Visual TaHs: software para auxiliar o ensino de tabelas *Hash* na disciplina de Estrutura de Dados

Fábio Carlos Moreno[1], Cinthyan R. Sachs C. de Barbosa[2], Edio Roberto Manfio[3]

[1]Centro de Ciências Tecnológicas – Universidade Estadual do Norte do Paraná (UENP) 86300-000 – Cornélio Procópio, PR – Brazil

[2]Programa de Pós-Graduação em Ciência da Computação – Universidade Estadual de Londrina (UEL) – 86057-970 – Londrina, PR – Brazil

[3]Faculdade de Tecnologia Deputado Júlio Julinho Marcondes de Moura (FATEC) – 17400-000 – Garça – SP – Brazil

fbio_moreno@yahoo.com.br, cinthyan@uel.br, edio.manfio@fatec.sp.gov.br

Abstract. *The Data Structure discipline is of fundamental importance for the Computer Science courses area and it has presented high repetition rates and low performance achievements. Thus, the purpose of this word is to present the "Visual TaHs" software, which can contribute to the teaching of Data Structure, based on the contextualization approach of the Hash table. An experiment with fourteen hash functions was implemented in the software, and an overload of elements in the lexicon was performed. With this experiment, it was possible to analyze and identify the best hashing functions through report and graphs. All the performed analysis can be evaluated by students, contextualizing the concept worked in the classroom.*

Resumo. *A disciplina de Estrutura de Dados é de suma importância para os cursos na área de Ciência da Computação e possui altos índices de reprovação e baixos desempenhos alcançados. Dessa forma, o objetivo deste trabalho é apresentar o software "Visual TaHs" que poderá contribuir para o ensino de Estrutura de Dados, com base na abordagem da contextualização da tabela Hash. Foi feito um experimento com quatorze funções hash implementadas no software, sendo realizada uma sobrecarga de elementos no léxico. Este experimento possibilitou, através dos relatórios e gráficos, analisar e identificar as melhores funções de espalhamento. Todas as análises realizadas poderão ser avaliadas pelos discentes, contextualizando o conceito trabalhado em sala de aula.*

1. Introdução às Tabelas Hash

A disciplina de Estrutura de Dados (ED) é de extrema importância para os cursos na área da Ciência da Computação [Barbosa e Júnior 2013]. Essa disciplina possui um conteúdo muito vasto de estruturas de programação, que tem a finalidade de resolver diversos problemas, como por exemplo, o de recuperação de dados, ordenação, além da percepção da lógica [Ziviani 2004]. Tal disciplina é considerada matéria essencial pela Sociedade Brasileira de Computação (SBC). Porém, a comissão de Educação da SBC entende que o tema evasão é preocupante para direção ou coordenação de cursos na área e isso tem afetado a vida de muitos estudantes, sendo um entrave para o desenvolvimento do país.

Em razão disso, Souza et al. (2011) e Boticki et al. (2012) defendem que a disciplina de ED deve ser abordada de forma prática, não se limitando apenas a livros, textos, lousa e giz, o que poderá acarretar numa dificuldade do processo de aprendizagem, visto a complexidade estrutural da disciplina. Portanto, o conteúdo aliado com o processo de ensino, pode ser uma das causas responsáveis pelos altos índices de reprovação e baixos desempenhos alcançados nas disciplinas de Algoritmos e Estrutura de Dados [Soares et al. 2004], [Wiedenbeck et al. 2004]. Esses índices podem ser comprovados pelos dados demonstrados pela UFPE-PROPLAN (2018), que detalha que tal disciplina teve uma taxa abaixo de 50% (cinquenta por cento) de aprovação, nos anos 2014 a 2017, em relação às outras disciplinas.

Dessa maneira, Souza et al. (2011) também destacam um motivo para esse alto índice de reprovação da referida disciplina que é a grande quantidade de conceitos abstratos ou complexos que dificultariam as implementações que os discentes teriam que realizar. Assim, a utilização de alguns recursos midiáticos pode auxiliar no contexto dessa disciplina, visto que, para Costa, Correa e Freitas (2014, p. 1336),

> o uso de equipamentos portáteis, como tablet, smartphone e netbook tornou-se cada vez mais comum. Com a ampliação do acesso à Internet, os estudantes podem utilizar esses dispositivos em qualquer lugar para ter acesso a conteúdos educacionais. No contexto do ensino superior essa facilidade é especialmente interessante para professores que tenham interesse em inovar suas práticas pedagógicas, apoiados pelas tecnologias de informação e comunicação.

Barbosa e Parreira Júnior (2013) apresentaram um mapeamento sistemático sobre ferramenta de apoio ao ensino da disciplina de ED, com a finalidade de buscar, catalogar, identificar e efetuar a classificação desse assunto, para que pesquisadores ou discentes realizem novos estudos e a possibilidade de propor novas ferramentas ou abordagens de ensino de ED. Nesse levantamento, constatou-se que a manipulação de vetores e ordenação como *QuickSort* e *HeapSort* tem recebido bastante atenção da comunidade científica, enquanto a estrutura de tabela *Hash* tem sido contemplada por poucos trabalhos.

