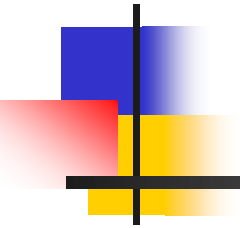
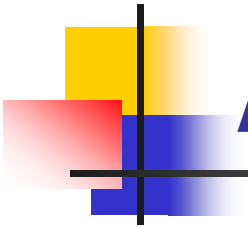


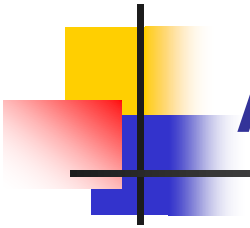
Encontrando Sementes e Repeats

Katia Guimarães



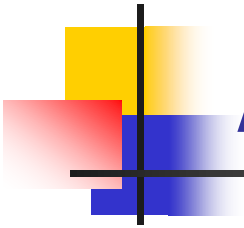
Árvores de Sufixos

- Apresentadas por Weiner (1973)
- Construção linear demonstrada por McCreight (1976)
- Modelo apresentado
  - Ukkonen (1995)
  - Compacto
  - Intuitivo
  - Versátil
  - ***On-line***



Árvores de Sufixos

- Definição
  - Uma árvore-Σ^+ é uma árvore n-ária tal que
    - Cada aresta tem um rótulo associado (seqüência não vazia de símbolos)
    - Rótulos de duas arestas com origem no mesmo vértice não podem começar com o mesmo símbolo.

