Finding Optimal Pairs of Cooperative and Competing Patterns with Bounded Distance

Shunsuke Inenaga[1], Hideo Bannai[2], Heikki Hyyrö[4], Ayumi Shinohara[3,4], Masayuki Takeda[3,5], Kenta Nakai[2], and Satoru Miyano[2]

[1]Department of Computer Science, University of Helsinki, Finland
[2]Human Genome Center, Institute of Medical Science, University of Tokyo, Japan
[3]Department of Informatics, Kyushu University, Japan
[4]PRESTO, Japan Science and Technology Agency (JST)
[5]SORST, Japan Science and Technology Agency (JST)

Pattern Discovery

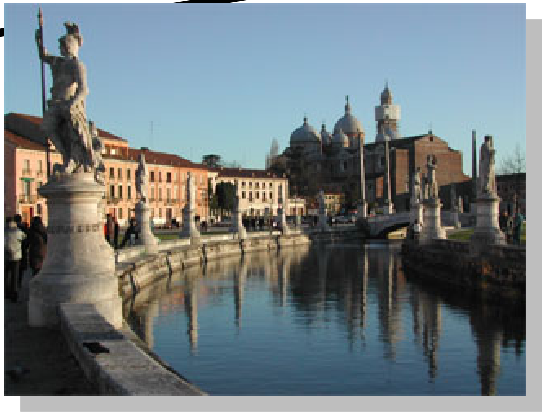
Input : set S of strings $s_1, s_2, \ldots, s_m$

Output : pattern p that characterizes input S

S

- Pattern discovery is a core of Discovery Science.
- Jump and dive into the sea during summer vacation.

p = **padova**



Pattern Discovery in Bioinformatics

Machine Discovery System BONSAI

(Shimozono et al. 1994)

- finds optimal substring patterns to characterize input sequence set S.
- has been extended to finding:

subsequence patterns (Hirao et al. 2000)
episode patterns (Hirao et al. 2001)
VLDC patterns (Inenaga et al. 2002)
approximate FVLDC patterns (Takeda et al. 2003)

Refer to the invited talk by Ayumi Shinohara in DS'04
"String Pattern Discovery"

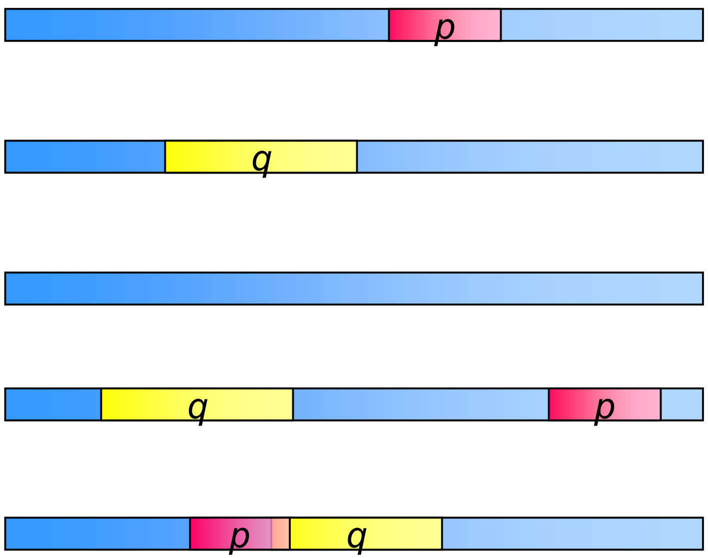
Composite Pattern Discovery

- More than one sequence element may act in ensemble!
- Sequence elements may not necessarily be **cooperative** – they may be **competitive**.
- Consider finding **Boolean combination** of patterns.

e.g.: $p \wedge q$: p AND q
$p \vee q$: p OR q
$p \wedge \neg q$: p AND (NOT q)

Examples

$S = \{s_1, s_2, s_3, s_4, s_5\}$



	p	q	p OR q	p AND (NOT q)
s_1	true	false	true	true
s_2	false	true	true	false
s_3	false	false	false	false
s_4	true	true	true	false
s_5	true	true	true	false

substrings: exact match only (no mismatches)