As tabelas *Hash* são um tipo de estruturação para o armazenamento de informação extremamente simples, fácil de implementar e intuitiva quando se trata de organizar grandes quantidades de dados. O conceito central é a divisão de um universo de dados a ser organizado em subconjuntos mais facilmente gerenciáveis. A estruturação da informação em *tabelas Hash* visa principalmente permitir armazenar e procurar rapidamente grande quantidade de dados [Cormen et al. 2002], [Botelho 2004].

As funções *Hash* perfeitas são aquelas que possuem técnicas de refinamento e de transformação da chave para endereço direto, ou seja, uma vez fornecida a chave, há uma busca única de chaves em uma tabela estática [Cercone 1988], [Cormen et al. 2002], Alguns critérios para uma boa função *Hash* são definidas como: **1)** endereço *Hash* facilmente calculado; **2)** fator de carga da tabela *Hash* ser elevado para um dado conjunto de chaves; **3)** os endereços de *Hash* de um determinado conjunto de chaves são distribuídos uniformemente na tabela *Hash* e uma função *Hash* perfeita se diz ótima quando existe distribuição uniforme de seus endereços [Ziviani 2004].

As tabelas *Hash* são tipicamente usadas para indexação de grandes volumes de informação (como por exemplo, bases de dados). A implementação típica busca uma função *Hash* que seja de complexidade *O* (1), não importando o número de registros na

tabela e desconsiderando colisões. O ganho com relação a outras estruturas associativas (como um vetor simples) passa a ser maior conforme a quantidade de dados aumenta. [Botelho 2004], [Cormen et al. 2002].

As Tabelas de Endereçamento Direto são funcionais quando o universo *U* é pequeno. Se esse for grande, o armazenamento na tabela pode ser impossível, acarretando colisões que necessitam de algoritmos bem elaborados para dar o elemento uma nova posição [Cormen et al. 2002]. Porém, utilizando Tabela *Hash* Encadeada, esses problemas com colisões são tratados de outra forma. Esse tipo de implementação é ideal para universos grandes, segundo Cormen et al. (2002) e por essa razão são ideais para a implementação de dicionários eletrônicos.

Todos os elementos que se colidem na mesma posição da tabela são colocados em uma lista encadeada (gerando *buckets*). A posição da tabela aponta para o primeiro elemento na lista e se não houver elementos a posição é NIL. Portanto, para a implementação inicial deste trabalho e considerando a estrutura do léxico selecionada foi escolhida a Tabela *Hash* Encadeada [Cormen et al. 2002]. Essa escolha se justifica pelo fato de não haver necessidade de recálculos de posicionamento e também por inferir-se que, dessa forma, haveria um ganho de tempo na busca e inserção.

Os conceitos de Tabela *Hash* até aqui elucidados são, como já mencionado, conteúdo da disciplina de Estrutura de Dados nos cursos de Ciência da Computação que conta com outros conteúdos de mesma complexidade. Para aperfeiçoar o aprendizado dos discentes em matérias com esse grau de dificuldade, alternativas como a utilização de instrumentos midiáticos educacionais disponíveis na internet podem auxiliar os professores e discentes no processo de ensino e aprendizagem para amenizar o alto índice de reprovação nas disciplinas de ED [Souza et al. 2011]. Na tentativa de novas possibilidades no processo de ensino e aprendizagem em Estruturas de Dados, este trabalho vislumbra alternativas para a construção de novos léxicos e assim tentar auxiliar nesse contexto educacional.

Outra disciplina que há grandes esforços na construção de dicionários e léxicos computacionais é a do Processamento de Linguagem Natural (PLN). Esses são criados geralmente de forma manual para posteriormente ter um tratamento digital [Machado 2015] e, muitas vezes, dependendo da maneira ou modelo de tratamento das bases lexicais, pode acarretar em perda de desempenho nas buscas [Greghi 2002]. Uma alternativa são as Tabelas *Hash* que são consideradas eficientes para acessar léxicos grandes de modo rápido e otimizado [Specia e Rino 2002].

Os léxicos e os sistemas de PLN são de grande valia, quando bem projetados [Wittman e Ribeiro 1998]. Assim, diversas universidades desenvolvem pesquisas sobre a análise morfológica contando com vários grupos de pesquisa sobre o léxico da língua geral e das chamadas línguas de especialidade - domínios técnicos, científicos e especializados [Barros e Isquerdo 2010]. Um exemplo é o Léxico da Emoção [Rigo et al. 2013] que buscou identificar e catalogar os sentimentos como apoio da atividade docente.

Com base nessa contextualização, o objetivo deste trabalho é identificar de que forma o *software* desenvolvido Visual TaHs (acrônimo para Tabelas *Hash*s) poderá contribuir para o ensino de Estrutura de Dados, com base na abordagem da contextualização da tabela *Hash*. Essa temática é essencial no curso das Ciências da Computação e áreas afins [Barbosa e Parreira Júnior 2013], [Silva 2011].

Este artigo apresenta três seções, onde a segunda descreve os encaminhamentos metodológicos; a terceira seção exibe o experimento; e a quarta e última seção apresenta as considerações finais.

