Implementação do algoritmo de busca em profundidade em um grafo: Utilização da Biblioteca Padrão do C++ - STL

[1]Vanessa Felisberto Bilésimo, [1]Patrícia José Porfírio, [1]Rosangela Westphal da Silva, [2]Priscyla Waleska Targino de Azevedo Simões, [2]Silvana Campos de Azevedo

[1]Acadêmico do Curso de Ciência da Computação - Departamento de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma / SC
[2]Professor do Curso de Ciência da Computação - Departamento de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma / SC

vanessabilesimo@gmail.com, {patpantera, rosawestphal}@hotmail.com, {pri, aze}@unesc.net

Resumo: *O presente artigo refere-se ao trabalho interdisciplinar de Teoria dos Grafos e Estrutura de Dados, disciplinas do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, que objetivou o desenvolvimento uma nova forma de resolução do algoritmo de Busca em Profundidade em grafos, sendo desenvolvido com a utilização do container list da Biblioteca Padrão de Gabaritos STL e representado por meio de multilistas.*

1. Introdução

No 1º semestre letivo de 2005 foi proposto pelas disciplinas de Teoria dos Grafos e Estrutura de Dados II do curso de Ciência da Computação da Universidade do Extremo Sul catarinense – UNESC, um trabalho interdisciplinar cujo objetivo foi o desenvolvimento do algoritmo de Busca em profundidade usando listas de listas e STL. Nas próximas seções serão apresentados alguns conceitos de Teoria dos Grafos; STL aplicada a Estrutura de Dados; algoritmos de busca: profundidade e largura; container list. Serão mostrados os resultados obtidos e por fim as considerações finais obtidas junto ao término deste pôster.

2. Teoria dos Grafos

Conforme Villas (1993), "A teoria dos grafos é a parte da matemática dedicada a estudar as relações entre entidades (objetos), que possuem características relevantes. Essa teoria engloba todas as estruturas de dados apresentadas."

Um grafo é uma estrutura G(A, B), onde A é o conjunto finito de nós ou vértice que compõem G e B são os elementos definidos como arcos ou arestas válidos para G, estes elementos são definidos através de uma função que deriva de A. O grafo pode ser dirigido e não-dirigido. Os dirigidos são aqueles que os arcos e as arestas indicam sentido da conexão, já os não-dirigidos são aqueles que os arcos indicam uma conexão dupla [Lopes 1999].

2.1 Algoritmos de Busca

São aqueles que realizam um caminhamento para explorar um grafo examinando todos os seus vértices e arestas. Um algoritmo eficiente é aquele que visita todos os vértices e arestas em tempo proporcional ao seu número Goodrich (2002). A seguir temos dois tipos de algoritmos de grafos não-dirigidos:

- Busca em Amplitude ou Largura: trabalha dividindo os vértices em níveis, sendo que todos os vértices de menor distância são percorridos antes dos vértices de maior distância.

- Busca em Profundidade: resulta numa seqüência de vértices, onde cada um é adjacente ao próximo vértice.

2.1.1 Busca em Profundidade

O algoritmo de Busca em Profundidade resulta numa seqüência de vértices, onde cada um é adjacente ao próximo vértice. Com o caminhamento em um grafo, procura-se visitar um conjunto de vértices, iniciando sempre por um dado vértice.

O caminhamento em profundidade inicia por um vértice V e, então visita o primeiro vértice Vi, adjacente a V. Em Vi visita-se o primeiro Vj adjacente a Vi, e assim por diante, visitando-se depois o segundo vértice adjacente a Vi, até que todos os vértices alcançáveis a partir de V tenham sido visitados.

É uma forma sistemática de como caminhar sobre os vértice e arestas do mesmo. Se o grafo é uma árvore uma das formas mais simples de se fazer a busca em profundidade do grafo é com a função recursiva Lopes (1999). Como descrito na figura 1.

```
Algoritmo profundidade recursiva(n)
Inicio
        Visitar_nó (n)
        Marcar_nó (n)
        Para cada nó m marcado não visitado adjacente a n faça
                Se (nómarcado (m) = "F" ) então
                        profundidade recursiva(m)
                Fim se
        Fim para
Fim.
```

Figura 1. Algoritmo de busca em profundidade recursiva
Fonte: Moraes (2001)

Com o conhecimento de várias estruturas de dados pode-se estudar com mais interesse e entendimento a teoria dos grafos. Os grafos na maioria das vezes são implementados com matriz de adjacência, destacando-se também listas de listas, e vetor de listas [Drozdek 2002].

3. STL aplicada a Estrutura de Dados

STL (*Standard Template Library)*, ou seja, Biblioteca Padrão de Gabaritos, é uma coleção de bibliotecas escritas na linguagem C++, conforme o padrão ANSI/ISSO de 1994.

É de extrema importância trabalhar com algoritmos inclusos nesta biblioteca, que auxiliam nos mais variados tipos de estrutura de dados, entre elas pilhas, filas e muitas outras estrutura de dados padrão (Savitch, 2004).

