VSTree: Uma Ferramenta de Execução e Visualização de Algoritmos de Construção e Busca em Árvores de Sufixo

Felipe Alves da Louza[1], João de Santana Brito Junior[1], Ricardo Rodrigues Ciferri[2], Cristina Dutra de Aguiar Ciferri[1]

[1]Departamento de Ciências de Computação – Universidade de São Paulo
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

[2]Departamento de Computação – Universidade Federal de São Carlos
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil

{louza|joaojr}@grad.icmc.usp.br,ricardo@dc.ufscar.br,cdac@icmc.usp.br

Abstract. *A suffix tree is a data structure largely used as a biological access method to search for homology relations among nucleotide and amino acid sequences. This paper introduces VSTree, a visual tool that assists the manipulation of suffix trees, thereby aiding their understanding. The tool allows for the interactive execution and visualization of the suffix tree's build and search algorithms. VSTree also tackles other important concepts, such as prefixed, implicit and compressed suffix trees.*

Resumo. *A árvore de sufixo é uma estrutura de dados muito usada por métodos de acesso biológicos para realizar pesquisa de similaridade em sequências de nucleotídeos e aminoácidos. Este artigo apresenta VSTree, uma ferramenta visual para a manipulação de árvores de sufixo, auxiliando assim seu aprendizado. A ferramenta permite a execução e a visualização interativa dos algoritmos de construção e busca de árvores de sufixo, além de manipular outros conceitos importantes, tais como árvores de sufixo prefixadas, implícitas e comprimidas.*

1. Introdução

O avanço tecnológico dos laboratórios de biologia molecular tem proporcionado um grande aumento no volume de dados armazenados em bancos de dados biológicos (BDBs). Os dados referem-se à estrutura do material genético dos seres vivos, os quais são compostos por sequências de caracteres (*strings*) que representam nucleotídeos formados pelas bases nitrogenadas adenina (A), citosina (C), timina (T), guanina (G) e uracila (U), ou por *strings* que representam os 20 aminoácidos que compõem as proteínas (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y) [Gusfield 1997].

A pesquisa de similaridade em *strings* armazenadas em BDBs é um problema fundamental em bioinformática, a qual tem sido usada para comparar duas ou mais *strings* e determinar se elas são homólogas. Por meio de buscas de relações de homologia, é possível responder questões importantes como: (i) Quais organismos possuem genes homólogos ao gene *eyeless* da mosca-das-frutas?; (ii) Qual parte do DNA é responsável por uma doença hereditária?; e (iii) Qual a função de um gene? [Korf et al 2003].

Desde que a maioria das consultas envolvendo pesquisa de similaridade possui apenas uma pequena parte do BDB como resposta, o processamento dessas consultas é

Os autores agradecem o apoio financeiro das seguintes agências de fomento à pesquisa do Brasil: FAPESP, CNPq, CAPES e FINEP.

usualmente auxiliado por métodos de acesso. Dentre os métodos de acesso existentes na literatura voltados à indexação de dados biológicos, uma gama expressiva é baseada em uma estrutura de dados na forma de árvore para armazenar as *strings* [Barsky et al. 2010]. Essa estrutura, chamada de árvore de sufixo, permite resolver eficientemente problemas que, em geral, envolvam buscas por *substrings* em uma *string*. Portanto, árvores de sufixo têm sido aplicadas em diferentes situações, e têm sido o foco de interesse de diversos pesquisadores. Surge, então, a necessidade de desenvolvimento de uma ferramenta que permita a manipulação de árvores de sufixo, auxiliando assim o aprendizado de diferentes conceitos relacionados a essas estruturas e, indiretamente, oferecendo suporte para o entendimento de métodos de acesso baseados em árvores de sufixo.

Este artigo apresenta a ferramenta VSTree (*Visual Suffix Tree*), a qual pode ser usada para demonstrar visualmente e interativamente a construção de árvores de sufixo e a busca de *substrings* nessas árvores. Os diferenciais da ferramenta são:

- Execução e visualização *interativa* e *completa* dos algoritmos de construção e de busca;
- Construção de árvores de sufixo usando diferentes métodos, tais como os métodos trivial e Naive Weiner;
- Construção de diferentes tipos de árvores de sufixo, como árvores de sufixo prefixadas e árvores de sufixo implícitas;
- Visualização de outros conceitos importantes relacionados a árvores de sufixo, como os sufixos que compõem cada árvore de sufixo e árvores de sufixo comprimidas e não-comprimidas; e
- Oferecimento de opções de formatação visual das árvores de sufixo exibidas.