2. Encaminhamentos Metodológicos

Este trabalho foi inspirado no Léxico das Orquídeas [Lisboa e Barbosa 2013] que tem catalogadas 130 espécies dessas. Porém, em nosso trabalho, para o desenvolvimento do *software*, utilizou-se do livro intitulado "Léxico das Ervas" [Kothe 2009] que descreve a história das ervas, seus primeiros usos, a maneira correta de secagem e o modo como guardá-las. Ao todo foram descritas 105 espécies de ervas contidas no referido livro, no qual cada erva tem 12 atributos que são seu Nome Científico, Família, Outras Denominações, Sinônimos, Origem, História, Floração, Características, Habitat, Propriedades, Cozinha, Saúde, além da Imagem da Erva. Na Figura 1 há a representação da erva AGASTACHE FOENICULUM com seus atributos e valores.

Nome Científico	AGASTACHE FOENICULUM	**Família**	Lamiáceas (Lamiaceae)
Outros Nomes	Hissopo-anisado	**Sinônimos**	Agastache anethiodora
Origem	América do Norte	**Floração**	De junho a setembro.
História	Esta planta já era conhecida pelos indígenas norte americanos devido aos seus valores nutritivo e medicinal. Foi introduzida na Europa por apicultores, pois as suas flores constituem uma excelente fonte de alimento para as abelhas.	**Características**	Planta vivaz, resistente no Inverno, cujo caule pode atingir 90 cm de altura. Possui grandes inflorescências de cor púrpura e folhas verde-prateadas, com comprimento até 8 cm, sabor adocicado e cheiro a anis.
Habitat	Esta planta cresce bem em vasos ou no jardim, em sitios expostos ao sol ou em locais semi-sombrios, com solo permeável. Na Europa Central não necessita de ser protegida da geada.	**Cozinha**	As folhas frescas podem ser utilizadas na preparação de saladas e na produção de licores; as flores são um excelente elemento decorativo em saladas e sobremesas.
Propriedades	Destacam-se os efeitos anti-inflamatórios e digestivos. As folhas devem ser colhidas antes de a planta florir, e podem ser utilizadas frescas ou secas.	**Imagem**	
Saúde e Cosmética	A partir das folhas secas pode-se fazer um chá que alivia dores de garganta, constipações e que acalma o estômago. As flores, frescas ou secas, servem de decoração em arranjos florais.		

Figura 1. Estrutura Léxico das Ervas

Fonte: [Kothe 2009]

Todos os atributos das ervas descritos na Figura 1 foram cadastradas em um Banco de Dados MySQL e depois exportados para um arquivo XML, a fim de permitir ao *software* uma certa flexibilidade. Esse arquivo XML é utilizado sempre que a aplicação for executada, adicionando todos os elementos na Tabela *Hash*. Ao final, caso haja alterações, o arquivo ERVAS.XML é atualizado. Vale salientar que essa estrutura de arquivos XML permite universalizar qualquer tipo de léxico futuro que o usuário quiser aplicar ou visualizar com tabelas hash.

Em relação aos atributos, esses foram criados com o tipo *string*, sendo desconsiderado o item imagem. Especificamente sobre as imagens foram salvas em uma pasta à parte e somente referenciados seus caminhos na tabela, para que possam ser exibidas de forma mais rápida durante a consulta. Também é importante destacar que algumas ervas não possuíam certos atributos como a história, o sinônimo e a saúde,

situação em que os respectivos campos ficam em branco quando da busca.

Para a chave de entrada na Tabela *Hash*, utilizou os nomes científicos das ervas por ser tratar de um nome único e que não se repete em todas as espécies. Exemplos: ACHILLEA MILLEFOLIUM, ALLIUM CEPA, ALLIUM URSINUM, VERBENA OFFICINALIS.

Para a ferramenta Visual TaHs, além da catalogação do léxico, foram implementadas 14 funções *hash*, das quais 7 delas são exibidas na Tabela 1 e são encontradas em literaturas de disciplinas de Estrutura de Dados e de Compiladores, porém outras 7 funções (Tabela 2) foram projetadas por Robert Jenkins (1997), que é um pesquisador da computação e autor de várias funções hash.

Tabela 1. Funções Hash comumente utilizadas

Função Hash	Detalhamento da função
Universal [Cormen et al. 2002]	Função hash = **((a*k+b) mod p) mod t** onde: • "k" – chave da tabela (nome científico da erva); • "p" - número primo que deve ser escolhido e maior que a chave; • "a" e "b" - valores aleatórios inteiros e menores o número primo; • "t" – tamanho da tabela hash.
Divisão [Cormen et al. 2002]	Função hash = **k mod m** onde: • "k" – chave da tabela (nome científico da erva); • "m" – tamanho da tabela hash.
Multiplicação [Cormen et al. 2002]	Função hash = **[t(k*A mod 1)]** • Primeira Etapa = chave da tabela "k" (nome científico) multiplicada por uma constante "A" num intervalo $0 < A < 1$; • Segunda Etapa= resultado de "k * A" é multiplicado pelo tamanho da tabela (t).
Dobra [Silberschatz et al. 2011]	Nesta função é necessário transformar o valor da chave (o nome cientifico da erva) em um valor numérico. Com esse valor, o método realiza uma dobra "como se fosse uma folha de papel", de maneira que os dígitos se sobreponham, sendo por meio de soma ou do operador xor (utilizando números binários).
AHO [Aho, Sethi e Ullman, 1995]	Função hash = **α*h+(Convert.ToInt32(texto[i]))** • Determinar um inteiro positivo para a variável *h*, transformando a cadeia de caracteres em números e realizando a soma desses. • Multiplicar o valor antigo de "*h*" por uma constante α antes de adicionar o próximo caractere. • O valor de hash é o resto de h mod m, sendo *m* um número primo e o tamanho da tabela.
Zobrist [Zobrist, 1970]	Uma matriz é definida pelo tamanho da chave e do alfabeto e os valores dessa são definidos por: • **Zoo[i,j]=tam_tabela*((Valor_Randômico)/(Valor_Randômico2)+1)**; Para o cálculo da hash, todas as letras da chave são utilizadas individualmente para a operação, sendo empregado o *ou exclusivo* para adicionar a variável soma: • **soma ^=Zoo[i ,Convet.ToInt32(texto[i])]**
PJW *Hash* [Aho, Sethi e Ullman 1995]	Um processo é realizado para cada caractere da chave, que é transformado em um valor numérico. • **hash=(hash≪4)+ Convert.ToUint32(Chave[i]);** • **g=hash & 0xf0000000;** • **hash=hash ^ (g≫24); hash=hash ^ g;**

