Trabalho Prático 1

Algoritmos e Estruturas de Dados III

27 de Setembro de 2018

Entrega: 09/10/2018 até 23:55.

1 Descrição

Nos anos 50, um popular evento de automobilismo surgiu e se tornou o mais popular do mundo. Este evento foi denominado "Auto-cone". Suas corridas não eram transmitidas pela TV antes da década de 70 devido a muitos problemas daquela época. Segundo alguns entusiastas, as corridas naquele tempo eram mais empolgantes que as corridas da atualidade, no qual toda e qualquer manobra pode ser avaliada como arriscada e passível de punição aos pilotos que ocasionarem acidentes.

Sabemos que o evento inicialmente contava com apenas 6 Grandes Prêmios (GPs) e foi evoluindo com o passar dos anos. A corrida inaugural do campeonato aconteceu no circuito de "Silvercex" e contou com participação de montadoras de diversas partes do mundo.

Um historiador quer analisar a fundo as corridas dos anos 50 no "auto-cone", entretanto ele terá que enfrentar diversos problemas. Um dos problemas é a falta de dados sobre as corridas, outro é o não televisionamento do evento naquela época. Como se não bastassem tais problemas, a maioria dos pilotos faleceu ou possui problemas de memória.

A tarefa principal do trabalho é gerar a ordem de classificação das corridas, de acordo com as observações/memórias feitas por cada um dos pilotos entrevistados.

Cada piloto será entrevistado sobre qual piloto ele tem certeza que ficou na sua frente.

2 Definição e Modelagem do Problema

Neste trabalho, nós estamos interessados em encontrar a real classificação dos participantes no torneio através das afirmações/memórias dos participantes.

A rede elaborada através das memórias dos participantes pode ser modelado como um grafo. Seja a tupla $G = (V, A)$ um grafo direcionado não ponderado (sem pesos nas arestas), onde $V = v_1, v_2, v_3, \ldots, v_n$ é um conjunto de vértices que representa os participantes e $A = a_1, a_2, a_3, \ldots, a_m$ um conjunto de arestas entre dois competidores v_i e v_j, onde $v_i \neq v_j$.

Exemplo: Considere o cenário da primeira corrida, onde existiram 5 pilotos e apenas 5 afirmações/memórias sobre tal corrida. Corrida, tal que:

- $Piloto_2$ afirmou que o $Piloto_1$ ficou na sua frente

- $Piloto_3$ afirmou que o $Piloto_2$ ficou na sua frente
- $Piloto_4$ afirmou que o $Piloto_3$ e $Piloto_2$ ficaram na sua frente
- $Piloto_5$ afirmou que o $Piloto_4$ ficou na sua frente

A Figura 1 mostra a sequência de acordo com as memórias alcançadas.

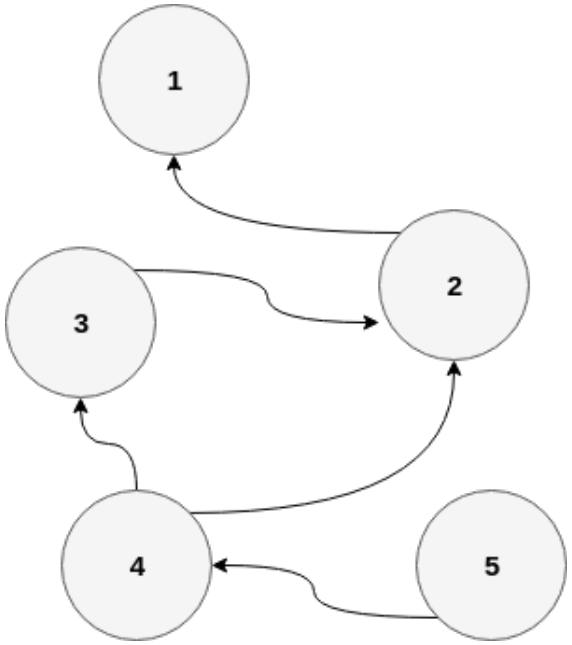

Figura 1: Primeira corrida do evento "auto-cone".

Neste exemplo, está claro que o $piloto_1$ foi o campeão da primeira corrida, o $piloto_2$ ficou em segundo, o $piloto_3$ em terceiro, o $piloto_4$ em quarto e o $piloto_5$ em último.