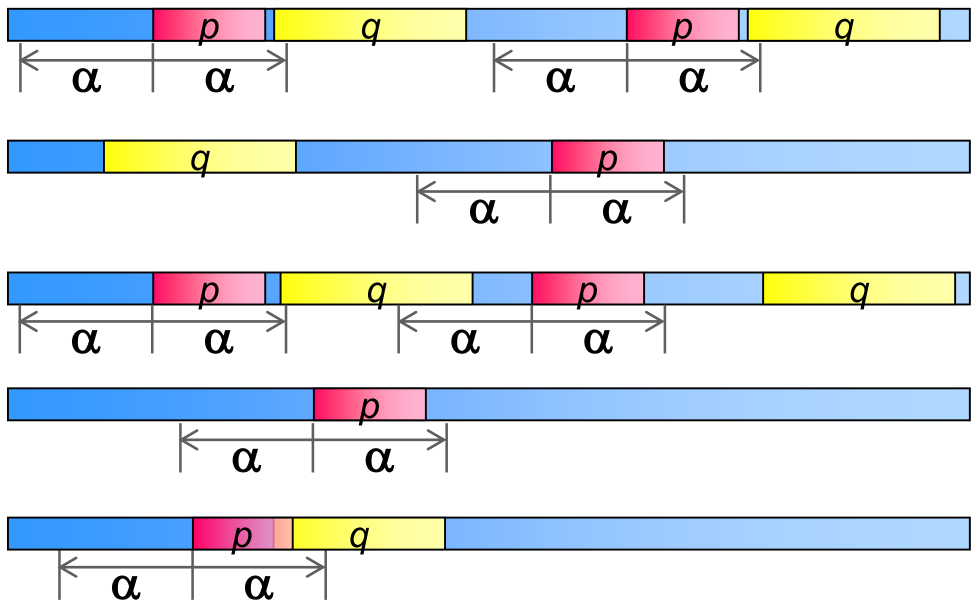
Recent Work by Bannai et al.

"Finding Optimal Pairs of Patterns"
(Bannai et al. in WABI'04)

- Efficient algorithm for finding optimal Boolean pattern pair of substring patterns
- $O(N^2)$ time & $O(N)$ space
  (N: total length of strings in input set ***S***)
- $O(N^k)$ time & $O(N)$ space
  for Boolean combination of k substring patterns

And This Work…

- Introduces the notion of **α-distance** between patterns in Boolean pair w.r.t. $\wedge$ (AND).
- Denoted $p \wedge_\alpha q$ and $p \wedge_\alpha \neg q$.



	$p \wedge_\alpha q$	$p \wedge_\alpha \neg q$
s_1	true	false
s_2	false	true
s_3	true	false
s_4	false	true
s_5	true	false

Our Result

Input : set S of strings and distance α

Output : optimal pattern pair $p \wedge_\alpha q$ and $p \wedge_\alpha \neg q$ w.r.t. S

- $O(N^2)$ time & $O(N)$ space (not depending on α)
- $O(N^k)$ time & $O(N)$ space for combination of k substring patterns
  - Improves the worst case complexity $O(\alpha^k N^{k+1} \log N)$ due to Arimura et al. (proximity patterns).

Optimality of Pattern Pair

Pattern pair π is optimal w.r.t. $\boldsymbol{S}$

$$\Updownarrow$$

Pattern pair π maximizes $score(M(\pi, \boldsymbol{S}))$

Examples of *score* function:

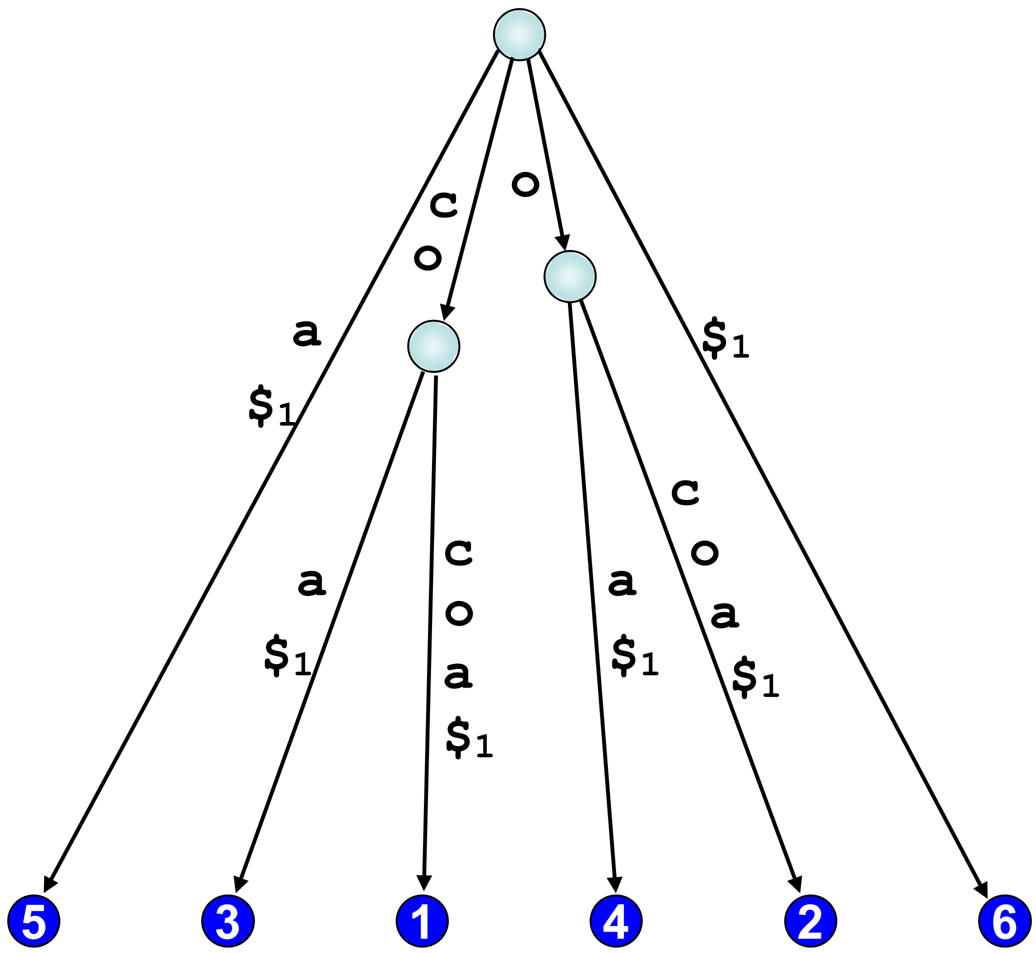
- Gini index
- Chi-square statistic
- Rank-sum test

$M(\pi, \boldsymbol{S})$: num. of strings in $\boldsymbol{S}$ that π matches.
Assume *score* can be computed in O(1) time.

Suffix Tree

Suffix Tree of *s* (*ST*(*s*))

- Tree structure which represents all suffixes of *s*.
- Each leaf is marked with its suffix number.
- *ST*(*s*) can be constructed in *O*(*n*) time (*n* = |*s*|).
  (Weiner 1973, McReight 1976, Ukkonen 1995, etc.)