Tabela 2. Funções Hash Jenkins

Função Hash	Detalhamento da função
Mix-32 bit [Jenkins 1997] Mix-64 bit [Jenkins 1997] *Lookup* 32 bits [Jenkins 2013] *Lookup* 64 bits [Jenkins 2013]	• Realiza um embaralhamento que ocorre para cada conjunto de 12 (versão 32 bits) e 24 (versão 64 bits) bytes da chave; • uma combinação (mistura) é realizada com os registradores; • caso o nome científico da erva (a chave para a hash) tiver um tamanho menor que 12 ou 24 bytes, dependendo da versão, serão realizadas combinações separada de bits; • a versão 64 bits gera um número para a hash maior em relação a versão 32 bits; • Mix 32 e 64 bits realizam o processo de "mistura de bits" em uma função separada de embaralhamento; • Lookup 32 e 64 bits realizam o processo "mistura de bits" na mesma função, logo após o embaralhamento.
CRC [Jenkins 2013] CRC Genérico [Jenkins 2013]	Função hash = **⟦(hash≫8) ^ vetor([(hash & 0xff) ^ texto[i]])⟧** • Utilizam de operações binárias para definição do valor; • Utilizam de um vetor com valores em hexadecimal já definidos; • Os vetores que tornam as funções distintas.
One-at-a-Time [Jenkins 2013]	• Utiliza no cálculo procedimentos para deslocamento de bits; • Realiza para cada caractere da chave (nome científico) o seguinte cálculo: **hash+=Texto[i]; hash+=(hash≪10); hash=hash ^ (hash≫6);** • Após as operações com todos caracteres são realizadas as últimas operações binárias **hash+=(hash≪3); hash=hash ^ (hash≫11); hash+=(hash≪15);**

Visual TaHs tem como objetivo auxiliar no aprendizado do conceito de tabelas *hash* na disciplina de Estrutura de Dados. Com essa aplicação é possível realizar operações de inclusão, alteração, exclusão e busca sobre as ervas. O Léxico das Ervas é exibido na Figura 2.

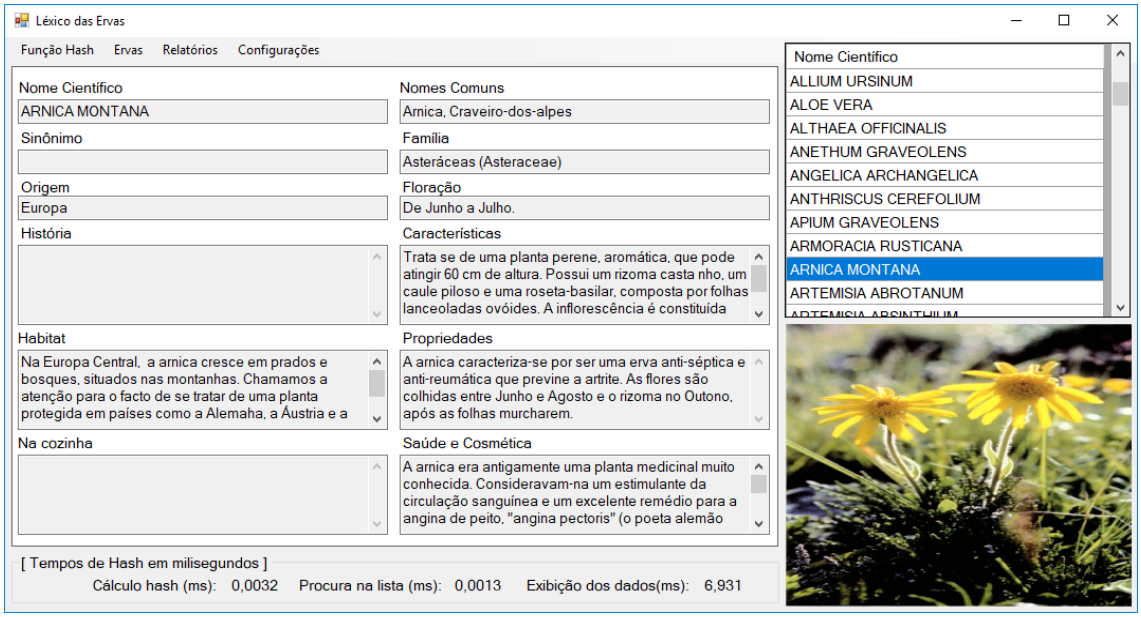

Figura 2. Software Visual TaHs

Também há uma lista de relatórios e gráficos disponibilizados para que o discente possa acompanhar a distribuição dos elementos na tabela e seus tempos de processamento. Barbosa e Parreira Júnior (2013) ressalta nossa capacidade humana de

compreender representações gráficas normalmente ser maior do que apenas com apresentações textuais. Desse modo, com os gráficos, os alunos poderão entender melhor os conceitos e aplicações de Estrutura de Dados.

