TRABALHO 2 DE MÉTODOS DE PROGRAMAÇÃO

ALUNOS: EDUARDO HENRIQUE COSTA MORESI
MATRÍCULA: 14/0019286
PROFESSOR: JAN MENDONÇA CORREA
TURMA: A

16 de maio de 2016

1 Forma de Realizar os Testes

Os testes foram divididos de acordo com a função que eles iriam testar. Para cada caso, houve a inicialização necessária do cenário que o teste necessita para atuar (como leitura de dados do grafo, por exemplo). Para cada função, tem-se uma "suíte de teste" que possuí diversos casos de teste.

Para a realização dos testes foram utilizados diversos arquivos. Esse documento vem acompanhado com um deles, denominado "TestFile.txt". Esse arquivo está setado como "default" na aplicação e caso seja necessário a utilização de outro arquivo a muudança deve ser feita no código.

2 Testes Realizados sobre cada Função da TAD Grafo

2.1 Função "int le_grafo(FILE *input, TpGrafo *grafo)"

Essa função cria um grafo vazio e le os dados do arquivo de entrada no formato especifcado. Em caso de sucesso retorna "EXIT_SUCCESS", caso contrário retorna "EXIT_FAILURE". A seguir, o(s) teste(s) realizado(s) sobre a função.

TESTE: Verifica se ocorre correta leitura do arquivo e criação do grafo:

1. Nome do teste: LeGrafo.
2. O que vai ser testado: O teste consiste em verificar se a função consegue detectar arquivos inválidos e se os processos de leitura e criação do grafo são realizados com sucesso.
3. Entrada: A entrada para esse conjunto de testes é a TAD grafo e suas estruturas, e, dependendo do teste, o arquivo de entrada.
4. Saída: "EXIT_FAILURE" para o teste em que se tenta ler dados de um arquivo inexistente e "EXIT_SUCCESS" quando o arquivo é lido com sucesso. Os demais testes verificam a correta inserção dos vértices e arestas.
5. Critério para ser aprovado: A função deve retornar os resultados esperados para cada teste.
6. Status da função: Aprovada.

2.2 Função "int escreve_grafo(FILE *output, TpGrafo *grafo)"

Essa função escreve os dados presentes na estrutura de dados grafo para um arquivo de saída.

TESTE: Verifica funcionamento na escrita dos dados:

1. Nome do teste: EscreveGrafo.
2. O que vai ser testado: Será testado se a função dada é capaz de escrever os dados da TAD grafo para um arquivo de saída, o qual deve ser válido.
3. Entrada: A entrada para os testes consistem no grafo e no ponteiro para o arquivo de saída. Os testes fazem chamada à função em questão.
4. Saída: Espera-se "EXIT_SUCCESS" no caso em que a escrita é feita com sucesso e "EXIT_FAILURE" no caso em que se tenta escrever em um arquivo que não existe.
5. Critério para ser aprovado: Para ser aprovada a função deve ser capaz de identificar arquivos que podem ter dados escritos neles e realizar essa ação.
6. Status da função: Aprovada.

2.3 Função "int insere_vertex(TpGrafo *grafo, TpVertex vertice, int type)"

Função utilizada para inserir um vértice no grafo, em que o usuário informa através do parâmetro "type" se o vértice é de origem ou não.

TESTE: Verifica se a inserção é feita corretamente:

1. Nome do teste: InsereVertex.
2. O que vai ser testado: Será testado se a função insere vértices corretamente no grafo e se ela consegue detectar quando haverá redundância, ou seja, inserção de um vértice que já existe.

3. Entrada: A entrada para os testes consiste no grafo, vértice e o tipo de vértice que será inserido. Os testes utilizam os resultados da função de inserir vértices.

4. Saída: A saída esperada é "EXIT_FAILURE" caso o vértice dado já exista e "EXIT_SUCCESS" caso a inserção tenha sido realizada com sucesso.

5. Critério para ser aprovado: A função deve ser capaz de inserir um vértice na estrutura do grafo, evitando, porém, inserir vértices que já existam.

6. Status da função: Aprovada.

2.4 Função "int remove_vertex(TpGrafo *grafo, TpVertex *vertice, char *nome)"

Função utilizada para remover os vértices de um grafo. O vértice é retornado pro referência e a busca é feita pelo nome.

TESTE: Verifica se a remoção de vértices é efetiva:

1. Nome do teste: RemoveVertex.

2. O que vai ser testado: Será testado se a função é capaz de retirar um vértice do grafo de acordo com o nome. Ao retirar o vértice, as relacionadas a ele são removidas de forma a manter o grafo consistente.

3. Entrada: A entrada consiste em um ponteiro para o grafo, um ponteiro para o vértice que será retirado e o nome do vértice. Os testes utilizam o retorno da função e os resultados que ela produz.

4. Saída: Espera-se "EXIT_FAILURE" caso se tente remover um vértice que não exite e "EXIT_SUCCESS" caso a ação seja feita com sucesso. Os demais testes verificam se o vértice ainda existe no grafo após a retirada (para passar no teste ele não pode mais existir na estrutura).

5. Critério para ser aprovado: A função deve ser capaz de remover um vértice do grafo caso ele exista e eliminar arestas que o possuam.

6. Status da função: Aprovada.

2.5 Função "int insere_aresta(TpGrafo *grafo, char *origem, char *destino, float peso)"

A função listada insere uma aresta na estrutura do grafo, tendo como parâmetros um endereço para o grafo que será modificado, o nome dos vértices de origem e destino e o peso entre eles. Caso os vértices não existam, a função retorna com "status" de erro.