3.1 Eficiência da STL

Na busca de soluções para determinados problemas focados no desenvolvimento de software, a STL permite uma economia de tempo e esforço consideráveis de programação, além de obter, como resultado final, softwares de maior qualidade. Com a STL o programador não perde tempo na construção de componentes que já existem na

biblioteca, algoritmos e estrutura de dados, que já estão implementados e otimizados ao máximo, podendo ser reutilizados [Frohlich 2002].

3.2 Container LIST

Dentre os vários containeres que a STL disponibiliza, encontra-se o List, que é um container seqüencial que fornece implementação eficiente para operações de inserção e remoção em qualquer posição. Como descreve Deitel (2001, p. 937) "a classe list é implementada como uma lista duplamente encadeada, isto é, cada nodo na list contém um ponteiro para o nodo anterior e para o nodo seguinte."

Desta maneira a classe list tem capacidade para suportar iteradores bidirecionais, fazendo com que o container seja acessado de várias formas e extremidades [Deitel 2001].

4. Estudo de Caso

Para atingir o objetivo proposto foram seguidas algumas etapas metodológicas: estudo do algoritmo de Busca em profundidade,a estrutura de dados lista da STL; desenvolvimento do algoritmo recursivo de Busca em profundidade utilizando listas de listas na STL; realizar o teste do algoritmo desenvolvido por meio de uma implementação em C++; ao final foram analisados os resultados obtidos bem como a integração desse trabalho na disciplinas de Teoria dos Grafos e Estrutura de Dados II.

4.2 Desenvolvimento e implementação do Algoritmo

A implementação do algoritmo foi feita na linguagem de programação C++ a qual oferece suporte a STL list. Este trabalho foi realizado nos meses de maio e junho de 2005. No desenvolvimento desta etapa buscou-se trabalhos semelhantes que abordassem o assunto proposto, foram encontrados muitos exemplos nacionais e internacionais sobre list, alguns exemplos internacionais sobre listas de listas na STL, e sobre a temática do trabalho foi encontrado referências somente com a implementação de matriz de adjacência.

4.3 Pseudocódigo do Algoritmo de Busca em Profundidade

Este pseudocódigo foi desenvolvido pelas autoras do artigo, propondo-se a resolução da Busca em Profundidade utilizando listas de listas e o container List da STL..

```
Algoritmo profundidade
                Escolhe-se um vértice inicial
                Enquanto o vértice inicial não estiver marcado como visitado.
                        Caminhe até achar a conexão do vértice inicial
                                Se conexão do vértice inicial não estiver marcada
                                como visitada
                                Empilha vértice inicial e sua conexão em um
                        conjunto T
                                Marca vértice inicial como visitado
                                Caminhe por todo o grafo marcando vértice inicial
                                como visitado
                                Vértice inicial recebe sua conexão
```

Fim do Se Senão
Marca vértice inicial como visitado
Caminhe por todo o grafo marcando vértice inicial como visitado
Fim do Senão
Verifica se tem mais conexões do primeiro vértice inicial.

O processo termina quando o vértice procurado for encontrado ou quando a pilha estiver vazia e todos os vértices tiverem sido visitados.

5. Considerações Finais

Com a integração das disciplinas de Teoria dos Grafos e Estrutura de Dados II percebeu-se que a prática da teoria vista em sala de aula foi bem aproveitada para que assim se pudesse fazer o trabalho com mais facilidade.

No desenvolvimento do algoritmo tentou-se solucionar o problema com uma solução recursiva, mas porém, conseguiu-se desenvolver uma solução não recursiva, pela facilidade encontrada na hora de desenvolver esse algoritmo, que trás as mesmas eficiências do algoritmo recursivo.

Na implementação em geral, apesar de poucas referências ao término deste trabalho atingiu-se o objetivo proposto, ou seja, Implementação do algoritmo de busca em profundidade em um grafo: Com a utilização da Biblioteca Padrão do C++ - STL.

Recomenda-se para trabalhos futuros desenvolver uma solução recursiva, otimizar a solução encontrada, por meio de projetos de pesquisa, trabalhos interdisciplinares ou até mesmo trabalho de conclusão de curso que contribuem para consolidação de conteúdos vistos em sala de aula.

6. Referências

Deitel, H.M; Deitel, P. J. (2001) **C++: Como programar**. 3.ed. Porto Alegre: Bookman, 1098.

Drozdek, Adam. (2002) **Estrutura de dados e algoritmos em C++**. São Paulo: Thomson.

Goodrich, Michael T.; Tamassia, Roberto. (2002) **Estruturas de dados e algoritmos em** Java. 2. ed Porto Alegre: Bookman.

Lopes, Arthur Vargas. (1999) **Estruturas de dados: para a construção de software**. Canoas: ULBRA. v.1.

Savitch, Walter J. (2004) **C ++ Absoluto**. São Paulo: Addison-Wesley.

Villas, Marcos Vianna. (1993) **Estruturas de dados : conceitos e técnicas de implementação**. Rio de Janeiro: Ed. Campus.

Fröhlich, A. A.(2002) "Programação Genérica**",** http://www.inf.ufsc.br/~guto/teaching/sce/ine5612-2001-2/work/gp.html ,Maio.