Trabalho Prático 1

Algoritmos I

Entrega: 24/09/2019

1 Introdução

Edward Oakley Thorp (nascido em 14 de agosto de 1932) é um professor de matemática americano, autor, gerente de fundos de hedge e pesquisador de blackjack. Ele foi pioneiro nas aplicações modernas da teoria das probabilidades, incluindo o aproveitamento de correlações muito pequenas para obter ganhos financeiros confiáveis.

Thorp é o autor de Beat the Dealer, que provou matematicamente que a vantagem da casa no blackjack poderia ser superada pela contagem de cartas.

No blackjack, ou 21, as cartas altas favorecem o jogador, as cartas baixas, o cassino. Assim, um contador de cartas mantém um registro constante em sua mente, adicionando 1 para cartas baixas e subtraindo 1 para cartas altas. Quando o registro deles aumenta (o que significa mais cartas altas do que as cartas baixas no baralho), eles sabem que é hora de começar a fazer apostas mais altas.

Os contadores de cartões não ganham sempre - eles geralmente perdem muito dinheiro - mas estatisticamente e com o tempo, as chances estão a seu favor.

Isso deve ser feito secretamente porque, embora não seja ilegal, os cassinos não gostam e têm o direito de se recusar a deixar alguém jogar.

Figura 1: Exemplo de Sistema de contagem de cartas no Blacjack

1.1 Equipe MIT Blackjack

A equipe do MIT Blackjack era um grupo de estudantes e ex-alunos do Instituto de Tecnologia de Massachusetts, Harvard *Business School*, Universidade de Harvard e outras faculdades líderes que usavam técnicas de contagem de cartas e estratégias mais sofisticadas para vencer cassinos no blackjack em todo o mundo.

A equipe e seus sucessores operaram com sucesso de 1979 até o início do século XXI. Muitas outras equipes de blackjack foram formadas em todo o mundo com o objetivo de vencer os casinos.

Ao contrário da crença popular, não existem leis federais ou estaduais que restrinjam os cartões de contagem como estratégia para vencer. Nevada não tem leis em vigor que os cidadãos estaduais não podem contar cartas para ganhar blackjack, pôquer ou outros jogos nos cassinos. No entanto, os cassinos de Las Vegas são de propriedade privada e, portanto, têm o direito de expulsar alguém.

Os cassinos geralmente proíbem os marcadores de cartas de jogar um determinado jogo, como o blackjack. Usando imagens de segurança de alta tecnologia, os cassinos podem acompanhar todos os jogadores em todas as mesas, observando sinais de contagem de cartas. Se o cassino acredita que você está contando cartas - ou se você é muito bom no jogo - ele tem todo o direito de pedir que você saia da mesa, não jogue esse jogo ou até mesmo que saia do cassino.

Os proprietários de cassinos podem pedir para você sair de um cassino se você for muito barulhento, beligerante ou se eles pegarem cartas de contagem. Se você for solicitado a deixar um cassino, precisará cumprir. Caso contrário, a polícia pode acusá-lo de invasão. O cassino o proibirá de retornar e você poderá enfrentar acusações criminais. As leis de invasão de Las Vegas são complexas e podem facilmente levar a uma prisão se você não conhecer os direitos dos proprietários de cassinos.

Figura 2: Mike Aponte - Ex-membro da Equipe de Blackjack do MIT

1.2 Grupo de Blackjack de alunos da UFMG

Um grupo composto por alunos de Matemática, Engenharia e de cursos de Computação da UFMG se organizaram para realizar um estudo aprofundado do livro Beat the Dealer de Edward Oakley Thorp e de técnicas do jogo, com o objetivo de realizar uma viagem para Las Vegas para ganhar muito dinheiro nos casinos contando cartas sem serem pegos em flagrante.

Para isso, foi realizado um estudo intensivo dos desafios enfrentados pela tradicional equipe de BlackJack do MIT de modo a permitir a criação de estratégias que não deixasse o grupo de BlackJack da UFMG serem pegos.

Após a análise de todos os desafios enfrentados pela equipe de BlackJack do MIT, um grupo de alunos da Matemática propuseram que os alunos se organizassem de forma hierárquica de modo que as responsabilidades do time fosse subdividida. Ou seja, um membro ficaria responsável por comandar determinados membros e poderia ser comandado por outros membros. Dessa forma as sub-equipes foram modelados para haver um representante por cada equipe e as mesmas.

Para facilitar a organização do grupo e evitar suspeitas, os alunos de Engenharia sugeriram que a cada certo período de tempo fosse remodelada a estrutura hierárquica da equipe. Assim, é realizada uma rotatividade de responsabilidades entre os membros da equipe.

