UFMG/ICEx/DCC — Algoritmos e Estrutura de Dados 3

Trabalho Prático 2

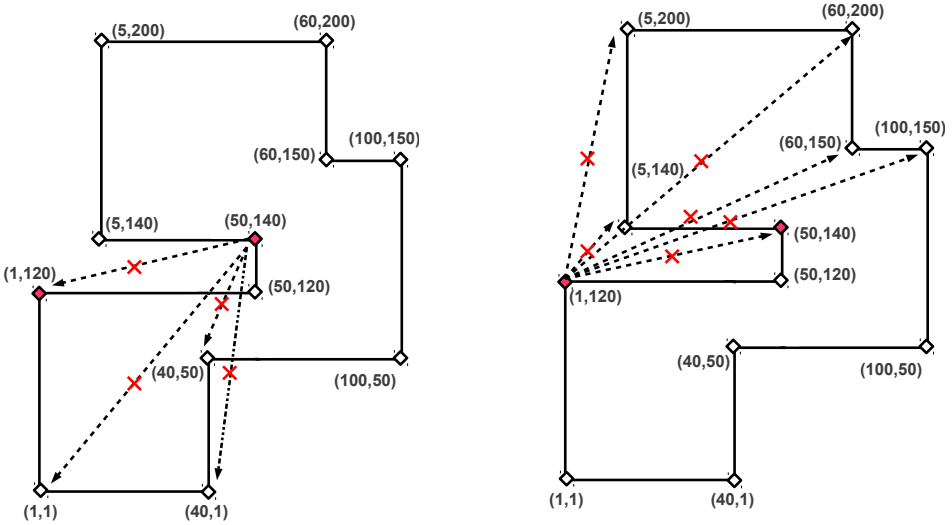
Bacharelado — 1º Semestre de 2012

Este trabalho prático tem por objetivo modelar um problema NP-Completo que envolve estruturas de grafos, elaborar diferentes heurísticas para solucionar um mesmo problema e, por fim, apresentar uma análise comparativa entre essas abordagens.

1 Definição do Problema

Este trabalho aborda o problema do Banco Central que consiste em encontrar o menor número de guardas necessários para vigiar um banco. Esse é um problema de visibilidade da geometria computacional. O layout do banco é representado por um polígono simples de arestas não colineares, isto é, as arestas não adjacentes do polígono não se interceptam e nenhuma aresta está alinhada a outra. Essas restrições impostas ao layout do banco simplificam o problema e evitam que o aluno tenha que tratar casos mais complexos de interseção de retas. Nesse layout (polígono), um possível guarda do banco é representado por um vértice. Dessa forma, um conjunto S de guardas (vértices) vigia um banco (polígono) se, e somente se, para cada vértice v do polígono, existe algum vértice $u \in S$ tal que um segmento de reta entre v e u **NÃO** abandona o polígono – isto é, esse segmento de reta está **totalmente** contido dentro do polígono, ou coincide com algum vértice/aresta desse polígono. As funções para interseção de segmentos de reta apresentadas no livro do Sedgewick podem ser utilizadas (ver Referências).

A Figura 1 apresenta o layout de uma instância do problema com as coordenadas indicadas para cada vértice que compõe o polígono. Para a instância apresentada, apenas dois guardas são suficientes para vigiar o banco (guardas representados por vértices em vermelho). Caso um guarda não seja capaz de vigiar um vértice, foi apresentado um segmento de reta do guarda a esse vértice que deixa o polígono. Apesar de existirem múltiplas soluções para uma dada instância, o número ótimo de vértices que vigia todo o polígono permanece o mesmo.



(a) O guarda (50,140) não vigia os vértices (1,1), (1,120), (40, 1) e (40,50).

(b) O guarda (1,120) não vigia os vértices (50,140), (5,140), (5,200), (60,200), (60,150) e (100,150).

Figura 1: Dado o layout acima, os guardas (1,120) e (50,140) vigiam todos os vértices do polígono, sendo o número mínimo (ótimo) de guardas necessários para vigiar esse banco igual a dois. Para cada figura, foram incluídos os segmentos de reta entre os guardas e os demais vértices que abandonam o polígono, o que indica os vértices não visíveis para cada guarda.

Neste trabalho, o aluno deve elaborar duas heurísticas distintas que solucionem o problema, e apresentar uma análise comparativa entre os resultados obtidos pelas abordagens para uma mesma instância.

PONTO EXTRA: apresentar e justificar o fator de aproximação das duas heurísticas em relação a solução ótima do problema para qualquer instância genérica. Dessa forma, pede-se que o aluno apresente o quanto sua solução proposta pode estar distante no máximo da solução ótima.