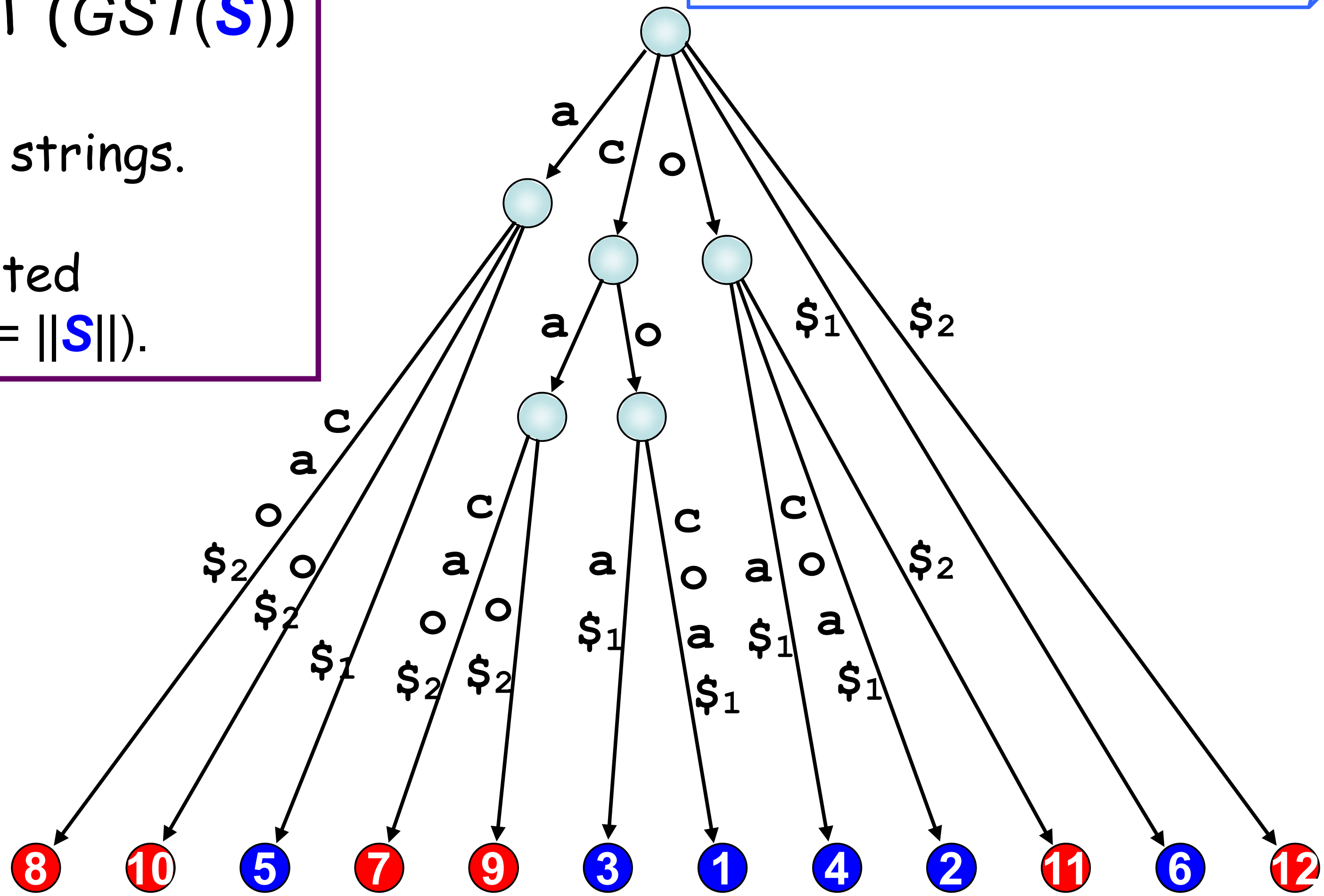
s = cocoa$_1$
123456

Generalized Suffix Tree

path(*v*) = concatenation of labels from *root* to node *v*.

Generalized ST (*GST*(***S***))

- ST for set ***S*** of strings.
- Can be constructed in $O(N)$ time ($N = ||S||$).



S = {cocoa$\$_1$, cacao$\$_2$}

Find Best Single Pattern

- We can restrict the candidates for pattern ***p*** to those represented by the nodes of *GST*(***S***).
  - there are only *O*(*N*) nodes
- For all nodes *v* in *GST*(***S***), *M*(*path*(*v*), ***S***) is computable in total *O*(*N*) time.
  (Color Set Size Problem, Hui 1992)
- Output *path*(*v*) of the best score.
- *O*(*N*) time & space for single pattern case.