Os alunos também fazem reuniões para discutir o andamento da pesquisa. Nas reuniões todos os alunos falam sobre a sua experiência até o momento. Porém, a ordem de fala deve seguir a hierarquia, ou seja, as pessoas com hierarquias maiores devem falar antes das outras. Assim, se uma pessoa A comanda a pessoa B. Na reunião a pessoa A deve falar antes da pessoa B.

Agora entra a sua tarefa, como estudante de Computação foi designado a desenvolver um programa que realize as seguintes instruções:

- O comando SWAP (S) - verifica se existe uma aresta entre os alunos A e B. Caso a relação anterior exista, troca a direção de uma aresta que representava que o aluno A comandava o aluno B para uma que representa que o aluno B comanda o aluno A. Essa troca só pode ser realizada se o resultado não ocasionar um ciclo. Caso um ciclo ocorra, o grafo permanece sem alteração;
- O comando COMMANDER (C) - recebe como entrada um aluno A e responde a idade da pessoa mais jovem que comanda A (direta ou indireta);
- o comando MEETING (M) - identifica uma possível ordem de fala dos alunos em uma reunião.

2 O que fazer?

O objetivo deste trabalho é desenvolver um programa que realize as operações de SWAP, COMMANDER e MEETING. Para modelar este problema, você deve utilizar um grafo $G(V, A)$, no qual V são os membros da equipe e existe uma aresta (u, v) se e somente se o aluno u comanda o aluno v. Considerando todas as instruções necessárias o seu programa deve funcionar em tempo de $O(V + A)$. Além disso, você deve se atentar aos seguintes pontos:

- Não existem alunos com a mesma idade na equipe de alunos de blackjack da UFMG;
- Na entrada inicial não há circularidade nos comandos, ou seja, se o aluno A comanda o aluno B então B não pode mandar no aluno A, direta ou indiretamente. Lembre-se que ao realizar o comando SWAP essa propriedade deve ser mantida;
- A entrada e saída do programa deve obedecer os modelos que serão apresentados na próxima seção.

3 O arquivo de entrada

A entrada do problema será um arquivo de texto, no qual a primeira linha contém três inteiros N, M e I, indicando respectivamente o número de pessoas no time, o número de relações diretas entre os membros e o número de instruções.

Cada membro da equipe da UFMG é identificado por um número de 1 a N. A segunda linha contém N inteiros $K_i (1 \leq K_i \leq 100$, para $1 \leq i \leq N)$, onde K_i indica a idade do membro da equipe de número i.

Cada uma das M linhas seguintes contém dois inteiros X e $Y(1 \leq X, Y \leq N, X \; != \; Y)$, indicando que X comanda Y diretamente. Seguem-se I linhas, cada uma descrevendo uma instrução. Uma instrução de SWAP de comando é descrita em uma linha contendo o identificador S seguido de dois inteiros A e $B(1 \leq A, B \leq N)$, indicando os dois membros que devem trocar seus lugares na cadeia de comando. Uma instrução COMMANDER é descrita em uma linha contendo o identificador C seguido de um inteiro $E(1 \leq E \leq N)$, indicando um membro. Uma instrução MEETING é descrita em uma linha contendo apenas o identificador M.

Para o exemplo de entrada a seguir considere o grafo da Figura 3, na qual o primeiro número corresponde ao identificador da pessoa e o segundo a idade.

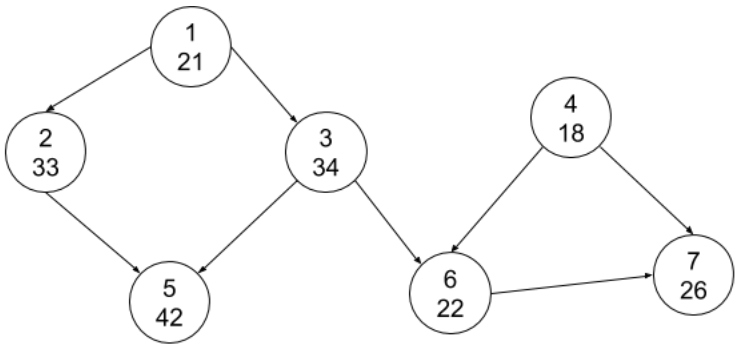

Figura 3: Grafo do exemplo de entrada

Exemplo de entrada

```
7 8 8
21 33 34 18 42 22 26
1 2
1 3
2 5
3 5
3 6
4 6
4 7
6 7
C 7
S 4 2
C 3
S 3 6
C 3
S 4 7
C 4
M
```

4 O arquivo de saída

Para cada instrução SWAP seu programa deve imprimir S T caso a troca tenha sido um sucesso. Caso contrário, se a troca não foi possível seu programa deve imprimir S N. Lembre-se que a troca pode não ocorrer devido a não existir uma aresta entre os alunos A e B ou devido a criação de um ciclo no grafo. Para cada instrução COMMANDER seu programa deve imprimir uma linha contendo o caractere C e um único inteiro que corresponde a idade da pessoa mais jovem que comanda (direta ou indiretamente) o membro nomeado na pergunta. Se o aluno não é comandado por ninguém, imprima o caractere C $*$ (asterisco). Para a instrução MEETING, o seu programa deve imprimir uma das alternativas possíveis de ordem de fala de todos os grupos, ou seja, a saída será o caractere R e os N membros do grupo considerando a ordem de fala.