3. Uso do Visual TaHs no Ensino de Estrutura de Dados

Ferramentas computacionais lexicais que utilizam da Estrutura de Dados do tipo *hash*, precisam levar em conta uma função *hash* perfeita que não altere seus tempos de pesquisas e que tenham sucesso na divisão dos espaços da tabela. Um exemplo de uma aplicação é o *chatbot* Tical [Moreno et al 2017] que utiliza a função de Mix [Jenkins 1997] para buscar em sua base, as respostas das perguntas, e segundo os referidos autores, possui uma ótima performance.

Além de *chatbots*, outras aplicações que utilizam algum tipo de léxico ou precisam de um tempo de resposta menor podem utilizar no seu desenvolvimento esse tipo de estrutura. Por se tratar de um tema de alta complexidade, uma contextualização adequada é fundamental para que não acarrete em um *déficit* na aprendizagem desse conceito e por essa razão Visual TaHs é uma alternativa para contribuir no ensino e aprendizagem.

Com o Visual TaHs, o docente poderá contextualizar sua aula demonstrando na prática o que é uma Tabela *Hash*. Os discentes poderão realizar testes nas 14 funções implementadas e compará-las em relação ao tempo e espalhamento, além de poder alterar o tipo de conteúdo de suas chaves (por exemplo, somente números). Com o código também disponível, os alunos poderão construir também a sua própria função e realizar todas as medições de tempo e espalhamento disponíveis no *software*.

Os relatórios disponíveis no *software* são: Gráfico de Barras, Gráficos de Colisões e Espalhamento, Listagem Ervas X Colisões, Teste de Desempenho de Todas as Funções (Tempos de Cálculo, de Busca, de Exibição), Gráfico de Pizza contendo o espalhamento de todas as funções e a Quantidade de Elementos sem Ervas na Tabela. A Figura 3 ilustra o relatório de Gráficos de Colisão e Espalhamento das 105 ervas.

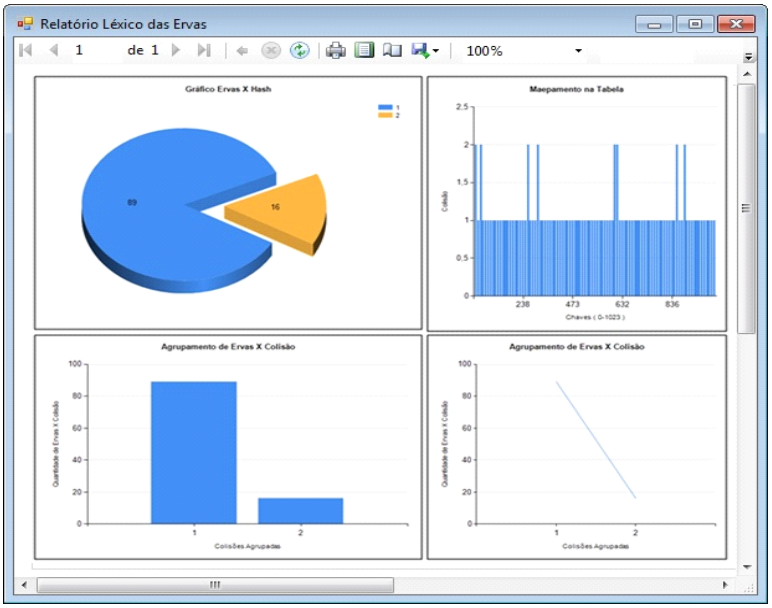

Figura 3. Gráfico de Colisões e Espalhamento

Outras funções importantes para os testes e aprendizado dessa disciplina também foram disponibilizadas. Uma delas é a opção de alterar a quantidade de chaves da tabela, sendo que a configuração padrão está com 1024 chaves (0 ao 1023). Também foi implementada a opção de sobrecarga da função, que consiste em uma função que irá inserir uma quantidade de ervas informada pelo aluno com palavras embaralhadas criadas aleatoriamente, sendo assim possível acompanhar o processo, o espalhamento e desempenho de cada função *Hash*.

Para isso, um experimento foi realizado e detalhado como demonstração do *software* Visual TaHs, sendo uma das maneiras de ser utilizada para auxiliar no aprendizado desse conceito. Neste experimento são realizados testes nas quatorze funções. Essas avaliações são dos espalhamentos, dos tempos de cálculo do *Hash* e de busca dos elementos na tabela. Foi utilizado o tamanho padrão configurado pelo *software* de 1024 posições na tabela, sendo da posição 0 ao 1023. Os testes foram iniciados com 105 ervas e depois uma sobrecarga foi efetivada usando uma proporção de 1000 elementos até contabilizar 10000 ervas, empregando sempre os mesmos elementos para todas as funções. As amostras deste experimento foram realizadas em torno de quatro horas, cujo tempo se deu em razão das coletas dos resultados e geração aleatória de dados (função do *software* que cria aleatoriamente dados para tabela, sem ter a necessidade de digitação). Os resultados obtidos através da aplicação, ilustrados na Figura 4, demonstraram equilíbrio no quesito de ocupação dos espaços da tabela. A única função que não teve um bom aproveitamento de seus espaços é a função de Dobra que ao final do experimento ficou com 940 endereços livres na tabela.