Este artigo está estruturado da seguinte forma. Na Seção 2 são resumidos trabalhos correlatos, enquanto que na Seção 3 é descrita a fundamentação teórica. Na Seção 4 é detalhada a ferramenta VSTree, em termos de objetivos, funcionalidades e interfaces. O artigo é concluído na Seção 5, a qual destaca também trabalhos futuros.

2. Trabalhos Correlatos

Duas ferramentas que possuem funcionalidades afins são *Suffix Tree Construction Applet* [Construction] e *Growing a Suffix Tree* [Growing]. Entretanto, essas ferramentas enfocam apenas a construção de árvores de sufixo, e não a busca por *substrings* nessas árvores. Outras limitações dessas ferramentas é que elas não tratam árvores de sufixo prefixadas e árvores de sufixo implícitas, que são conceitos importantes usados pelos métodos de acesso voltados à indexação de BDBs. Ademais, elas possuem interfaces estáticas no sentido que não permitem explorar outros conceitos de árvores de sufixo, como a visualização dos sufixos que compõem cada árvore e a mudança de árvores de sufixo comprimidas para não-comprimidas, e vice-versa. Tais limitações dificultam o uso dessas ferramentas, e motivam o desenvolvimento de uma ferramenta flexível que supra essas limitações. A ferramenta VSTree foi desenvolvida com esse objetivo.

3. Árvores de Sufixo

Seja $\sum = \{\alpha_1, ..., \alpha_\sigma\}$ um alfabeto com σ caracteres. Seja $\sum^*$ um conjunto com todas as

strings que podem ser construídas a partir de $\sum$ e seja $S \in \sum^*$ uma *string* com m caracteres. Seja $S[i:j]$ ($1 \leq i \leq j \leq m$) uma *substring* entre (e inclusive) o i-ésimo e j-ésimo caracteres de S. Sejam todas as *substrings* $S[1:i]$ prefixos e todas as *substrings* $S[i:m]$ sufixos de S. Seja $\$ \notin \sum$ um caractere terminal que representa o final de uma *string*.

Uma árvore de sufixo para uma *string S,* representada por *Ts*, é uma árvore enraizada com as seguintes propriedades [Gusfield 1997]. (i) Ela possui $|S|$ nós folhas numerados de 1 a $|S|$, onde $|S|$ é o número de caracteres de S. (ii) Cada nó interno, exceto a raiz, tem pelo menos 2 filhos. (iii) Cada aresta representa, por meio de seu rótulo, uma *substring* de S. (iv) Duas arestas que saiam do mesmo nó não podem representar *substrings* com prefixo comum. (v) Para cada folha i, a concatenação dos rótulos das arestas que estão no caminho da raiz até essa folha forma o sufixo de S que começa na posição i. (vi) Sufixos com prefixos em comum fazem parte da mesma subárvore.

O algoritmo trivial de construção de árvores de sufixo primeiro cria um nó raiz R[T] de *Ts*, depois adiciona um filho em R[T] para representar o sufixo de maior tamanho $S[1:|S|]$ e, em seguida, adiciona os outros sufixos de S (i.e, $S[2:|S|]$, ..., $S[|S|:|S|]$). A cada iteração, conforme necessário, as arestas são particionadas de forma a preservar as propriedades (ii) e (iv). Em contrapartida, embora na construção Naive Weiner a árvore seja construída de forma similar ao método trivial, os sufixos são inseridos em ordem decrescente (i.e, $S[|S|:|S|]$, $S[|S\text{-}1|:|S|]$, ..., $S[1:|S|]$). Com relação à busca por uma *substring* em uma árvore de sufixo, ela é feita de forma determinística, uma vez que as arestas dos filhos de um nó pai não compartilham prefixos comuns.

