Trabalho Prático 1: This city is on fire!

Entrega: 18/10/2015

September 23, 2015

1 Introdução

Você foi contratado como bombeiro em uma cidade conhecida por seus altos número de incêndios. No primeiro dia de trabalho você recebe um mapa da cidade onde estão descritos os quarteirões e os trechos das ruas que os conectam. Além disso, no mapa estão as localizações de todos os corpos de bombeiros espalhados pela cidade. Para cada trecho, existe também a probabilidade $P(u, v)$ de acontecer um incêndio entre o quarteirão u, v. Considere que $P(u, v) = P(v, u)$ e todas as ruas são de mão dupla.

Porém, você é um bombeiro preguiçoso e gostaria de evitar ao máximo ter que apagar um incêndio. Com base nessas informações, você decide que ao sair de um corpo de bombeiro para outro, você:

1. Nunca ficará a mais de k quarteirões de algum corpo de bombeiro durante o seu trajeto, assim você pode chamar outros bombeiros para fazer seu trabalho.

2. Escolherá o caminho onde há a menor probabilidade de acontecer um incêndio.

Seu trabalho é: dadas as informações que descrevem o mapa da cidade (com seus corpos de bombeiros, ruas e probabilidades de incêndio), descubra o caminho que respeita as restrições descritas acima. Modele e mostre uma solução para o problema utilizando grafos.

2 Entrada

Seu programa deverá ler a entrada da entrada padrão ($stdin$) e gravar a saída na saída padrão ($stdout$). Na primeira linha da entrada estará um número inteiro N, $0 < N \leq 100$, que representa o número de instâncias do problema a serem simuladas. A primeira linha de cada instância contém o número de quarteirões Q ($0 < Q \leq 10^3$), o número de ruas R ($0 < R \leq \frac{Q(Q-1)}{2}$) que liga esses quarteirões, o id do ponto de saída S e chegada C, um número K que é a qual distância você pode estar de algum corpo de bombeiros e um inteiro D ($0 < D \leq Q$) que indica o número de corpos de bombeiros presente na cidade. A seguir, nas próximas R linhas estarão descritos os

trechos entre quarteirões com dois inteiros - que representam os ids dos quarteirões, um número real positivo com precisão de duas casas decimais no intervalo [0,1] que representa a probabilidade de incêndio no trecho. Nas próximas D linhas é mostrado quais quarteirões possuem um corpo de bombeiro. Veja o exemplo a seguir:

```
1
9 12 0 8 2 1
0 1 0.30
0 3 0.20
1 2 0.30
1 4 0.40
2 5 0.10
3 4 0.20
3 6 0.40
4 7 0.10
4 5 0.15
5 8 0.10
6 7 0.30
7 8 0.10
6
```

3 Saída

A saída para cada teste de entrada deve conter a probabilidade de passar por um incêndio (com duas casas decimais) no caminho a ser percorrido desde o quarteirão inicial até o quarteirão final que respeita as restrições já enunciadas. A ordem de escolha da melhor rota deve respeitar a seguinte prioridade:

1. Menor probabilidade de acontecer um incêndio
2. Ordem lexicográfica

Veja o exemplo de saída para o problema mostrado anteriormente:

```
0.48 0 3 4 7 8
```

Caso não seja possível resolver o problema seguindo as restrições enunciadas, a saída deverá conter apenas o número -1.

4 Cálculo da probabilidade

Fique atento à forma de calcular a probabilidade de passar por um incêndio. Note a probabilidade do caminho $0-3-4$ NÃO é dado por $P(0,3)*P(3,4)$ - isso representa a probabilidade de passar por um incêndio EM AMBOS os trechos. No entanto, você quer saber a probabilidade de passar por um incêndio em qualquer um desses trechos.

Essa probabilidade pode ser calculada por $1-P_N$, onde P_N representa a probabilidade de NÃO passar por um incêndio em nenhum dos trechos.

Dado que $P_N(u,v) = 1-P(u,v)$, sabemos que a probabilidade de não passar por um incêndio em qualquer trecho do caminho (0,3,4) - que, por abuso de notação, chamaremos de $P_N(0,3,4)$ - é:

$P_N(0,3,4) = (P_N(0,3) * P_N(3,4))$
$P_N(0,3,4) = ((1-P(0,3)) * (1-P(3,4)))$
$P_N(0,3,4) = ((1-0.20) * (1-0.20)))$
$P_N(0,3,4) = ((0.80) * (0.80)))$
$P_N(0,3,4) = 0.64$

Portanto a probabilidade de passar por um incêndio durante o caminho $0-3-4$ é $1-0.64 = 0.36$.

5 O que entregar

Você deve submeter uma documentação de até 10 páginas contendo uma descrição da solução proposta, detalhes relevantes da implementação, além de uma análise de complexidade de tempo dos algoritmos envolvidos e de complexidade de espaço das estruturas de dados utilizadas. Siga as diretrizes sobre como fazer uma documentação que foram disponibilizadas no portal *minha.ufmg*.

Além da documentação, você deve submeter um arquivo compactado no formato.*tar.gz* contendo todos os arquivos de código (*.c* e *.h*) que foram implementados. Além dos arquivos de código, esse arquivo compactado deve incluir um *makefile*. Consulte os tutoriais disponibilizados no *minha.ufmg* para descobrir como fazer um. Finalmente, lembre-se de não incluir nenhuma pasta no arquivo compactado.