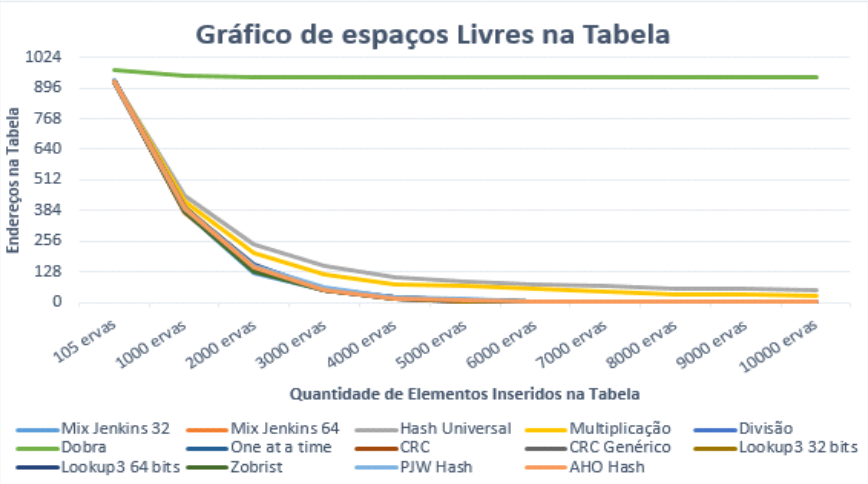

Figura 4. Gráfico dos Espaços Livres na Tabela

Já os tempos de cálculos do número *hash* de todas as funções foram em milissegundos e não apresentaram o mesmo comportamento que foi mostrado anteriormente. Como demonstrado na Figura 5, Dobra, apesar de não conseguir utilizar os endereços adequadamente (Figura 4), realizou mais rápido o cálculo do *hash* do que as funções *One at a Time*, CRC e CRC Genérico, que aproveitaram melhor os espaços da tabela (Figura 4). Já na Figura 5 é possível notar que sete funções foram as melhores e não modificaram seu tempo com o aumento dos elementos. Essas funções foram a Mix Jenkins 32 bits, Mix Jenkins 64 bits, *Lookup* 32 bits, *Lookup* 64 bits, Zobrist, PJW e a função de AHO.

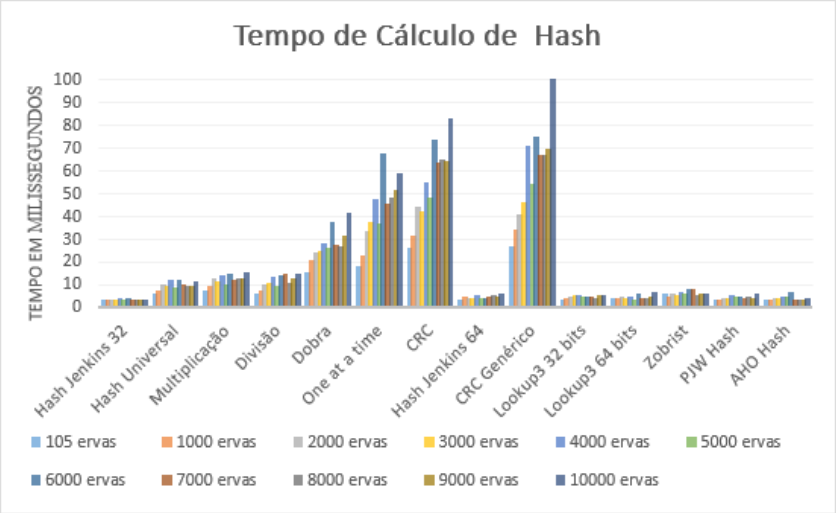

Figura 5. Gráfico Tempo de Cálculo *Hash*

Os tempos de busca pelos elementos na tabela não tiveram influência no tempo final e, portanto, não alteraram a disposição do gráfico apresentado na Figura 5. Através do Visual TaHs também é possível visualizar a disposição (espalhamento) dos elementos da tabela, podendo isso, ser contemplado no momento do entendimento do conteúdo em sala de aula. Na Figura 6 está ilustrada a função Mix 32 bits (uma das melhores funções), Dobra (a função com pior espalhamento) e CRC Genérico (a função com o pior tempo).