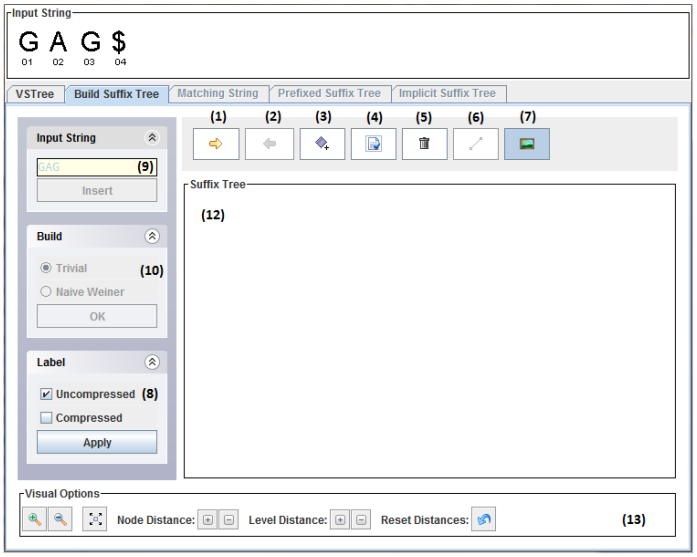
Conceitos importantes usados pelos métodos de acesso voltados à indexação de BDBs são árvores de sufixo prefixadas, implícitas e (não-)comprimidas. Uma árvore de sufixo prefixada Ps^p, para uma *string S,* é uma árvore de sufixo que contém apenas sufixos de S que começam com o prefixo p [Barsky et al. 2010]. Já uma árvore de sufixo implícita *Is* é uma árvore gerada a partir de *Ts* eliminando-se o caractere \$. Finalmente, árvores de sufixo que atendem à propriedade (iii) são chamadas não-comprimidas. Uma árvore de sufixo comprimida, por outro lado, não mostra cada *substring* de S que seus rótulos representam. Para cada rótulo, elas mostram dois índices: um indicando a posição de S na qual a *substring* se inicia e outro indicando o tamanho da *substring*.

4. A Ferramenta VSTree

VSTree é uma ferramenta de execução e visualização de algoritmos de construção e busca em árvores de sufixo. Essa ferramenta oferece uma interface gráfica para: (i) construção de árvores de sufixo; (ii) buscas em árvores de sufixo; (iii) construção de árvores de sufixo prefixadas; e (iv) geração de árvores de sufixo implícitas. Cada funcionalidade é manipulada por meio de uma etiqueta, como mostrado na Figura 1, sendo que todas as etiquetas são semelhantes em termos de opções oferecidas e de layout.

A Figura 1 ilustra a interface da etiqueta **Build Suffix Tree**. A execução e a visualização passo-a-passo dos algoritmos de construção e de busca podem ser feitas usando-se o botão **(1)** para executar o próximo passo, ou o botão **(2)** para voltar ao passo anterior. Por outro lado, o botão **(3)** executa os algoritmos de uma única vez e o botão **(4)** desfaz todos os passos de uma única vez. O botão **(5)** remove a árvore de sufixo exibida na área de exibição (**12**) e permite a especificação de uma nova *string* de entrada ou de busca, enquanto que o botão **(6)** permite a visualização dos sufixos que compõem

a árvore. Detalhes sobre o botão **(7)**, o qual permite a manipulação de opções visuais (**13**), são descritos na Seção 4.5. Outra funcionalidade presente nas interfaces de todas as etiquetas é a visualização da árvore de sufixo comprimida e/ou não-comprimida **(8)**.



Etiquetas:

(i) Build Suffix Tree: Construção de uma árvore de sufixo T_s a partir de uma *string* de entrada *S*, usando os métodos de construção trivial ou Naive Weiner.

(ii) Matching String: Busca por *substrings* em T_s.

(iii) Prefixed Suffix Tree: Construção das árvores de sufixo prefixadas de T_s.

(iv) Implicit Suffix Tree: Geração da árvore de sufixo implícita de T_s.

Figura 1. Interface da etiqueta Build Suffix Tree.

4.1. Etiqueta Build Suffix Tree

A etiqueta **Build Suffix Tree** oferece funcionalidades voltadas à construção de uma árvore de sufixo *Ts* a partir de uma *string* de entrada *S*, usando os métodos de construção trivial ou Naive Weiner. No exemplo da Figura 1, as opções escolhidas são: *S* = GAG$ **(9)**; método de construção trivial **(10)**; e árvore não-comprimida **(8)**.

A Figura 2 mostra as árvores intermediárias construídas na VSTree que demonstram, passo-a-passo, a execução do algoritmo de construção trivial para *Ts*. Primeiramente, cria-se um vértice inicial, a raiz R[T] (Figura 2a). Em seguida, adiciona-se a *Ts* o sufixo de maior tamanho, S[1:3] = GAG$ (Figura 2b). O terceiro passo adiciona o sufixo S[2:3] = AG$. Desde que não há nenhum filho de R[T] que comece com o mesmo prefixo de S[2:3], S[2:3] é inserido em R[T] (Figura 2c). O último sufixo a ser adicionado é S[3:3] = G$. Neste caso já existe um sufixo que começa com o caractere G. Assim, primeiro a aresta que representa S[1:3] é particionada e, em seguida, S[3:3] é adicionado a *Ts* (Figura 2d). A Figura 2e mostra os sufixos de *Ts*. Pode-se visualizar a propriedade (v), selecionando-se um desses sufixos. Note que, se a execução fosse feita de uma única vez, apenas a árvore da Figura 2d seria exibida. Todas as figuras estão apresentadas exatamente da mesma forma que foram geradas pela VSTree.

