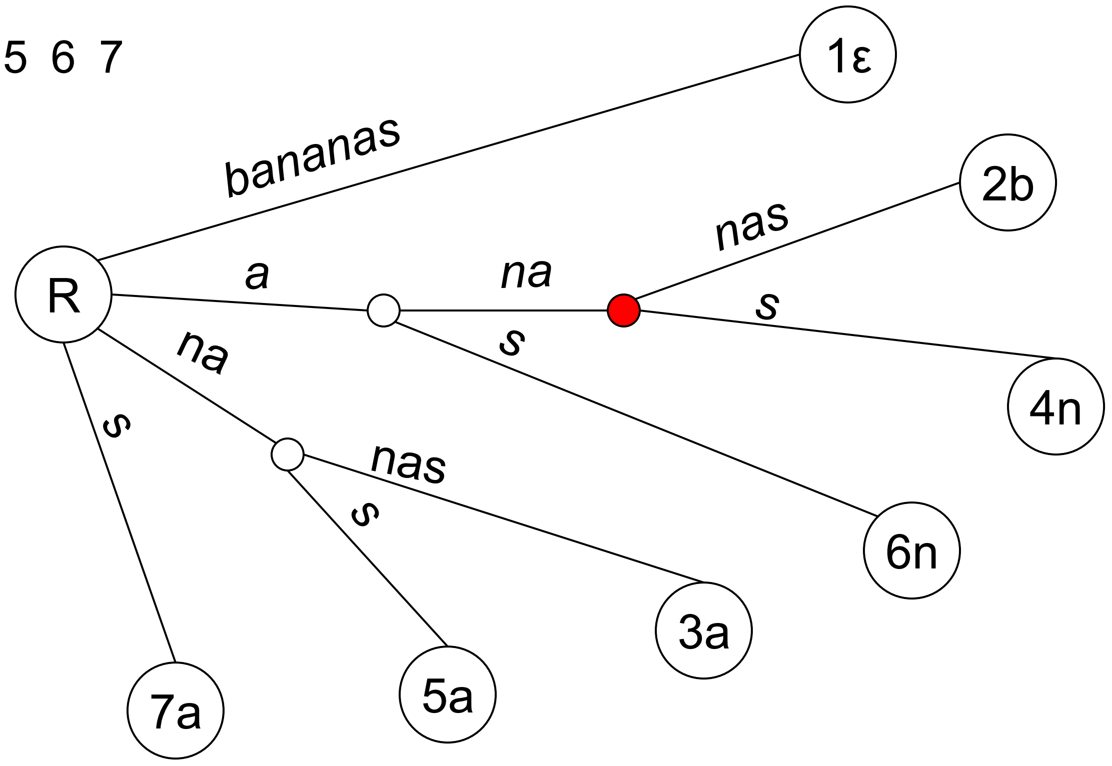


The parent of maximal repeat is a maximal repeat too (why?)

Step 3. Output

b a n a n a s

1 2 3 4 5 6 7



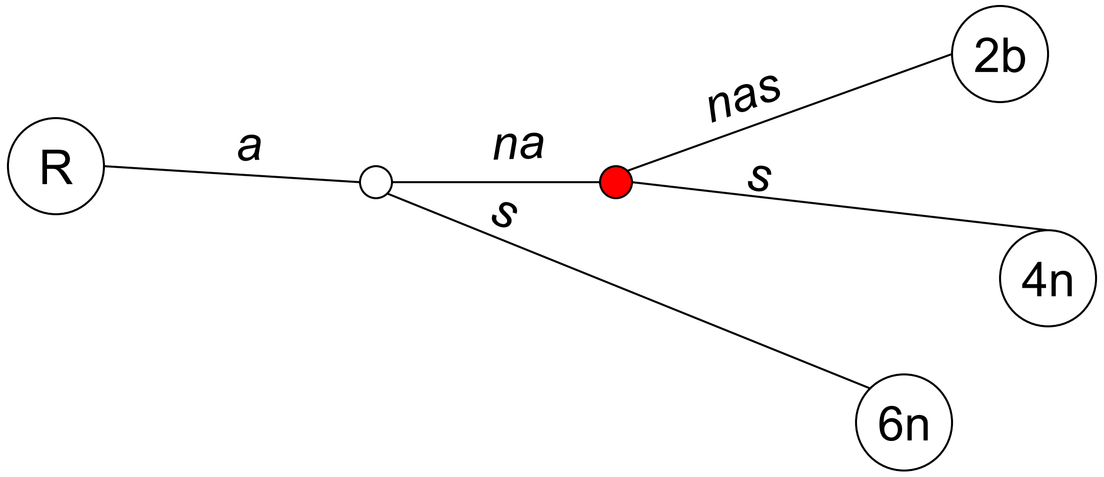
Maximal repeat is *ana* (2,4)

Maximal repeat is also *a* (2,6)

Step 3. Output

b a n a n a s

1 2 3 4 5 6 7



There can be up to *N* maximal repeats in any string (why?)

These maximal repeats can be efficiently represented in a linear space using the same suffix tree with the nodes corresponding to repeats only