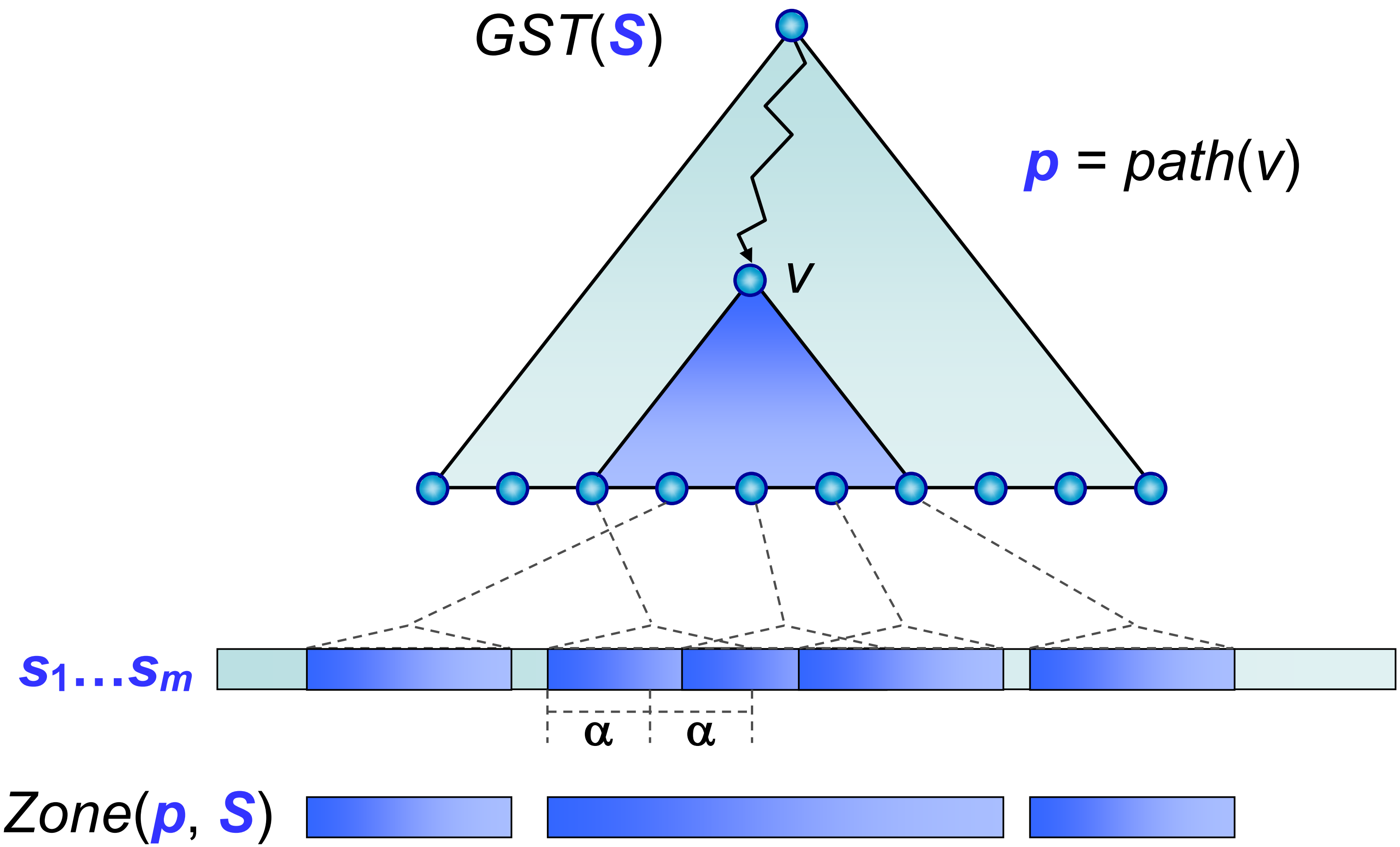
Find Best Pair $p \wedge_\alpha q$

Algorithm Sketch:

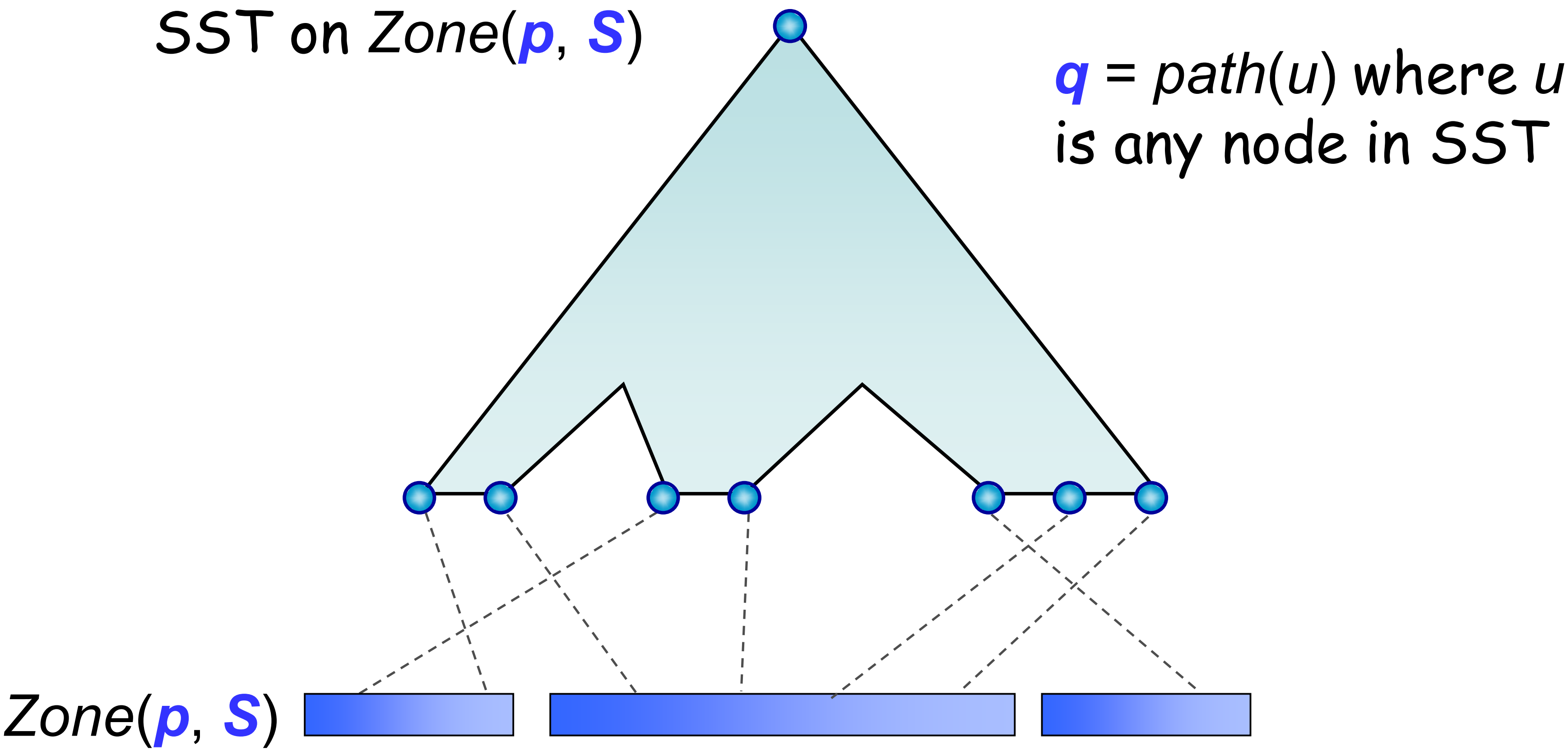
- For each candidate of first pattern ***p***
  1. Compute **Zone(*p*, *S*)** - the region covered by α from each position of ***p*** in ***S***.
  2. Build **sparse** suffix tree (SST) on *Zone*(***p***, ***S***).
  3. For each node *u* in SST, compute score.
- Output pattern pair of the best score.