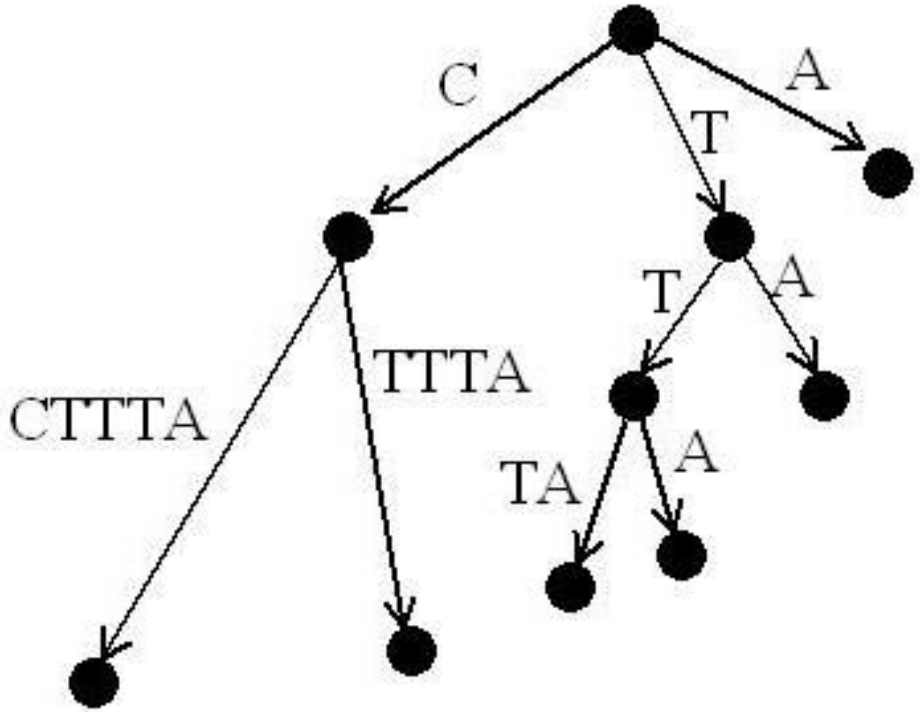


Árvores de Sufixos - Variações

- Árvore de Sufixos Compacta –
  Cada aresta tem um rótulo maximal

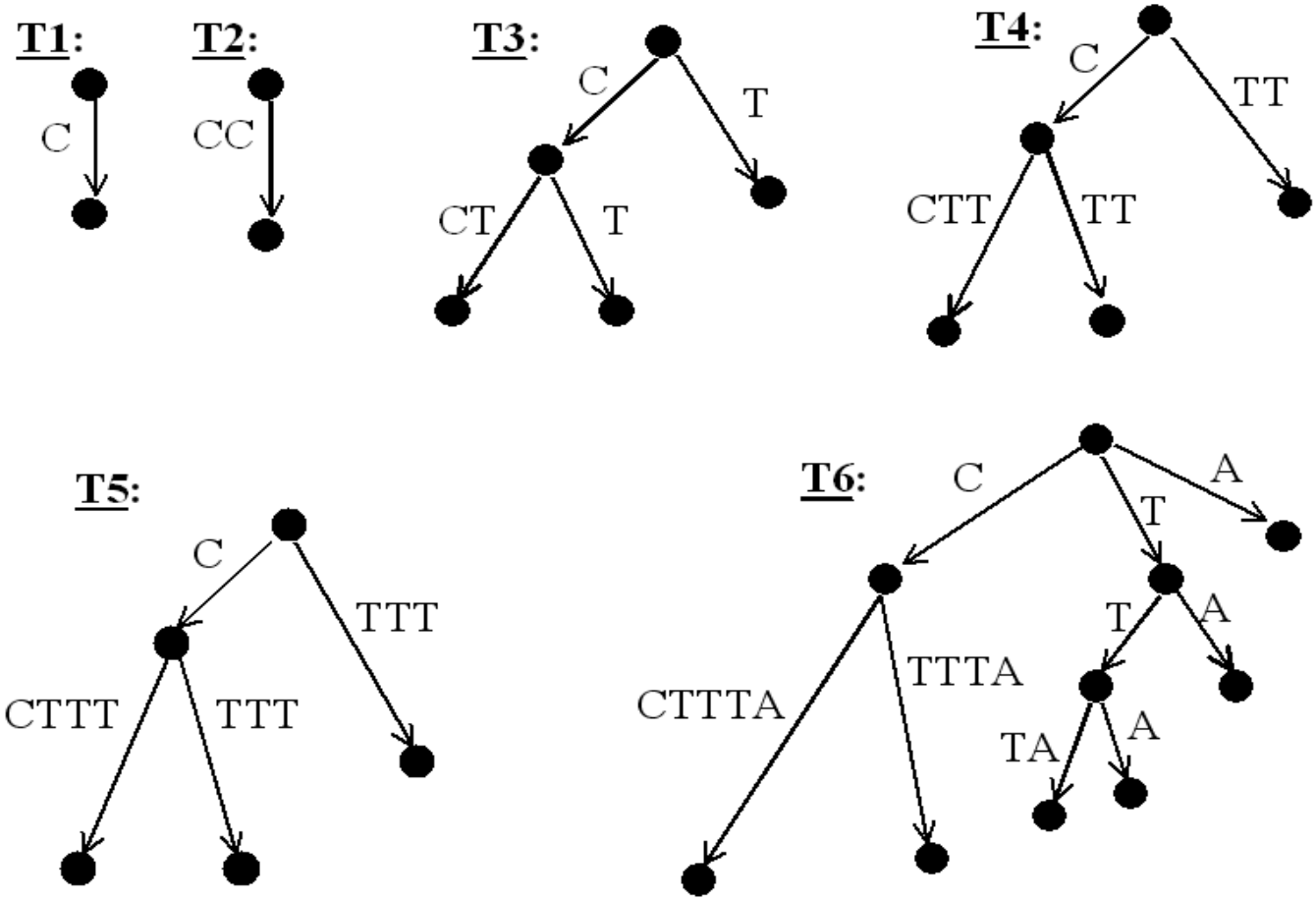
- Árvore de Sufixos Expandida
  Acrescenta-se um símbolo especial $
  ao final da cadeia