4.2. Etiqueta Matching String

A etiqueta **Matching String** oferece funcionalidades voltadas à busca por *substrings* na árvore de sufixo visualizada na área de exibição. A Figura 3 mostra a execução passo-a-passo da busca pela *substring Ss* = G$ na árvore *Ts* ilustrada na Figura 2d. A busca inicia-se na raiz R[T] de *Ts* (Figura 3a). Em seguida, segue-se um caminho a partir dos casamentos entre *Ss* e as *substrings* presentes nas arestas do caminho (Figura 3b). Por

fim, atinge-se a folha 3 (Figura 3c). A ferramenta exibe, então, uma mensagem, a qual indica que *Ss* = *G$* é um sufixo que ocorre na posição 3 de *S* = GAG$.

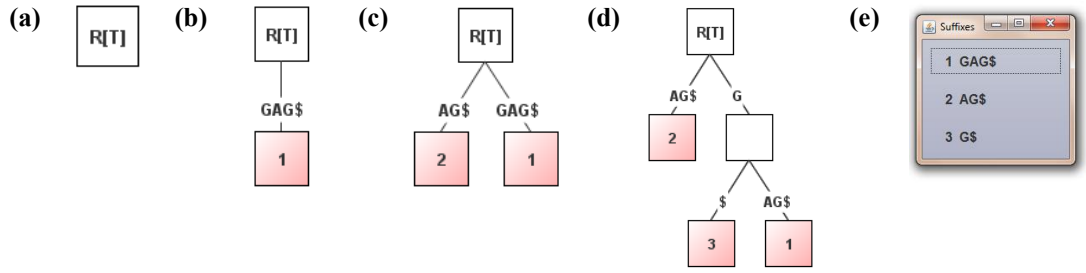


Figura 2. Construção passo-a-passo da árvore de sufixo T_S.

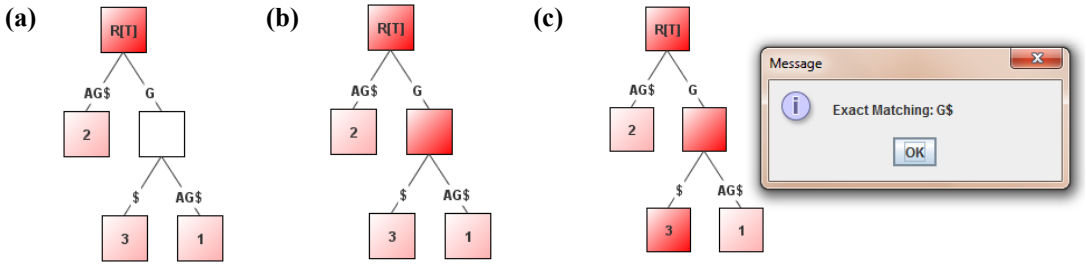


Figura 3. Busca passo-a-passo pela *substring Ss* = G$ em *Ts*.

4.3. Etiqueta Prefixed Suffix Tree

A etiqueta **Prefixed Suffix Tree** oferece funcionalidades voltadas à construção de árvores de sufixo prefixadas. Essa construção é semelhante à construção trivial descrita na Seção 3, diferindo-se pelo fato de que, a cada iteração, os sufixos que não começam com o prefixo desejado são descartados. A Figura 4 mostra a execução passo-a-passo da construção da árvore prefixada Ps^p para o prefixo p = G da árvore *Ts* ilustrada na Figura 2d. Primeiramente, cria-se um vértice inicial, a raiz R[T] (Figura 4a). Em seguida, adiciona-se a Ps^p o sufixo de maior tamanho, S[1:3] = GAG$ (Figura 4b). O terceiro passo descarta o sufixo S[2:3] = AG$, pois esse não possui o prefixo p = G. Por fim, adiciona-se a Ps^p o sufixo S[3:3] = G$ (Figura 4c).