6 Avaliação

Seu trabalho será avaliado quanto a documentação escrita e à implementação. Eis uma lista **não exaustiva** de critérios de avaliação utilizados.

Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho e um resumo da sua solução.

Solução do Problema Você deve descrever a solução do problema de maneira clara e precisa. Para tal, artifícios como pseudocódigos, exemplos ou figuras podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. **Não** é preciso incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando os mesmos influenciem o seu algoritmo principal, o que se torna interessante.

Análise de Complexidade Inclua uma análise de complexidade de tempo dos principais algoritmos implementados e uma análise de complexidade de espaço das principais estruturas de dados de seu programa. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita.

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. Por exemplo: se esse trabalho fosse sobre ordenação, seria interessante mostrar como o tempo de execução de cada algoritmo varia quando o número de items a serem ordenados aumenta. 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.

Limite de Tamanho Sua documentação deve ter no máximo 10 páginas. Todo o texto a partir da página 11, se existir, será desconsiderado.

Guia Há um guia e exemplos de documentação disponíveis no moodle.

Implementação

Linguagem e Ambiente

- Implemente tudo na linguagem **C**. Você pode utilizar qualquer função da biblioteca padrão da linguagem em sua implementação, mas **não** deve utilizar outras bibliotecas. **Trabalhos em outras linguagens de programação serão zerados. Trabalhos que utilizem outras bibliotecas também.**
- Os testes serão executados em **Linux**. Portanto, garanta que seu código compila e roda corretamente nesse sistema operacional. A melhor forma de garantir que seu trabalho rode em Linux é escrever e testar o código nele. Há dezenas de máquinas com Linux nos laboratórios do DCC que podem ser utilizadas. Você também pode fazer o download de uma variante de Linux como o **Ubuntu** (`http://www.ubuntu.com`) e instalá-lo em seu computador ou diretamente ou por meio de uma máquina virtual como o **VirtualBox** (`https://www.virtualbox.org`). Há vários tutoriais sobre como instalar Linux disponíveis na web.

Casos de teste Seu trabalho será executado em um conjunto de entradas de teste.

- Essas entradas *não* serão disponibilizadas para os alunos até que a correção tenha terminado. Faz parte do processo de implementação testar seu próprio código.

- Você perderá pontos se o seu trabalho produzir saídas incorretas para algumas das entradas ou não terminar de executar dentro de um tempo limite preestabelecido. Esse tempo limite é escolhido com alguma folga. Garanta que seu código roda a entrada de pior caso em no máximo alguns minutos e você não terá problemas.
- A correção será **automatizada**. Esteja atento ao formato de saída descrito nessa especificação e o siga precisamente. Por exemplo: se a saída esperada para uma certa entrada o número 10 seguido de uma quebra de linha, você deve imprimir apenas isso. Imprimir algo como "A resposta e: 10" contará como uma saída **errada**.
- Os exemplos mostrados nessa especificação são parte dos casos de teste.
- Os testes disponíveis no MOODLE são apenas preliminares. Os monitores executarão os trabalhos nos demais casos de teste.
- **Você deve entregar algum código e esse código deve compilar e executar corretamente para, pelo menos, um dos testes disponibilizados no MOODLE. Se isso não ocorrer, a nota do trabalho prático será zerada.**

Alocação Dinâmica Você 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.

Makefile Inclua um makefile na submissão que permita compilar o trabalho.

Qualidade do Código Seu código deve ser bem escrito:

- Dê nomes a variáveis, funções e estruturas que façam sentido.
- Divida a implementação em módulos que tenham um significado bem definido.
- Acrescente comentários sempre que julgar apropriado. Não é necessário parafrasear o código, mas é interessante acrescentar descrições de alto nível que ajudem outras pessoas a entender como sua implementação funciona.
- Evite utilizar variáveis globais.
- Mantenha as funções concisas. Seres humanos não são muito bons em manter uma grande quantidade de informações na memória ao mesmo tempo. Funções muito grandes, portanto, são mais difíceis de entender.
- Lembre-se de indentar o código. Escolha uma forma de indentar (tabs ou espaços) e mantenha-se fiel a ela. Misturar duas formas de indentação pode fazer com que o código fique ilegível quando você abri-lo em um editor de texto diferente do que foi utilizado originalmente.
- Evite linhas de código muito longas. Nem todo mundo tem um monitor tão grande quanto o seu. Uma convenção comum adotada em vários projetos é não passar de 80 caracteres de largura.
- **Tenha bom senso**.

7 Considerações Finais

- Essa especificação não é isenta de erros e ambiguidades. Portanto, se tiverem problemas para entender o que está escrito aqui, pergunte aos monitores.
- A penalização por atraso seguirá será de $\frac{2^{d-1}}{0.32}\%$, em que d é o número de dias **úteis** de atraso.
- Você **não** precisa de utilizar uma IDE como Netbeans, Eclipse, Code::Blocks ou QtCreator para implementar esse trabalho prático. No entanto, se o fizer, **não** inclua os arquivos que foram gerados por essa IDE em sua submissão.
- Esteja atento ao tamanho da entrada e às complexidades de seus algoritmos.
- **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 tomem as devidas providências.