Trabalho Prático 1: Mortal Kontest

Algoritmos e Estruturas de Dados III – 2017/2
Entrega: 04/10/2017

1 Introdução

Nubby está de volta, como sabemos, ele é um adolescente que possui vários hobbies, sendo um deles jogar videogame. Ele e seus amigos costumam se reunir há anos para competir no jogo *Mortal Kontest* e já jogaram várias partidas entre si ao longo de vários anos. No entanto, sabemos que Nubby tem gostos incomuns e um deles é registrar os resultados de cada campeonato. Assim ele possuí dados de todos as partidas que ele e os amigos jogaram até hoje.

Nubby decidiu organizar um campeonato de *Mortal Konstest* em sua casa, com N amigos. Como organizar o campeonato daria muito trabalho, Nubby decidiu não participar, somente organizar as partidas entre os seus amigos, pois não possuiria tempo para praticar (dizem que Nubby nunca ganhou um campeonato). Para facilitar o sorteio das partidas, Nubby usou programação, como vocês já sabem é um de seus hobbies, e estruturou o campeonato com as seguintes regras:

- O campeonato terá $n - 1$ rodadas.
- Os N participantes da competição estarão em uma lista $V = \{v_1, v_2, ..., v_n\}$.
- Cada rodada possui uma única partida. Para essa partida, dois amigos v_i e v_j, $i \neq j$ serão sorteados sendo que a probabilidade de qualquer par de amigos ser sorteado é a mesma.
- Os dois amigos sorteados jogam uma partida em que o vencedor se classifica para a rodada seguinte e o perdedor é eliminado definitivamente do campeonato e removido da lista de participantes V.
- Na rodada de número $n - 1$ sobram apenas dois amigos que realizam a final do campeonato, e o vencedor da partida é o grande campeão de *Mortal Konstest*.

Após definir como seria a estrutura do campeonato, Nubby convidou todos os seus amigos. Bryan, seu melhor amigo, confirmou sua presença, e com grande confiança falou que com certeza venceria o campeonato, pois se dizia o melhor e com mais chance de vencer entre os amigos. Nubby, intrigado com a afirmação do amigo, decidiu comprová-la.

Usando os registros que está coletando há anos, Nubby calculou a probabilidade de cada amigo vencer o outro e montou uma matriz P de tamanho $N \times N$. Assim, se um amigo v_i é sorteado para jogar contra o amigo v_j, a probabilidade de v_i vencer está armazenada na célula P_{ij} da matriz. De forma similar, a probabilidade de v_j vencer está armazenada na célula P_{ji} da matriz, sendo que $P_{ji} = 1 - P_{ij}$. Nubby preencheu todos os elementos da diagonal principal da matriz com o valor 0, que é a probabilidade de um amigo vencer a si mesmo.

Após montar essa matriz, Nubby utilizou programação para calcular a probabilidade de cada um dos seus N amigos ganhar o campeonato, seguindo as regras que ele havia imposto, com o objetivo de verificar se seu amigo Bryan realmente tinha maior probabilidade de vencer o campeonato. No entanto, a complexidade do seu algoritmo ficou muito alta e seu programa muito lento, e após horas executando sem resultados, Nubby desistiu desse algoritmo, ficando sem a resposta.

Nubby já estava desanimado, porém ele lembrou do desafio de Bryan em que foi convencido que utilização de algumas técnicas podem melhorar o desempenho dos algoritmos. Em seguida, Nubby pesquisou e descobriu que o paradigma de programação dinâmica poderia ajudá-lo a aumentar a eficiência do seu algoritmo. Com o pouco tempo até seus amigos chegarem para o campeonato, Nubby conta com você para desenvolver um algoritmo, utilizando programação dinâmica, que calcule a probabilidade de cada um dos seus amigos ganhar o campeonato.

2 Entrada e saída

Entrada A primeira linha contém um inteiro positivo $\mathbf{N}(1 \leq N \leq 25)$, indicando o número de amigos de Nubby que participarão do campeonato. Cada uma das N linhas seguintes possuí N números reais. Esses $N \times N$ números representam a matriz de probabilidade $P^{N \times N}(0 \leq p_{ij} \leq 1)$, que Nubby acredita ser a probabilidade de seu amigo i vencer seu amigo j . É garantido que na diagonal principal todos os elementos serão iguais a zero e para todas as demais células $A_{ji} = 1 - A_{ij}$.

Exemplo de entrada

```
2
0.0000  0.3000
0.7000  0.0000
```

Saída Uma sequência $P = [p_1, p_2, ..., p_n]$ de N números reais separados por espaço que representam a probabilidade de cada amigo de Nubby ganhar o campeonato. Cada número $p_i \in P$ deve ter 4 casas decimais e p_i representa a probabilidade do amigo v_i ganhar o campeonato. Como os resultados são probabilidades, cada $p_i \in P$ estará no intervalo $[0, 1]$ e $\sum p_i = 1$.

