Documentação do Trabalho

Arthur Alexsander Martins Teodoro[1]
Saulo Ricardo Dias Fernandes[1]

[1]Instituto Federal de Minas Gerais - Campus Formiga

Resumo

Tal documento tem a finalidade de explicar abordagens e estruturas realizadas na implementação da TAD(Tipo Abstrato de Dados) que implementa grafos e a implementação de um algoritmo de menor caminho utilizando a TAD implementada como trabalho interdisciplinar das disciplinas Programação II e Algoritmos e Estruturas de Dados I.

1. Introdução

A proposta de realização deste trabalho interdisciplinar é a implementação de uma TAD que implementa o tipo Grafo e o uso de tal TAD para implementação de um algoritmo que calcula o menor caminho entre dois vértices de um grafo dirigido com arestas ponderadas.

2. Implementação

Nesta seção deste documento será mostrada as escolhas de implementação, demonstração da estrutura de dados, e demais questões de implementação

2.1. Descrição sobre as decisões de projeto e implementação

Para a implementação da TAD e do algoritmo de caminho mínimo, a implementação foi feita em dupla, nenhum integrante ficou responsável por partes específicas. Foi usada a linguagem de programação C para a implementação de tal trabalho. Ambos usaram o editor de texto Sublime Text 3[1] para a codificação, o compilador GCC versão 4.8.4, ambos usaram sistema operacional GNU/Linux, distribuição Ubuntu 14.04 e para controle de versão foi usado o método Git versionado no Bitbucket[2]

2.2. Formato de entrada e saída de dados

Como o objetivo do trabalho era a implementação da TAD de grafo e o algoritmo de caminho mínimo, não é lido nenhum dado via teclado e só é mostrado na tela o caminho mínimo gerado pelo código.

2.3. Como executar o código

Como se trata de uma TAD, deve existir um arquivo com a função main que possui o include ”grafo.h”. Tal comando incluirá na arquivo main as funções desenvolvidas no trabalho. Também foi escrito um arquivo Makefile que, pode ser executado usando o comando make via terminal.

[1]`https://www.sublimetext.com/`
[2]`https://bitbucket.org/`

3. Detalhamento do trabalho

Nesta seção será mostrado de forma mais detalhada a estrutura de dados criada e algumas funções desenvolvidas tanto sobre a TAD quanto sobre o caminho mínimo.

3.1. TAD Grafo

A TAD de grafo foi disponibiliza pelos professores das disciplinas para qual este trabalho foi desenvolvido e o objetivo dos integrantes era implementar todas as funções que, quando terminado, implementaria um grafo tanto direcional e não-direcional.

3.1.1. Estrutura de Dados usada

A estrutura de dados criada possui dois estruturas, uma que representa as arestas do grafo e a outra que representa o grafo como um todo. O diagrama da estrutura grafo pode ser vista na Figura 1

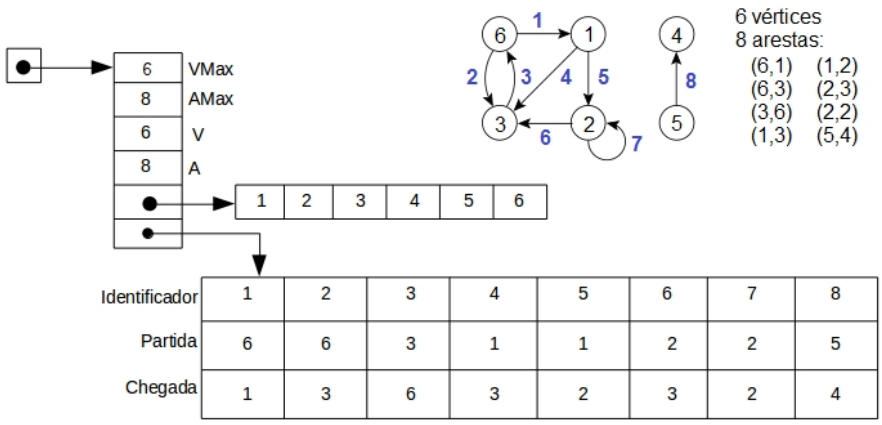

Figura 1. Estrutura de Dados criada

A estrutura mostrada na Figura 1, existe um campo que representa o número máximo de vértices, o número máximo de arestas, a quantidade atual de vértices e arestas, um ponteiro para inteiro que armazena os identificadores dos vértices e um ponteiro para a struct aresta que armazena todos os dados das arestas criadas. Foi escolhida usar vetores para armazenamento dos dados uma vez que a quantidade de vértices e arestas já era definido não era necessário a criação de uma estrutura dinâmica. As estruturas criadas e suas definições de tipo pode ser visto abaixo.