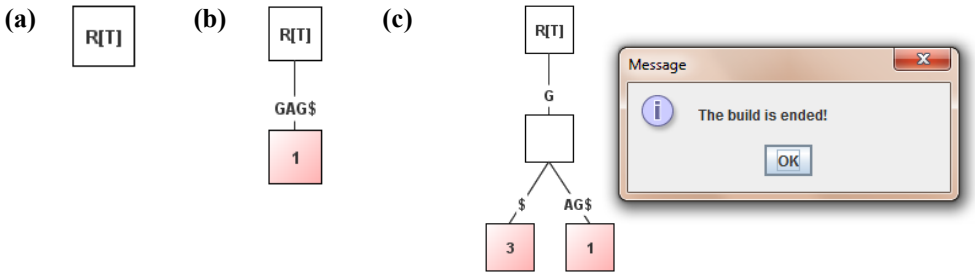


Figura 4. Construção passo-a-passo da árvore prefixada para o prefixo *p* = G de*Ts*.

4.4. Etiqueta Implicit Suffix Tree

A etiqueta **Implicit Suffix Tree** oferece funcionalidades voltadas à geração da árvore de sufixo implícita de uma árvore de sufixo. A Figura 5 mostra a execução passo-a-passo da geração da árvore implícita *Is* para a árvore *Ts* ilustrada na Figura 2d. O primeiro passo remove todos caracteres $ dos rótulos de *Ts* (Figura 5a), enquanto que o segundo passo remove os nós com arestas vazias (Figura 5b). Já o terceiro passo remove os nós com menos do que dois filhos e os une aos seus respectivos nós pais (Figura 5c).

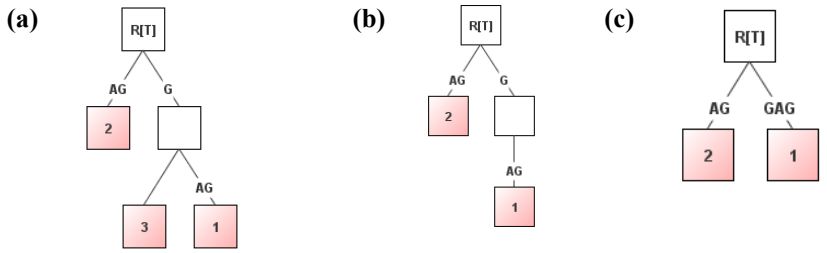


Figura 5. Construção passo a passo da árvore de sufixo implícita *Is* de *Ts*.

4.5. Opções de Formatação Visual

Outras funcionalidades oferecidas pela VSTree, ilustradas na Figura 1 (**13**), incluem opções de ajustes da forma na qual as árvores de sufixo são exibidas. Esses ajustes referem-se: (i) às opções de *zoom out, zoom in* e visualização uniforme da árvore; (ii) ao aumento e à redução da distância entre os nós das árvores; (iii) ao aumento e à redução da distância entre os níveis das árvores; e (iv) à possibilidade de se mover os nós e arestas das árvores e depois retornar às distâncias originais.

5. Conclusões

Este artigo apresentou a ferramenta VSTree, a qual pode ser usada para demonstrar visualmente e interativamente a construção e a busca em árvores de sufixo, facilitando assim o aprendizado de conceitos relacionados a essas estruturas de dados amplamente usadas por métodos de acesso voltados à indexação de BDBs. A implementação da VSTree foi realizada usando-se a linguagem Java e a biblioteca de visualização JGraph. Isso permite que ela seja portátil, funcionando tanto em Linux quanto em Windows. Como atividades futuras, serão incorporados outros algoritmos de construção e busca na VSTree, além de possibilitar uma forma de compará-los.

Referências Bibliográficas

Barsky, M., Stege, U., Thomo, A., Upton, C. (2010). “A survey of practical algorithms for suffix tree construction in external memory”. In *Softw. Pract. Exper*. (*accepted*).

Construction. “Suffix tree construction applet”. Disponível em www14.informatik.tu-muenchen.de/konferenzen/Ferienakademie99/maass/applet.html. Acesso: 06/2010.

Growing. “Growing a suffix tree”. Disponível em pauillac.inria.fr/~quercia/documents-info/Luminy-98/albert/JAVA+html/SuffixTreeGrow.html. Acesso: 06/2010.

Gusfield, D. (1997). “Algorithms on strings, trees, and sequences: computer science and computational biology”. Cambridge University Press.

Korf, I., Yandell, M., and Bedell, J. (2003). “BLAST”. O’Reilly.