Exemplo de saída

```
0.3000  0.7000
```

3 O que deve ser entregue

Deverá ser submetido um arquivo **.zip** contendo apenas uma pasta chamada tp1, esta pasta deverá conter: (i) Documentação **em formato PDF** e (ii) Implementação.

Documentação Deverá ter no máximo 10 páginas e seguir tanto os critérios de avaliação discutidos na Seção 4.1, bem como as diretrizes sobre a elaboração de documentações disponibilizadas no *moodle*. Além disso, a documentação deverá conter análise experimental validando as complexidades de tempo e espaço e equação de recorrência utilizada para memorizar as soluções.

Implementação Código fonte do seu TP (*.c* e *.h*), a implementação deve estar separadas em arquivos ċ e ḣ, **por exemplo** crie um arquivo *kontest.*{*c/h*}.

Makefile Inclua um *makefile* na submissão que permita compilar o trabalho. **Deverá ser produzido um executável** chamado *kontest*.

É obrigatório o uso das *flags*: **-Wall -Wextra -Werror -std=c99 -pedantic -O2**. **Pode-se assumir que bibliotecas padrões da linguagem serão automaticamente lincadas pelo compilador**.

Execução A entrada e saída devem ser lida e impressa utilizando a entrada padrão (stdin). Um exemplo de execução utilizando o executável *kontest* para um arquivo de entrada *in_1.in* e imprimindo o resultado em *out_1.out* será da seguinte forma:

$$./kontest < in\_1.in > out\_1.out$$

4 Avaliação

Eis uma lista **não exaustiva** dos critérios de avaliação que serão utilizados.

4.1 Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho.

Solução do Problema Você deve descrever cada solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. 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** é necessário incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que se torna interessante. **Inclua a equação de recorrência do problema**.

Análise de Complexidade Inclua uma análise de complexidade de tempo e espaço dos principais algoritmos e estrutura de dados utilizados. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita. Faça uma análise para cada tipo de algoritmo implementado. Mostre vantagens e desvantagens de cada um dos modelos utilizados.

Avaliação Experimental Sua documentação deve incluir os resultados de experimentos que avaliem o tempo de execução de seu código em função de características da entrada. Cabe a você gerar entradas para esses experimentos. Para tal, um gráfico mostrando o tempo de execução em função do tamanho da entrada pode ser interessante. Você também deve interpretar os resultados obtidos. Comente sobre cada gráfico ou tabela que você apresentar mostrando o que é possível concluir a partir dele.

4.2 Implementação

Linguagem & Ambiente O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação ou utilizem outras bibliotecas que não seja padrão ou variáveis globais serão zerados. Além disso, é recomendado que se certifique que seu código compile e funcione corretamente nas máquinas **Linux** dos laboratórios do DCC.

Casos de teste A sua implementação passará por um processo de correção automatizado, portanto, o formato da saída do seu programa deve ser idêntico aquele descrito nessa especificação. Saídas com qualquer divergência serão consideradas erradas, mesmo que as divergências sejam *whitespaces*. e.g. espaços, *tabs*, quebras de linha, etc. Para auxilia-lo na depuração do seu código, será fornecido um pequeno, **não-exaustivo**, conjunto de entradas e suas respectivas saídas. É seu dever certificar-se que seu código funciona corretamente para qualquer entrada válida.

Alocação Dinâmica Algoritmos e estruturas de dados deverão fazer uso de memória alocada dinamicamente (`malloc()` ou `calloc()`). Certifique-se que seu programa utiliza essas regiões de memória corretamente, pois os monitores penalizarão implementações que realizam *out-of-bounds access* e que tenham vazamento de memória (não desalocar memória dinâmica).
A alocação dinâmica deverá fazer uso das funções malloc() ou calloc() da biblioteca padrão C, bem como liberar tudo o que for alocado utilizando free(), para gerenciar o uso da memória. **DICA:** Utilize `valgrind` antes de submeter o seu TP.

Qualidade do código Seu código também será avaliado no quesito de legibilidade, dando atenção, porém não limitando-se, aos seguintes items: (i) **INDENTAÇÃO**; (ii) nomes de variável e função descritivos e claros; (iii) Modularização adequada; (iv) Comentários dentro de funções, explicando o que certos trechos mais complicados fazem; (v) Comentários fora de funções, explicando, em alto-nível, o que as funções mais importantes fazem; (vi) funções concisas que desempenham somente uma tarefa; (vii) **Proibido uso de variáveis globais**.

Atrasos Trabalhos poderão ser entregues após o prazo estabelecido, porém sujeitos a uma penalização regida pela seguinte fórmula:

$$\Delta_p = \frac{2^{d-1}}{0.32}\,\%$$

Por exemplo, se a nota dada pelo corretor for 70 e você entregou o TP com 4 dias corridos de atraso, sua penalização será de $\Delta_p = 25\%$ e, portanto, a sua nota final será: $N_f = 70 \cdot (1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

5 Considerações Finais

Assim como em todos os trabalhos dessa disciplina **é estritamente proibida a copia parcial ou integra de códigos, seja da internet ou de colegas**. Utilizaremos o algoritmo *MOSS* para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.