Trabalho Prático 2: Reducing the costs!

Entrega: 16/10/2017

Introdução

No Brasil, diversos produtos têm sofrido aumento nos preços, e os combustíveis não ficaram fora disso. Devido a este aumento, você foi contratado por uma empresa de transporte para reduzir seus custos. Como a empresa não quer mandar nenhum funcionário embora, para manter a produtividade e o lucro, ela quer reduzir a quantidade de combustível utilizada por seus caminhões. Desta forma, a sua função é reduzir a quilometragem rodada pelos caminhões na busca e entrega de produtos.

Todo caminhão da empresa faz sempre o seguinte percurso: sai de uma determinada *cidade x*, percorre todas as cidades atendidas pela empresa e retorna à *cidade x* novamente. Em algumas cidades esses caminhões devem pegar os produtos e em outras eles devem entregar os produtos. Desta forma, se um produto que está na *cidade x* deve ser entregue na *cidade y*, o caminhão da empresa deve passar na *cidade x* antes de passar na *cidade y*, pois assim ele não precisa ir em uma mesma cidade mais de uma vez.

Assim, sabendo onde os produtos estão e onde eles deverão ser entregues, você deve determinar a ordem em que as cidades devem ser visitadas para que os caminhões da empresa percorram o menor caminho possível. É importante lembrar que, mesmo que um caminho seja menor, se você visita a *cidade x* antes da *cidade y* e esta última tem produtos para serem entregues na *cidade x*, então esta não é uma ordem válida.

Código

Você deve implementar, em linguagem C, um programa que seja capaz de:

- Dado um número *N* de cidades, suas respectivas coordenadas e um conjunto de restrições de precedência, encontrar qual é a menor distância a ser precorrida saindo de uma cidade de origem, percorrendo todas as cidades que devem ser atendidas pela empresa de transporte e retornando a cidade de origem novamente.

- Você deve respeitar as restrições de precedência, ou seja, se na entrada indica que a *cidade x* contém produtos que devem ser entregues na *cidade y*, então a *cidade x* deve ser visitada antes da *cidade y*.

- Cada uma das cidades só podem ser visitadas apenas uma vez, exceto a cidade de origem, que é visitada uma segunda vez quando o caminhão retorna à cidade de origem.
- Caso não seja possível chegar a uma solução por causa das restrições de precedência, deve-se imprimir, ao invés da menor distância, a palavra ”Deadlock”.
- Uma abordagem de grafos deve ser utilizada para resolver o problema.

Entrada e Saída

Seu programa deverá ler a entrada da entrada padrão (`stdin`) e gravar a saída na saída padrão (`stdout`). A entrada de teste é composta por vários casos de teste. Em cada caso de teste, a primeira linha contém um inteiro (*N*) ($1 \leq N \leq 22$) que corresponde ao número de cidades atendidas pela empresa de transporte. Cada uma das *N* linhas seguintes contém dois inteiros separados por espaço, que correspondem, respectivamente, aos valores *x* e *y* ($0 \leq x, y \leq 10^9$) da coordenada daquela cidade. A primeira coordenada da entrada é referente a cidade que tem o identificador 0, a segunda coordenada é referente a cidade com identificador 1, a n-ésima coordenada é referente a cidade com identificador n-1. O ponto de partida do caminhão é a cidade com identificador 0.

Depois de ler as *N* coordenadas das cidades, a entrada da instância contém um inteiro (*R*) ($0 \leq R \leq 462$) que representa o número de restrições de precedência que devem ser consideradas na resolução do problema. Cada uma das *R* linhas seguintes contém dois inteiros, *a* e *b*, que determinam que a *cidade a* deve ser visitada antes da *cidade b*, pois a *cidade a* contém produtos que devem ser entregues na *cidade b*.

Para cada instância do problema, deve ser impresso na saída apenas uma linha com a menor distância necessária para sair da cidade de origem, passar por todas as outras cidades atendidas pela empresa de transporte apenas uma única vez e retornar novamente à cidade de origem, respeitando todas as restrições de precedência listadas na entrada do programa. Esse valor deve ser impresso com precisão de 2 casas decimais.

A distância é calculada de acordo com a fórmula:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{1}$$

Um exemplo de entrada e saída para o programa está na Tabela 1.

Será avaliada a capacidade de vocês “podarem” parte do espaço de solução para certas situações. Entretanto, uma “poda” pode não ocorrer sempre, ou seja, ela pode depender de algumas situações para ser aplicada. Para o trabalho, você deve ser capaz de procurar situações as quais você pode efetuar ”podas”.

O que entregar

Você deve submeter uma documentação de até 12 páginas contendo uma descrição de como cada uma das operações foi implementada, além de uma análise de complexidade de tempo dos algoritmos envolvidos.

ENTRADA	SAÍDA
4 0 0 0 8 4 8 4 0 4 2 1 2 3 3 1 1 3 6 1 1 2 1 2 2 3 2 2 4 1 4 2 2 1 5 4 20 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 19 20 20 19 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 18 18 19 12 1800 1000 900 200 1300 1500 2200 600 1300 1700 200 700 600 1100 1100 400 100 1500 1300 1400 2500 800 2500 140 0	Deadlock 9.24 53.74 7462.06

Table 1: Toy example: teste o qual o TP deve gerar a saída correta.

Além disso, você deve cobrir os seguintes itens em sua documentação:

- **A modelagem do problema como um problema de decisão.**
- **Discutir porque esse problema de decisão é NP-completo fazendo uma correlação com outro problema NP-Completo similar.**
- **Mostrar a existência de uma instância fácil para o problema de decisão.**
- **Descrição das "podas" realizadas com sua eficácia, ou seja, verificar se ela se aplica a qualquer situação. Caso não se aplique, mostrar a configuração ideal da "poda" (a qual ela tem o máximo de sua eficácia) e o pior caso (quando ela não é aplicada).**

Além da documentação, você deve submeter um arquivo compactado contendo todos os arquivos de código (*.c* e *.h*) que foram implementados. Além dos arquivos de código, esse arquivo compactado deve incluir um *makefile*.

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, artificios como pseudo-códigos, 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.

Valor

- O trabalho vale 10 pontos, sendo 8 pontos para o código e 2 pontos para a documentação. Ambos devem ser entregues. O aluno que entregar somente um dos dois terá sua nota zerada.
- O trabalho que não compilar receberá nota 0.
- O trabalho poderá ser feito em dupla.

Considerações Finais

- Essa especificação não é isenta de erros e ambiguidades. Portanto, se tiver problemas para entender o que está escrito aqui, me pergunte e pergunte ao monitor.