Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
Primeiro Semestre de 2014

Trabalho Prático 0 - Consulta em Grafos

Este trabalho prático tem como objetivo familiarizar o aluno com primitivas básicas da linguagem C (estrutura de dados, modularização, alocação dinâmica de memória e compilação do código através da ferramenta Makefile) e padrões de documentação da disciplina.

Este trabalho será útil para o trabalho prático 1. Se implementadas corretamente, as estruturas de dados utilizadas nesse trabalho poderão ser reutilizadas no TP 1.

Problema

Neste trabalho, o aluno deve fazer consultas em grafos utilizando a estrutura de dados *Pilha*. Uma consulta consiste em uma busca em profundidade a partir de um nó inicial até um nó final, terminando quando o nó final é descoberto. A consulta deve retornar o caminho percorrido pela busca em profundidade listando os vértices à medida em que são descobertos (mudam de branco para cinza).

Dado um grafo e um conjunto de consultas, o programa deverá indicar a ordem de visita dos vértices no grafo. As arestas são simétricas. Em caso de empate, seguir a ordem lexicográfica.

A consulta não pode ser implementada recursivamente. É obrigatório utilizar o tipo abstrato de dados *pilhas*.

A representação do grafo e a pilha devem ser alocadas dinamicamente.

Entrada e Saída

O programa deverá solucionar múltiplas instâncias do problema em uma única execução. As arestas a serem inseridas no grafo devem ser lidas de um arquivo de entrada e o resultado do programa deve ser impresso em um arquivo de saída. Ambos arquivos devem ser passados por parâmetro na chamada do executável:

./tp0 input.txt output.txt

O arquivo de entrada possui um inteiro K na primeira linha onde K é o número de instâncias a serem simuladas. Em seguida as K instâncias são definidas da seguinte forma. A primeira linha possui dois inteiros V e E, representando o número de vértices (V) e arestas (E). As próximas E linhas possuem dois números inteiros indicando a aresta. Após isto, haverá uma nova linha contendo um inteiro M, que representa o número de consultas que deve ser feitas no grafo. As próximas M linhas possuem dois números inteiros representando os vértices inicial e final que serão utilizados na consulta.

Para cada instância, deve ser impresso no arquivo de saída todos os vértices visitados. Uma linha em branco deve ser impressa entre cada consulta.

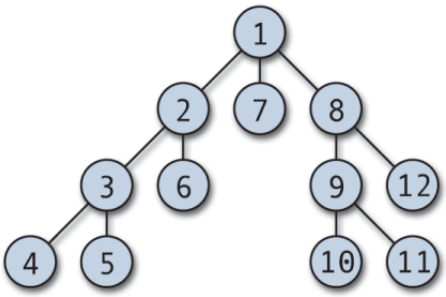

Figura 1: Exemplo do grafo de entrada

Exemplo

A seguir temos um exemplo de funcionamento do programa:

Entrada:

```
1
12 11
1 2
1 7
1 8
2 3
3 4
3 5
2 6
8 9
8 12
10 9
11 9
3
1 12
2 7
4 9
```

Saída:

```
1 2 3 4 5 6 7 8 9 10 11 12

2 1 7

4 3 2 1 7 8 9
```

Entrega

- A data de entrega desse trabalho é **07/03/2014**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixo devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua **nenhuma pasta**. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saída descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da soluço implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significativamente a execução.
  - Espera-se ao menos um gráfico de *Número de Pontos X Tempo de Execução* para diversas instâncias juntamente com sua respectiva análise.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- **Variáveis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanto **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- O arquivo executável deve ser chamado tp0.
- **Legibilidade e boas práticas** de programação serão avaliadas.