O exemplo a seguir corresponde a saída do exemplo da entrada. Lembre-se que a instrução MEETING pode ter **várias respostas válidas** diferentes, você pode fornecer qualquer uma delas.

Exemplo de saída

```
C 18
S N
C 21
S T
C 18
S N
C *
M 4 6 7 1 3 2 5
```

5 Avaliação Experimental

Para a avaliação experimental o aluno deverá criar entradas para o problema de forma a avaliar se o tempo de execução do algoritmo está de acordo com a complexidade reportada na documentação, aumentando o número de vértices da entrada, guardando os tempos de execução e colocando esses valores em gráficos para melhor visualização. É **necessário** executar mais de uma vez o algoritmo para entradas com o mesmo tamanho e número de trocas (no mínimo 10 vezes), guardando o tempo de execução e reportando a **média** e o **desvio padrão** deste tempo. Faça isso para times de vários tamanhos (exemplo: 4, 8, 12, 16, 20, ... etc). Isso é necessário para observar como o tempo está se comportando dado diferentes execuções e diferentes configurações do grafo.

Além disso, responda as seguintes perguntas no relatório:

- Por que o grafo tem que ser dirigido?
- O grafo pode ter ciclos?
- O grafo pode ser uma árvore? O grafo necessariamente é uma árvore?

6 O que deve ser entregue

Você deve submeter um arquivo compacto (zip ou tar.gz) no formato **seu_nome_sua_matrícula** via Moodle contendo:

- todos os arquivos do código *.c*, *.cpp* e *.h* que foram implementados,
- um arquivo *makefile*[1] **que crie um executável tp1**,
  - **ATENÇÃO:** O makefile é para garantir que o código está sendo compilado corretamente, de acordo com o modo que vocês modelaram o programa. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp1**.
- sua documentação.

Sua documentação deve ter até 10 páginas contendo:

- uma breve descrição do problema,
- explicações das estruturas de dados e dos algoritmos utilizados para resolver o problema. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. Não inclua textos de códigos em sua documentação.
- análise de complexidade da solução proposta (espaço e tempo). Cada complexidade apresentada deverá ser devidamente justificada para que seja aceita.
- avaliação experimental.

O seu TP deverá ser entregue de acordo com a data especificada no moodle. A penalidade em porcentagem para os TPs atrasados é dada pela fórmula $2^{d-1}/0.16$.

7 Implementação

7.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado na linguagem **C** ou **c++** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos. Além disso, certifique-se que seu código compile e execute corretamente nas máquinas **Linux** dos laboratórios do **DCC**.

ATENÇÃO: O arquivo da entrada deve ser passado como parâmetros para o programa através da linha de comando (e.g., *$ ./main equipe.txt*) e imprimir a saída no **stdout** (com printf), não em um arquivo.

ATENÇÃO: certifique-se que o programa execute com os três arquivos passados como exemplos junto com a documentação no Moodle (dataset.zip), sem alterar os arquivos passados. Isso vai garantir que seu programa vai conseguir ler corretamente os arquivos do corretor automático.

7.2 Testes

A sua implementação passará por um processo de correção automática, o formato da saída de seu programa deverá ser idêntico aquele descrito nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. Para auxiliar na depuração do seu trabalho, será fornecido um pequeno conjunto de entradas e suas saídas. É seu dever certificar que seu programa atenda corretamente para qualquer entrada válida.

[1] `https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles`

7.3 Qualidade do código

É importante prestar atenção para a qualidade do código, mantendo-o organizado e comentado para não surgir dúvidas na hora da correção. Qualquer decisão que não estiver clara dada a documentação e a organização do código será descontada na nota final.

8 Consideração Final

Alunos são encorajados a discutir problemas e soluções com os colegas, entretanto, assim como em todos os trabalhos dessa disciplina, é estritamente proibido a cópia parcial ou integra de códigos, seja da internet ou de colegas. Seja honesto! Você não aprende nada copiando código de terceiros. Se a cópia for detectada, sua nota será zerada e o professor será informado para que as devidas providências sejam tomadas.