Sparse suffix trees have been studied by Kärkkäinen and Ukkonen 1996, Andersson et al. 1999

Zone(p, S)



Build SST on *Zone*(*p*, *S*)



q = *path*(*u*) for some node *u* in SST on *Zone*(**p**, **S**)

⇕

q is such a pattern that **p** $\wedge_\alpha$ **q** matches **S**

Analysis

- We have $O(N)$ choices for first pattern ***p***.
  - For each candidate ***p***
    - Compute *Zone*(***p***, ***S***) – $O(N)$ time
    - Construct SST on *Zone*(***p***, ***S***) – $O(N)$ time

- Thus it takes ***$O(N^2)$ time*** in total.

- We need ***$O(N)$ space*** since we use one SST at each stage of the algorithm.

Find Best Pair $p \wedge_{\alpha} \neg q$

$O(N^2)$ time $\quad$ $O(N)$ time $\quad$ $O(N^2)$ time

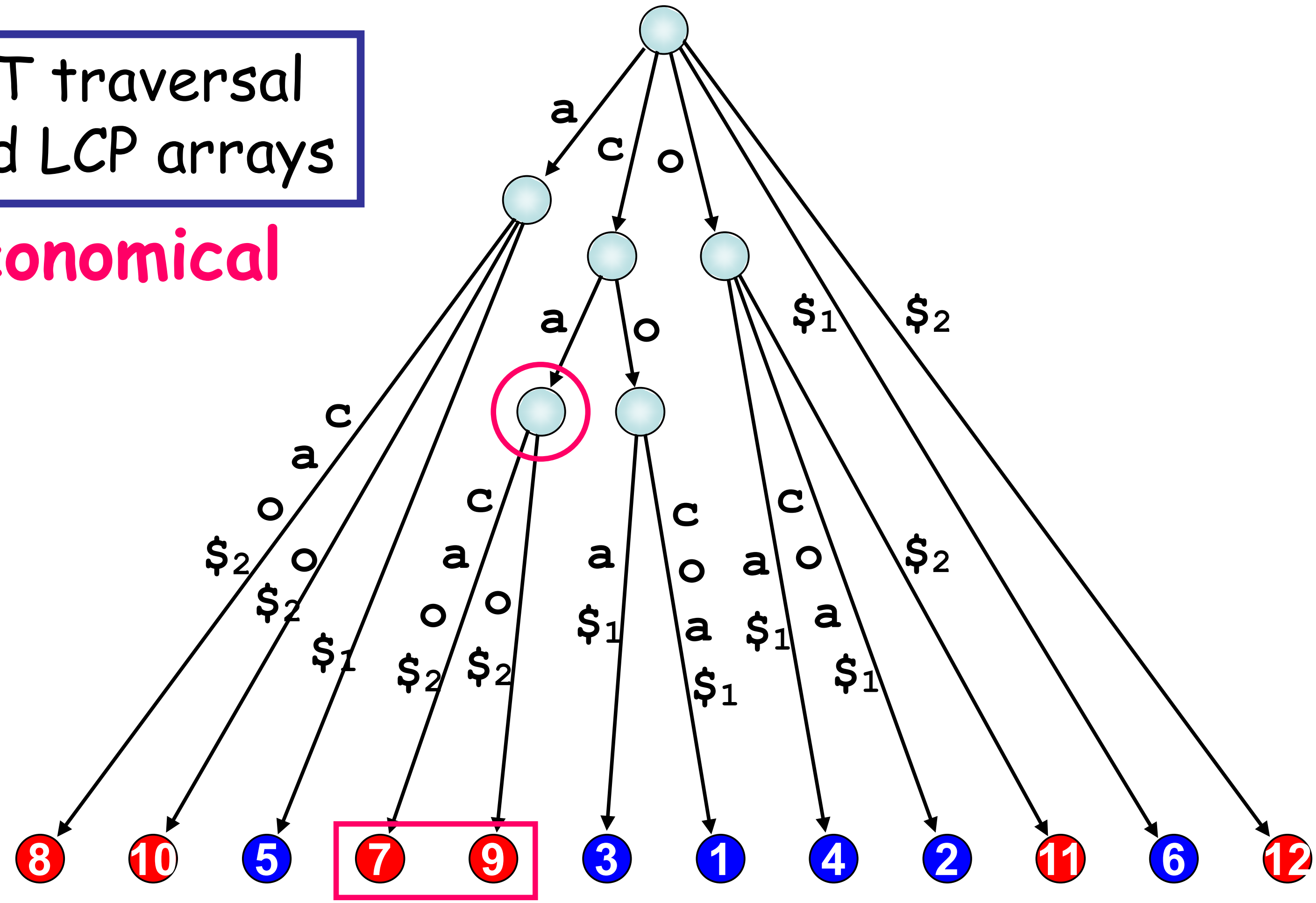
$$M(\overline{\pi}, S) = M(p, S) - M(\pi, S)$$