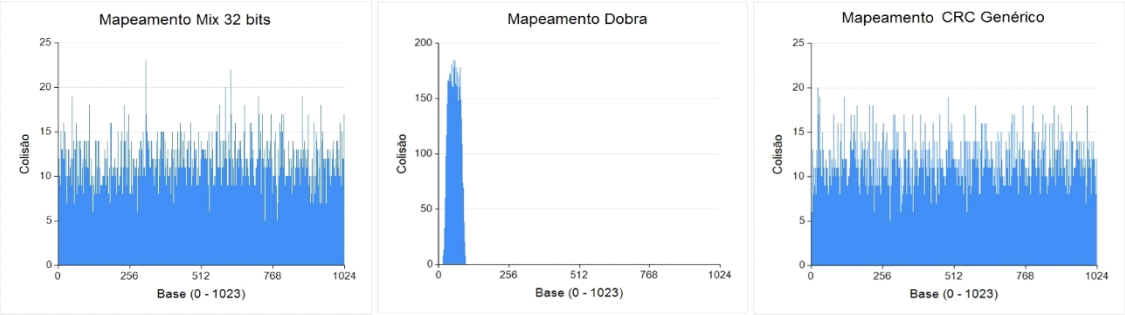

Figura 6. Gráfico do Espalhamento

Com esse experimento foi possível identificar quais funções são mais eficientes para o Léxico das Ervas [Kothe 2009] e também conferir os espalhamentos de cada função, sendo que esses não são um fator determinante para o tempo final, diferente do cálculo do *hash* que é a principal causa para uma busca rápida ou mais lenta.

Esses experimentos que foram realizados representam uma maneira de enfatizar todo o processo de aprendizagem que o "Visual TaHs" pode oferecer aos alunos de Estruturas de Dados. Através das ações de inserção, alteração, exclusão, sobrecarga, relatórios e gráficos, os discentes poderão realizar outros experimentos utilizando dos conceitos adquiridos na disciplina de ED e acompanhar todo o processo de espalhamento e de tempo das funções *Hash*.

Por ser uma ferramenta de código aberto, os discentes poderão estudar cada uma das funções e entender seu funcionamento. Eles poderão alterar as funções *hash* existentes, inserir outras ou criar novas e depois comparar seus desempenhos. O experimento demonstrado neste artigo foi realizado somente com texto e com tamanho

das chaves até vinte caracteres. Entretanto, outros experimentos com números de tamanhos de chaves maiores, variação do tamanho da tabela e outras quantidades de elementos podem ser realizados como forma de auxiliar na aprendizagem desse conceito.

4. Considerações Finais

Uma das principais dificuldades dos alunos de Ciências da Computação é assimilar os conceitos de Estruturas de Dados e fazer as implementações dos algoritmos [Santos e Costa 2005]. Um dos motivos para essa realidade é a dificuldade encontrada tanto por professores quanto por alunos em abordar todo o conteúdo dessa disciplina de maneira prática, pela razão de haver grande quantidade de assuntos e diversos conceitos de algoritmos que são, em sua maioria, complexos e abstratos [Souza et al. 2011]. Uma das alternativas descritas por Souza et al (2011) é encontrar ferramentas que explorem essa prática.

Nesse contexto, o objetivo deste trabalho foi apresentar o *software* Visual TaHs, que é uma aplicação prática de Estrutura de Dados capaz de auxiliar na contextualização da explicação em sala de aula e também ajudar no processo de ensino e de aprendizagem dessa disciplina. Isso é de extrema importância, como enfatizam Boticki et al. (2012) e Barbosa e Parreira Júnior (2013), pois o ensino de ED com representações gráficas pode facilitar o processo de aprendizagem, visto que, normalmente as figuras podem tornar os conteúdos mais compreensíveis. Visual TaHs é diferenciado no contexto de ensino, por ser um sistema que insere, altera, exclui e exibe relatórios utilizando uma estrutura de dados, mostrando na prática uma aplicação com um conceito que muitas vezes só é visto na teoria. O *software* desenvolvido pode proporcionar um novo leque no que diz respeito à perspectiva apontada na pesquisa de Barbosa e Parreira Júnior (2013), com a qual identificaram que a tabela *Hash* foi pouco discutida pela comunidade científica nos últimos vinte anos, de acordo com revisão feita por esses autores.

Esta pesquisa também apresentou o Léxico das Ervas que aplicado à tabela *hash* pode ser uma alternativa de implementação para novos dicionários. Como trabalhos futuros almeja-se inserir a opção para realizar testes de desempenhos em diferentes tipos de léxicos, como por exemplo, no Léxico das Orquídeas.

Referências

Aho, A. V., Sethi, R. e Ullman, J. D. (1995). *Compiladores, princípios, técnicas e ferramentas*, Rio de Janeiro: Guanabara Koogan.

Barbosa, W. A. e Parreira Júnior, P. A. P. (2013). Um Mapeamento Sistemático sobre Ferramentas de Apoio ao Ensino de Algoritmo e Estruturas de Dados. In: *II Congresso Brasileiro de Informática na Educação e XXIV Simpósio Brasileiro de Informática na Educação*. Campinas - SP. p. 406-415.

Barros, L. A. e Isquerdo, A. N. (2010). *O léxico em foco*. São Paulo: Cultura Acadêmica.

Bob Jenkins (2013). Bob Jenkins *hash* function web page. Paper published in *Dr Dobb's journal*. http://burtleburtle.net/bob/*hash*/doobs.html

Botelho, F. C. (2004). *Estudo Comparativo do Uso de Hashing Perfeito Mínimo*.

Dissertação de Mestrado. Belo Horizonte: Universidade Federal de Minas Gerais.

Boticki, I., Barisic, A., Martin, S. and Drljevic, N. (2012). Sortko: Learning Sorting Algorithms with Mobile Devices. In: *IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education*, p. 49-56.

Cercone, N. (1988). Finding and applying perfect hash functions. In: *Applied Mathematics Letters*, v.1, n.1, p. 25-28.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. e Stein, C. (2002). *Algoritmos* – Teoria e Prática. Rio de Janeiro: Campus.