TESTE: Verifica a inserção de arestas:

1. Nome do teste: InsereAresta.
2. O que vai ser testado: Será testado se a função insere novas arestas no grafo, dado um vértice origem e outro de destino. Se os vértices não existem no grafo, a inserção não pode ser feita. Além disso, a inserção só pode ocorrer se o peso for estritamente positivo e maior do que zero.
3. Entrada: O grafo que será modificado, o nome da origem e destino e o peso da aresta. Os testes irão utilizar o retorno da função de inserção e os resultados gerados por ela.
4. Saída: A função deve retornar "EXIT_FAILURE" caso se queira inserir uma aresta com pelo menos um vértice que não pertence ao grafo. Caso a inserção seja feita com sucesso, deve-se retornar "EXIT_SUCCESS".
5. Critério para ser aprovado: A função deve inserir arestas apenas entre vértices que existem na estrutura do grafo e garantir que o peso entre eles será estritamente positivo e maior do que zero.
6. Status da função: Aprovada.

2.6 Função "int remove_aresta(TpGrafo *grafo, char *origem, char *destino)"

Essa função remove uma aresta do grafo dado o nome da origem da aresta e do destino da aresta. Tanto a origem como o destino devem existir no grafo para que eles possam ser removidos.

Verifica remoção de arestas:

1. Nome do teste: RemoveAresta.

2. O que vai ser testado: O teste irá verificar a função consegue remover arestas de um grafo com sucesso. Não pode haver remoção caso alguns dos vértices (origem ou destino) não exista.

3. Entrada: A entrada consiste em um ponteiro para a estrutura do grafo e o nome dos vértices de origem e destino. Os casos de teste fazem uso dos valores de retorno da função e das consequências geradas por ele.

4. Saída: Espera-se "EXIT_FAILURE" caso algum dos vértices não exista e "EXIT_SUCCESS" caso a remoção tenha sido feita com sucesso. Os demais testes verificam se a aresta realmente foi retirada da estrutura do grafo através de acesso direto.

5. Critério para ser aprovado: A função deve ser capaz de remover arestas apenas no caso em que elas existam.

6. Status da função: Aprovada.

2.7 Função "int calculaDistancia(TpGrafo *grafo, char *origem, char *destino)"

A função irá utilizar a estrutura do grafo para calcular a distância entre um vértice qualquer do grafo (origem) até outro (destino).

Verifica se o cálculo das distâncias é correto:

1. Nome do teste: CalculaDistancia.

2. O que vai ser testado: Será testado se a função calcula a distância correta entre dois vértices, retornando o valor calculado. Em caso de erro, ela deve retornar um código de erro indicando isso.

3. Entrada: A entrada é um ponteiro para o grafo que será usado e o nome dos vértices de origem e destino. O retorno da função é utilizado nos testes para avaliar a sua corretude.

4. Saída: Nos casos em que não existe ligação entre os vértices espera-se que a função retorne "-1". Caso exista um caminho entre os vértices, espera-se a distância entre eles.

5. Critério para ser aprovado: A função deve calcular a distância corretamente entre dois vértices e, caso não exista um caminho entre eles ela deve retornar "-1".

6. Status da função: Aprovada.

2.8 Função "int grafoConsistente(TpGrafo *grafo)"

Função que recebe um grafo e verifica se ele é consistente, ou seja, se o peso das arestas é estritamete posivito e maior que zero e se os vértices das arestas pertencem ao grafo dado.

Verifica se a inconsistência é detectada:

1. Nome do teste: GrafoConsistente.
2. O que vai ser testado: Vai ser testado se a função consegue detectar um grafo inconsistente.
3. Entrada: A entrada consiste no grafo que será analisado. O retorno da função é utilizado nos testes.
4. Saída: A saída esperada é "EXIT_SUCCESS" caso o grafo seja consistente e "EXIT_FAILURE" caso o grafo seja inconsistente.
5. Critério para ser aprovado: Deve-se verificar corretamente a consistência de um grafo bem como a inconsistência de tal estrutura.
6. Status da função: Aprovada.

2.9 Função "int grafoConexo(TpGrafo *grafo)"

Essa função é responsável por verificar se um grafo é conexo, ou seja, se todos os seus vértices são conectados por um caminho.

TESTE: Verifica se o grafo é conexo:

1. Nome do teste: GrafoConexo.
2. O que vai ser testado: Será testado se a função consegue verificar corretamente quando um grafo é conexo e quando um grafo é desconexo.
3. Entrada: A entrada consiste no grafo que será analisado. O retorno da função será usado nos casos de teste.

4. Saída: A saída esperada é “EXIT_SUCCESS” se o grafo for conexo e “EXIT_FAILURE” se o grafo for desconexo.

5. Critério para ser aprovado: A função deve detectar corretamente se um grafo é conexo ou desconexo.

6. Status da função: Aprovada.

2.10 Função “int destroi_grafo(TpGrafo *grafo)”

Função responsável pela destruição das estruturas presentes dentro do grafo.

TESTE: Verifica se o grafo é conexo:

1. Nome do teste: DestroiGrafo.

2. O que vai ser testado: Será testado se a função é capaz de encerrar as estruturas do presentes dentro do grafo.

3. Entrada: A entrada consiste no grafo que será destruído. O resultado da função é analisado nos casos de teste.

4. Saída: A saída esperada é “EXIT_SUCCESS” para a destruição correta do grafo.

5. Critério para ser aprovado: Deve-se destruir o grafo sem vazamentos de memória.

6. Status da função: Aprovada.