$$\pi = p \wedge_{\alpha} q, \quad \overline{\pi} = p \wedge_{\alpha} \neg q$$

Implementation with Suffix and LCP Arrays

Simulate GST traversal by suffix and LCP arrays

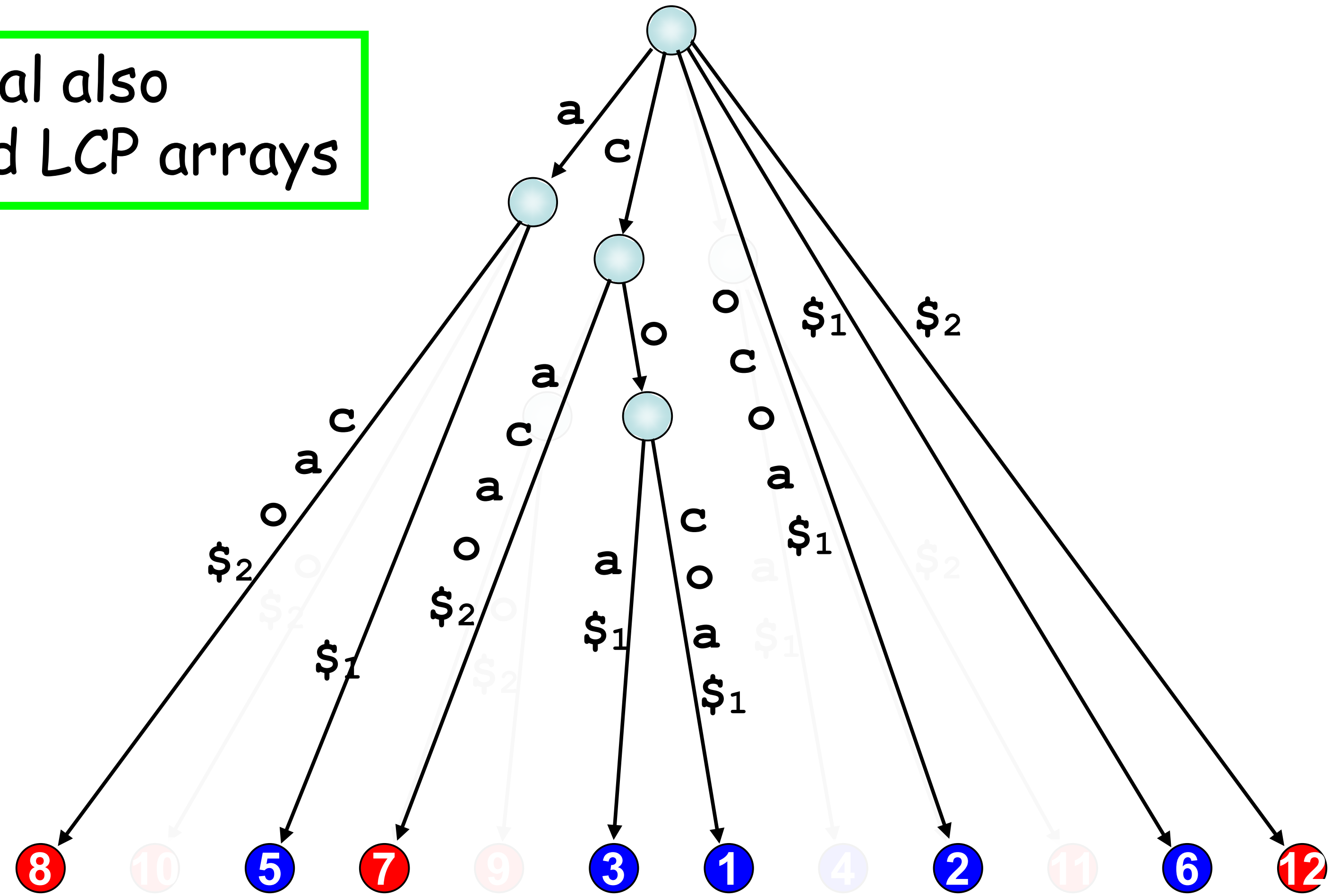
→ **space-economical**



SA(***S***)	8	10	5	7	9	3	1	4	2	11	6	12
LCP(***S***)	0	1	1	0	2	1	2	0	1	1	0	0

Implementation with Suffix and LCP Arrays

SST traversal also by suffix and LCP arrays



SA(***S***)	8	-	5	7	-	3	1	-	2	-	6	12
LCP(***S***)	0	1	1	0	2	1	2	0	1	1	0	0

Extended to Correlated Patterns

- Each sequence s_i in S may be associated with numeric value r_i, such as gene expression level.

- Wanted!:
  pattern pair π that matches sequences s_i with high r_i, but doesn't match sequences s_j with low r_j (or vice versa).

➔ pattern pair π that maximizes $score(M(\pi, S), R(\pi, S))$.

- $O(N^2)$ time & $O(N)$ space to find optimal pairs $p \wedge_\alpha q$ and $p \wedge_\alpha \neg q$

$R(\pi, S)$: sum of r for all s which π matches.

Computational Experiments

Yeast mRNA

- 3'UTR predicted processing site sequences (100nt each)
    - 379 fast degrading   ($t_{0.5} < 10$ min.)
    - 393 slowly degrading ($t_{0.5} > 50$ min.)

  Divided according to mRNA decay rate measurements (Wang et al. 2002, Graber 2003)

- score function: chi-squared test statistic

Computational Experiments (Contd.)

BEST

	fast	slow
AUA $\wedge_{10}$ $\neg$ **UGUA**	*159/393*	*248/379*

known biding site of the PUF protein which is important in mRNA regulation

---

AUA $\wedge$ **UGUA**	*268/393*	*190/379*
AUA $\wedge_{10}$ **UGUA**	*231/393*	*123/379*

may influence how efficiently **UGUA** functions, when it is close to **UGUA**??

Conclusions

- $O(N^2)$ time $O(N)$ space algorithm to find optimal pattern pair with bounded distance α
- Efficient implementation with suffix arrays
- Biologically relevant patterns discovered
- It can be extended to more advanced versions of bounded distance:
  - B) $O(m^2N^2)$ time & $O(m^2N)$ space
  - C) $O(N^3)$ time & $O(N^2)$ space
  - D) $O(m^2N^3)$ time & $O(N^2+m^2N)$ space