Costa, F. P. B., Correa, S. C. S. e Freitas, R. C. (2014). Recursos Educacionais Abertos sobre IPv6: um processo de construção e de avaliação de qualidade. In: *XXXIV Congresso da Sociedade Brasileira de Computação e XXII Workshop sobre Educação em Computação*. Brasília – DF, p. 1336-1345.

Greghi, J. G. (2002). *Projeto e desenvolvimento de uma base de dados lexicais do português*. Dissertação de Mestrado. ICMC da Universidade de São Paulo.

Jenkins, B. (1997). Algorithm Alley - What makes one hash function better than another? Bob knows the answer, and he has used his knowledge to design a new hash function that may be better than what you're using now. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, v. 22, n. 9, p. 107-110.

Kothe, H. W. (2009). *Léxico das Ervas*. 1.ª Edição, Lisboa: Dinalivro.

Lisboa, A. R. B. S. and Barbosa, C. R. S. C. de. (2013). Lexicon of Orchids. Procedia Social and Behavioral Sciences. In: *5th International Conference on Corpus Linguistics*, v.95, p. 81-88.

Machado, A., Longhi, M., Nunes, M. A. S. N. e Pardo, T. (2015). Personalitatem Lexicon: Um Léxico em Português Brasileiro para Mineração de Traços de Personalidade em Textos. In *IV Congresso Brasileiro de Informática na Educação e XXVI Simpósio Brasileiro de Informática na Educação*, Maceió - AL, p. 1122-1126.

Moreno, F. C, Manfio, E. R, Barbosa, C. R. S. C. de. e Brancher, J. D. (2015). Tical: Chatbot sobre o Atlas Linguístico do Brasil no WhatsApp, In *IV Congresso Brasileiro de Informática na Educação e XXVI Simpósio Brasileiro de Informática na Educação*, Maceió - AL, v. 26, n.1, p. 279-288.

Rigo, S. J., Alves, I. M., Gazola, O., Belau, F., Barbosa, J. L. e Costa, C. (2013). Abordagem linguística para identificação da dimensão afetiva expressa em textos de Ambientes Virtuais de Aprendizagem–um Léxico da Emoção. In: *II Congresso Brasileiro de Informática na Educação e XXIV Simpósio Brasileiro de Informática na Educação*. Campinas–SP, p.738-747.

Santos, R. P. e Costa, H. A. X. (2005). TBC-AED e TBC-AED/WEB: Um Desafio no Ensino de Algoritmos, Estruturas de Dados e Programação. In *IV Workshop em Educação em Computação e Informática do Estado de Minas Gerais*.

Silberschatz, A., Korth H. F. and Sudarshan S. (2011). *Database system concepts*. Vol. 6. New York: McGraw-Hill.

Silva, G. C., Ré, R., Kawamoto, A. e Schwerz, A. (2011). Uma Experiência na Aplicação de Práticas de Apoio no Ensino- Aprendizado de Algoritmos. In *XVII*

Workshop sobre Informática na Escola e XXII Simpósio Brasileiro de Informática na Educação. Aracajú - SE, p. 1378-1381.

Soares, T. C. A. P., Cordeiro, E. S., Stefani, I. G. A. e Tirelo, F. (2004). Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não-Intrusiva de Algoritmos. In: *III Workshop de Educação em Computação e Informática do Estado de Minas Gerais*, p. 1-14.

Souza, C. C., Medeiros, T., Gadelha, R., Sousa, T. D. N., Silva, E. L. e Azevedo, R. R. (2011). Um Ambiente Integrado de Simulação para Auxiliar o Processo de Ensino/Aprendizagem da Disciplina de Sistemas Operacionais". In: *XVII Workshop sobre Informática na Escola e XXII Simpósio Brasileiro de Informática na Educação*. Aracaju – SE, p. 406-414.

Specia, L. e Rino, L. H. M. (2002). *O desenvolvimento de um léxico para a geração de estruturas conceituais UNL*. Série de Relatórios Técnicos do NILC, NILC-TR-02-14. São Carlos, Setembro, 25p.

UFPE-PROPLAN (2018) *Disciplinas que mais reprovam na UFPE*: 2014. https://www.ufpe.br/documents/38954/1317627/Relat%C3%B3rio+Disciplinas+que+mais+reprovam+2014+a+2017.pdf/f1ba5ea6-dbdb-4edd-b5e7-9b993f0ee838.

Wiedenbeck, S., LaBelle, D. and Kain, V. N. (2004). Factors affecting course outcomes in introductory programming. In: *16th Annual Workshop of the Psychology of Programming Interest Group*, p. 97-110.

Wittman, L. H. e Ribeiro, R. D. (1998). Recursos lingüísticos e processamento morfológico do Português: o PALAVROSO e o projeto LE-PAROLE. In: *III Encontro para Processamento Computacional da Língua Escrita e Falada*, PUCRS, Porto Alegre - RS, p. 109-117.

Ziviani, N. (2004). *Projeto de Algoritmos com implementações em Pascal e C*. 2ª Edição. São Paulo: Pioneira Thomson Learning.

Zobrist, A. L. (1970). A new hashing method with application for game playing. *ICCA journal*, v.13, n.2, p. 69-73.