Árvores de Sufixos Compacta



S = CCTTTA

Construindo a Árvores de Sufixos da Seqüência CCTTTA

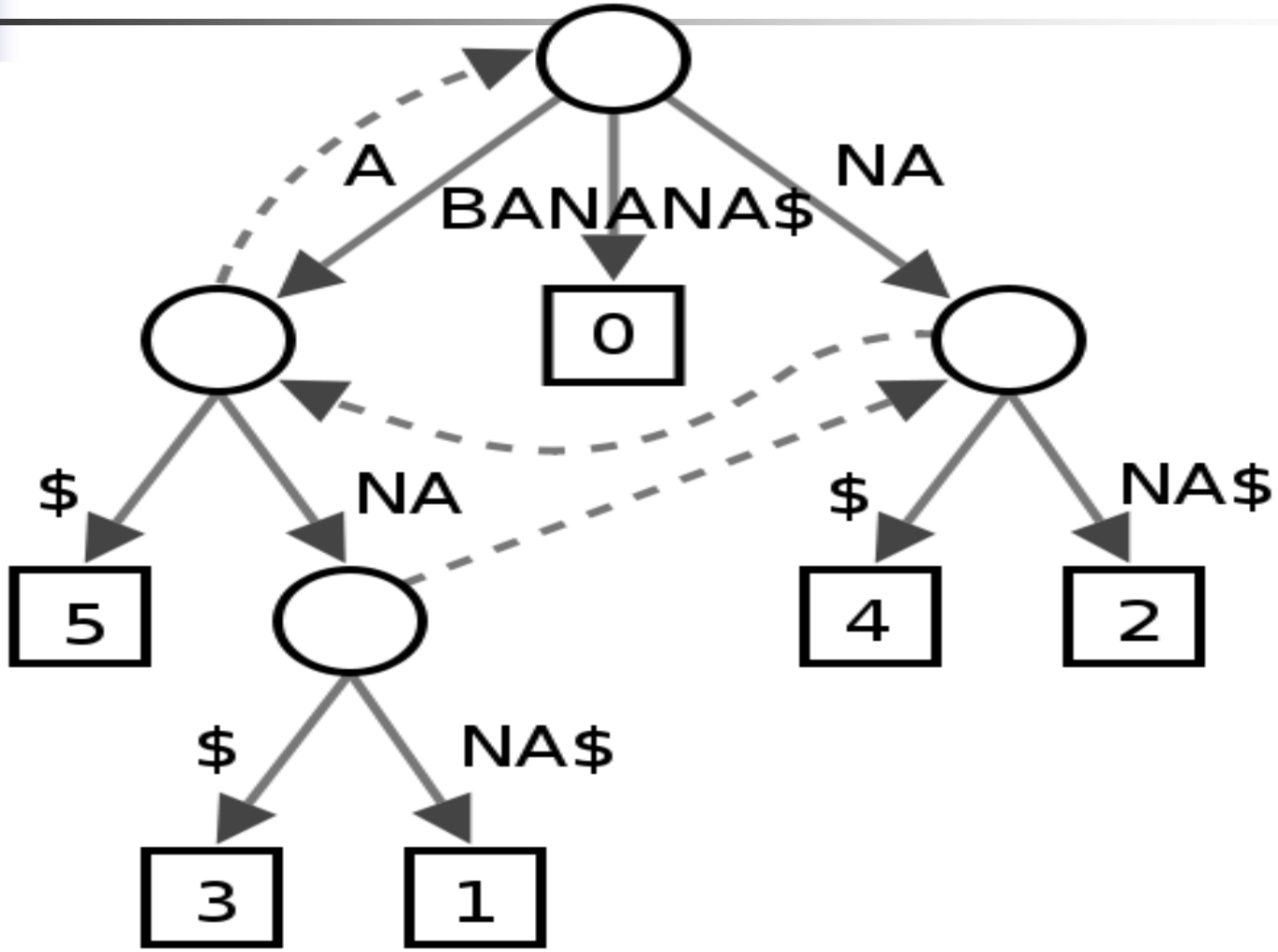


Árvores de Sufixos

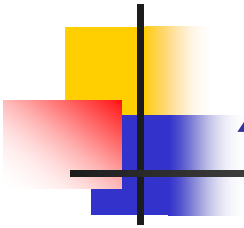
- Para fazer uma construção eficiente:
  - Obter, durante a *i*-ésima iteração as referências para cada vértice de rótulo $X[j..i-1]$, $j = 1, 2, ..., i-1$, na árvore T_{i-1}
  - *Suffix links*
    - Identificar a fronteira da árvore T_{i-1}

      ou seja, a seqüência dos *loci* de todos os sufixos de *X*, do maior para o menor

Árvores de Sufixos da Cadeia BANANA com Suffix Links

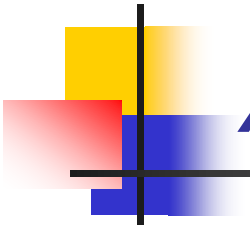


katia@cin.ufpe.br



Arrays de sufixos

- Manber and Myers (1993)
- Organiza todos os sufixos de uma seqüência em ordem lexicográfica crescente
- Possibilita busca binária
- Espaço
  - *O(n)*
- Tempo
  - *O(n. log n)*
- Tempo de busca por um padrão Y
  - *O(|Y| + log n)*



Arrays de sufixos

- Tempo de construção um pouco mais longo do que as árvores de sufixo
- Vantagens
  - Construção simples
  - Econômico em termos de espaço

Arrays de sufixos

11	ε
10	A
7	ABRA
0	ABRACADABRA
3	ACADABRA
5	ADABRA
8	BRA
1	BRACADABRA
4	CADABRA
6	DABRA
9	RA
2	RACADABRA

X = ABRACADABRA

0 1 2 3 4 5 6 7 8 9 0 1

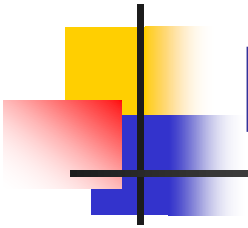
Pos:

11	10	7	0	3	5	8	1	4	6	9	2

Repetições com substituições

- Diferentes tipos de erros introduzem diferentes graus de dificuldade
  - Distância de Hamming
- Estratégia
  - Identificar repetições exatas de tamanho pequeno
    - Sementes
  - A partir destes, identificar trechos de tamanho maior

CDEFGHIJKL**MATRIX**GJKFSVECQ**MADRIX**MNOPQRSTUVW
CDEFGHIJKL**CARANDIRU**FSVECQ**CATANVIRU**PQRSTUVW



Repetições com substituições

- A computação das sementes pode ser feita em tempo *O(n)* usando uma árvore de sufixos *S*
- A verificação das possíveis extensões de uma semente pode ser feita por programação dinâmica

Repetições com substituições

- Seqüência

$$S = CCTTTAACCCCGGGGCCAATTTCACTTGGGGTA.$$

- Sementes na seqüência S de comprimento pelo menos 3

Seqüência	Par 1	Par 2
$GGGG$	(11,14)	(27,30)
TTT	(3,5)	(19,21)

Repetições com substituições

- Extensões da semente GGGG

Semente $GGGG$						
q	T_{esq}	Extensão 1	Extensão 2	Extensão 1	Extensão 2	T_{dir}
0	0	$GGGG$	$GGGG$	$GGGG$	$GGGG$	0
1	1	$CGGGG$	$TGGGG$	$GGGGC$	$GGGGT$	1
2	2	$CCGGGG$	$TTGGGG$	$GGGGCC$	$GGGGTA$	2

Repetições com substituições

- Extensões da semente TTT

Semente TTT						
q	T_{esq}	Extensão 1	Extensão 2	Extensão 1	Extensão 2	T_{dir}
0	0	TTT	TTT	TTT	TTT	0
1	1	$CTTT$	$ATTT$	$TTTAAC$	$TTTCAC$	3
2	2	$CCTTT$	$AATTT$	$TTTAACC$	$TTTCACT$	4

Considerando inserções e remoções

- Modelo mais complexo, porém, mais realístico
- Distância de Levenshtein
- *k*-erro repetição

Considerando inserções e remoções

- U e V cadeias de comprimento *m* e *n* respectivamente
- q $\in$ [0, k]
- Em direita$_E$ (esquerda$_E$) as cadeias U e V representam as extensões à direita (respectivamente, `a esquerda) das sementes que são maximais com relação ao erro, considerando a distância de edição

Considerando inserções e remoções

- Valores de U e de V para extensões à esquerda e à direita de

GGTATGCAGGGGCGAACTATAGCGGGGGGGACTTAGAT

- semente GGGG e q≤2

Semente $GGGG$				
q	$U(esquerda)$	$V(esquerda)$	$U(direita)$	$V(direita)$
0	$GGGG$	$GGGG$	$GGGG$	$GGGG$
1	$GCAGGGG$	$GCGGGG$	$GGGGCGA$	$GGGGGGA$
2	$TATGCAGGGG$	$TATAGCGGGG$	$GGGGCGAACT$	$GGGGGGACT$

Considerando inserções e remoções

Algoritmo *MDR (S, l, k)*

início

Compute todas as sementes

Para cada semente $((i_1, j_1), (i_2, j_2))$ faça

Compute as tabelas $T_{dir}(q) = direita_E(s[j_1+1,n], s[j_2+1,n], q)$

$T_{esq}(q) = esquerda_E(s[1, i_1-1], s[1, i_2-1], q)$

Para cada $q \in [0, k]$ faça

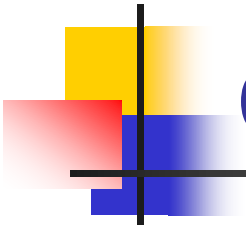
Para cada par $(x_l, y_l) \in$ esquerda(q) e

cada par $(x_r, y_r) \in$ direita(k-q) faça

Se $(j_1 - i_1 + 1 + x_l + x_r \geq l)$ e $(j_2 - i_2 + 1 + y_l + y_r \geq l)$ então

reporte a *k*-erro repetição$((i_1 - x_l, j_1 - x_r), (i_2 - y_l, j_2 - y_r))$

fim *MDR*



Considerando inserções e remoções

- As sementes podem ser estendidas em tempo
  - $O(n^2)$
    - Programação dinâmica simples
  - $O(kn)$
    - Algoritmo em [Ukkonen, 1985].