Listing 1. Estruturas e Tipos criados

```
struct
{
  int partida;
  int chegada;
```

```
  int identificador;
};
struct grafo
{
  int V;
  int A;
  int aMax;
  int vMax;
  struct aresta *arestas;
  int *vertices;
};
typedef struct grafo *Grafo;
```

3.1.2. Criação do Grafo

A função responsável de criação do grafo é uma de maior dificuldade, uma vez que nela é alocada a memória para a estrutura e para os vetores, sendo esta função de estrema importância, uma vez que sem ela não é possível representar o grafo.

3.1.3. Carrega Grafo de Arquivo Texto

Existe a opção de carregar um grafo de um arquivo texto com formato especificado pelos professores da disciplina. Tal função pode ser vista abaixo:

Listing 2. Função responsável de carregar grafo

```
Grafo GGcarregaGrafo(char *f)
{
  FILE *fp;
  fp = fopen(f,"rt");
  if(fp == NULL)
  {
    return NULL;
  }
  int v1, v2, i = 0;;
  fscanf(fp,"%d%d",&v1,&v2);
  Grafo p = GGcriaGrafo(v1,v2);
  for(i = 0; i < v1; i++)
  {
    GVcriaVertice(p);
  }
  while(fscanf(fp,"%d%d",&v1,&v2) != EOF)
  {
    GAcriaAresta(p,v1,v2);
  }
  fclose(fp);
  return p;
}
```

No decorrer da implementação, foi encontrado a dificuldade com a leitura do arquivo uma que poderia existir no arquivo texto uma linha vazia, fazendo assim com que a ultima aresta seria duplicada. Para resolver isso, foi escrita uma estrutura de repetição

que lia todos os valores até a função *fscanf* retornar o valor de EOF, que representava o final de arquivo.

3.2. Caminho Mínimo

Na especificação do trabalho era pedido a construção de um algoritmo que buscasse o menor caminho entre dois vértices de um grafo direcional ponderado. Para tal abordagem, a dupla escolheu realizar a implementação do Algoritmo de Dijkstra. Tal escolha foi feita com base na maior facilidade para implementação de tal algoritmo.

3.2.1. TAD Lista

O Algoritmo de Dijkstra necessita algumas estruturas de dados para armazenar dados como distâncias entre os vértices, vértices já visitados e vértices pai. Para tal armazenamento foi escolhido entre a dupla a escolha de uma TAD de Lista Simples, uma que tal TAD representa facilmente um conjunto como é representado em diversos livros o local onde tais dados são mantidos.

3.2.2. Algoritmo de Dijkstra

Segundo [da Silva 2014], o funcionamento do Algoritmo de Dijkstra é relativamente simples. É necessário três conjuntos de dados, neste trabalho, guardados em Listas Simples. A Lista contendo os valores de distâncias recebem os valores de ∞ em todos os vértices. Após isso, o primeiro vértice recebe como distância o valor 0. A partir daí, o algoritmo analisa todos as ligações de saída que tal vértice possui, alterando o valor da distância. Após isso, o vértice que teve suas ligações analisadas é colocado na lista de visitados. Após isso, um novo vértice é escolhido para ser analisado com base na sua distância e se está ou não em visitados.

4. Conclusão

Os objetivos do trabalho que eram realizar a implementação da TAD e sua utilização, foram concluídos, uma vez que, usando teste disponibilizados pelos professores, tais foram implementadas corretamente. Além disso, a implementação do trabalho reforçou o entendimento de TAD e melhorou consideravelmente a aprendizagem da linguagem C.

Referências

da Silva, D. M. (2014). Caminho mínimo. Disponível em `https://goo.gl/MoKPL7`. Acessado em 20/06/2016.