Observe que nem todas as arestas podem estar disponíveis, pois os pilotos não se lembram bem dos resultados das corridas. Outro aspecto é que os pilotos podem simplesmente se confundirem quanto a sua colocação na tentativa de melhorar a sua classificação ou devido a falta de memória. Em alguns casos, essa confusão/falta de memória pode impossibilitar a recuperação de uma classificação válida para a corrida.

3 Entrada

A primeira linha do arquivo de entrada contém 2 números inteiros P e M, onde P é o número de pilotos ($2 \leq P \leq 5 \cdot 10^7$) e M ($1 \leq M \leq 5 \cdot 10^7$) o número de memórias alcançadas pelo historiador durante as entrevistas aos pilotos vivos. As M linhas seguintes contém tuplas $< piloto_i, piloto_j >$ indicando qual o piloto entrevistado e o piloto que estava na sua frente. O fim da entrada de dados é caracterizada por uma tupla $< 0, 0 >$.

Abaixo é apresentado um exemplo de entrada, referente a Figura 1:

```
5 5
2 1
3 2
4 3
4 2
5 4
0 0
```

Estarão disponíveis 5 casos de testes com as saídas esperadas, o restante deve ser gerado pelo aluno.

4 Saída

A saída esperada deve conter **um número inteiro e a ordem da classificação das corridas com os *Ids* dos pilotos, separados por vírgula em uma única linha**. O primeiro valor inteiro possui 3 categorias:

0 — não é possível definir a ordem da classificação através das memórias dos pilotos.

1 — é possível gerar uma ordem de classificação única.

2 — é possível gerar mais de uma ordem de classificação. Neste caso, de acordo com o historiador a ordem crescente do *id* deverá ser preservada.

A saída esperada contém os *Ids* dos pilotos ordenados de acordo com as memórias descobertas através da entrevista, separados por vírgula em uma única linha. No caso em que as memórias levam a mais de uma ordem de classificação (Veja um exemplo no anexo A), a ordem crescente do *Id* deverá ser preservada. Isto é, se não for possível determinar qual piloto ficou na frente, então considera-se que o piloto com menor *Id* fique na frente.

A saída deve ser feita em uma única linha, do seguinte modo:

// X[**espaço em branco**]ordem de classificação

X P_1,P_2,...,P_n

Os números da ordem de classificação devem estar separados somente por vírgulas. Não inclua espaços entre eles. No caso do exemplo apresentado na Figura 1, a saída seria:

```
1 1,2,3,4,5
```

Esta saída contém o inteiro **1** indicando que só existe uma solução possível para a classificação da corrida, **um espaço em branco** e em seguida a **ordem de classificação da corrida**.

Observação: Fique atento quanto a complexidade do seu algoritmo, para cada caso de teste é esperado que ele execute em no máximo 1 segundo. Além disso,

utilize o **diff** do Linux para conferir se o seu algoritmo está com a saída correta e de acordo com as respostas de exemplo.

5 Entrega

- A data de entrega desse trabalho é até às 23:55 do dia **09/10/2018** (09 de Outubro de 2018).
- A penalização por atraso obedece à formula apresentada na Equação 1, onde d são os dias úteis de atraso na entrega do projeto.

$$\frac{2^{d-1}}{0.32\%} \tag{1}$$

- Submeta apenas um arquivo chamado *<número 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!**
- A sua nota final será 0 se uma das partes (**documentação ou execução**) não for apresentada.

6 Documentação

A documentação não deve exceder o limite de **10 páginas**, sendo submetida no formato PDF e deve conter pelo menos os seguintes itens:

- Uma introdução do problema em questão com uma explicação clara e objetiva;
- Modelagem e solução proposta para o problema. O algoritmo deve ser explicado de forma clara. Se necessário, utilize pseudo-código e/ou esquemas ilustrativos;
- Análise de complexidade de tempo e espaço da solução usando o formalismo da notação assintótica visto em sala de aula;

- Análise experimental, variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significativamente a execução do algoritmo;
  - **Cabe a você gerar as entradas para esses experimentos**.
- Especificação da máquina utilizada nos experimentos realizados;
- Uma breve conclusão do trabalho implementado.

Um tutorial para elaboração da documentação e também um exemplo são apresentados no *minha.ufmg*[1]. Para visualizar o tutorial para elaboração da documentação (**clique aqui**), já para visualizar o exemplo de uma documentação pronta (**clique aqui**).

7 Código

O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux[2] dos laboratórios de graduação.

- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido. O utilitário deverá gerar um executável com o nome **tp1** que deverá **obrigatoriamente** ser capaz de receber o nome do arquivo de entrada e do arquivo de saída. O comando para executá-lo deverá seguir o exemplo abaixo:
  /.tp1 input.txt output.txt
- 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.
- **Legibilidade e boas práticas** de programação serão avaliadas
- **Não é permitido o uso de bibliotecas de terceiros**[3]
- **Não é permitido o compartilhamento de código entre os estudantes**. Indícios de plágio serão investigados e, caso confirmados, serão severamente punidos.

Bom trabalho!

[1] `https://sistemas.ufmg.br/idp/login.jsp`
[2] `http://www.crc.dcc.ufmg.br/infraestrutura/laboratorios/labs_unix`
[3] Dúvidas quanto a utilização de uma biblioteca em específico deverá ser sanada com o monitor responsável.

A Exemplos

Cenário 2

Considere o seguinte cenário para a segunda corrida. Seja $N = 5$, $M = 4$ e as seguintes memórias dos pilotos:

- $Piloto_2$ afirmou que o $Piloto_1$ ficou na sua frente
- $Piloto_4$ afirmou que o $Piloto_3$ e $Piloto_2$ ficaram na sua frente
- $Piloto_5$ afirmou que o $Piloto_4$ ficou na sua frente

Veja a Figura 2.

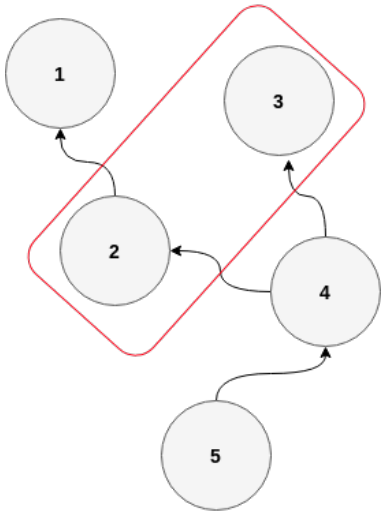

Figura 2: Segunda corrida do evento "auto-cone".

Os arquivos de entrada e a saída podem ser vistos na Tabela A.

input
5 4 2 1 4 3 4 2 5 4 0 0
output
2 1,2,3,4,5

Neste exemplo, de acordo com as regras apresentadas na Seção 4, não é possível encontrar uma **única** classificação.O $piloto_1$ foi o campeão, em seguida o $piloto_2$ ficou em segundo, o $piloto_3$ em terceiro, o $piloto_4$ em quarto e o $piloto_5$ em último.

Cenário 3

Considere outro cenário para a segunda corrida. Seja $N = 5$, $M = 6$ e as seguintes memórias dos pilotos:

- $Piloto_2$ afirmou que o $Piloto_1$ e $Piloto_5$ ficaram na sua frente
- $Piloto_4$ afirmou que o $Piloto_3$ e $Piloto_2$ ficaram na sua frente
- $Piloto_5$ afirmou que o $Piloto_4$ ficou na sua frente
- $Piloto_6$ afirmou que o $Piloto_5$ ficou na sua frente

Veja a Figura 3.

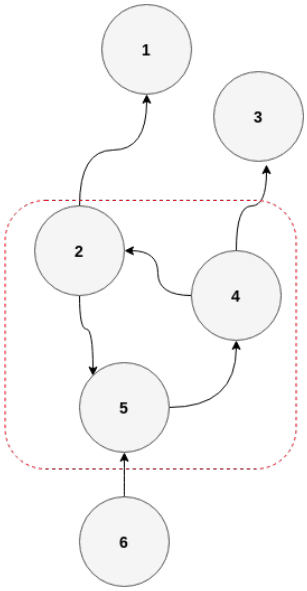

Figura 3: Terceira corrida do evento "auto-cone".

Os arquivos de entrada e a saída para este exemplo podem ser vistos na Tabela A.

input
6 6 2 1 2 5 4 3 4 2 5 4 6 5 0 0
output
0 -1

Neste exemplo, de acordo com as regras apresentadas na Seção 4, não existe uma classificação correta.

Referências

[1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[2] Steven Halim and Felix Halim. *Competitive Programming 3*. Lulu Independent Publish, 2013.

[3] Nivio Ziviani. Projeto de algoritmos com implementação em pascal e c. 2ª. *São Paulo: Thomson*, 2004.