A implementação do algoritmo **DEVE** utilizar alocação dinâmica de memória.

1.1 Entrada e Saída

O arquivo executável deve ser chamado de *tp2* e deve receber como parâmetro o arquivo de entrada *input.txt* e o arquivo de saída *output.txt*, conforme demonstrado a seguir:

```
./tp2 -i input.txt -o output.txt
```

O arquivo *input.txt* deve conter as seguintes informações, na ordem indicada abaixo:

1. número de instâncias do problema;
2. número de vértices v de um polígono;
3. v linhas com: coordenadas x e y para cada vértice do polígono.

As coordenadas dos vértices de id entre 1 e v são apresentadas em ordem no arquivo *input.txt* por duas partes inteiras, numerador e denominador. A ordem de apresentação desses vértices segue o sentido anti-horário do polígono. Deve-se assumir que v é maior que zero, nunca há um vértice repetido e todos os vértices estao no primeiro quadrante (sendo assim, tanto x quanto y são valores reais não-negativos). Considere que podem haver entradas com polígonos de até 1000 vértices.

Exemplo de entrada genérico (arquivo *input.txt*):

```
1 // número de instâncias
4 // número v de vértices do polígono
1/1 1/1 // coordenada (1,1) do vértice de id = 1
100/2 1/1 // coordenada (50,1) do vértice de id = 2
500/10 50/1 // coordenada (50,50) do vértice de id = 3
1/1 100/2 // coordenada (1,50) do vértice de id = 4
```

A saída do programa deve ser impressa no arquivo *output.txt* e deve conter as seguintes informações, na ordem indicada abaixo, por heurística:

1. número g de guardas utilizados;
2. g linhas com: id do guarda, seguido dos $id's$ dos vértices que esse guarda vigia. Os guardas devem ser apresentados em ordem crescente de $id's$, assim como os $id's$ dos vértices que o guarda vigia.

Exemplo de saída genérico (arquivo *output.txt*):

```
#Heuristica1:
1 // um guarda vigia a instância
2 : 1 2 3 4 // guarda de id = 2 vigia os vértices de id = {1, 2, 3, 4}
#Heuristica2:
2 // dois guardas vigiam a instância
1 : 1 2 3 4 // guarda de id = 1 vigia os vértices de id = {1, 2, 3, 4}
2 : 1 2 3 4 // guarda de id = 2 vigia os vértices de id = {1, 2, 3, 4}
```

Os comentários nos exemplos de entrada e saída foram apresentados apenas por propósitos didáticos, logo eles **NÃO DEVEM** ser incluídos nos arquivos para submissão. Além disso, entrada e saída padrão devem seguir rigorosamente o formato descrito, inclusive no caso de múltiplas instâncias. Sendo assim, em ambos os arquivos, dados de diferentes instâncias devem ser apresentados na mesma ordem e em sequencia (na linha seguinte do último dado da instância anterior). Instâncias distintas para o problema devem ser geradas pelo próprio aluno para testar e avaliar seu algoritmo.

2 O que deve ser entregue:

2.1 Documentação: deve abranger pelo menos os seguintes pontos

- Introdução do problema apresentado.
- Modelagem e solução do problema: explique as heurísticas utilizadas para resolver o problema, tais como suas respectivas complexidades de tempo e espaço.
- Principais decisões de implementação.
- Análise comparativa entre as heurísticas.
- A documentação não pode exceder 10 páginas.

2.2 Código:

- O código fonte do trabalho deve ser submetido para compilação e execução em ambiente Linux, tendo como padrão os computadores dos laboratórios de graduação do DCC;
- Deve ser **obrigatoriamente** escrito na linguagem C (trabalhos implementados em outras linguagens como C++/Java/Python e outras **não** serão aceitos);
- As estruturas de dados devem ser alocadas dinamicamente e o código deve ser modularizado (ou seja, dividido em múltiplos arquivos fonte e fazendo uso de arquivos cabeçalho - .h);
- O utilitário Make deve ser utilizado para compilar o programa **(o arquivo de makefile deve ser submetido juntamente com o código fonte)**;
- A saída deve ser impressa seguindo estritamente o formato da especificação, caso contrário o resultado será considerado errado;
- O arquivo executável deve ser chamado de **tp2**. Não serão aceitos outros nomes de executáveis além do mencionado;
- Faça seu código de forma legível.

2.3 Entrega:

- Data de entrega: 23/04/2012
- Submissão: a documentação e o código do trabalho devem ser submetidos ao *minha.ufmg*. Para isso, compacte os dois (formato *tp2_ NomeSobrenome.zip*) e faça a submissão. Teste seu arquivo compactado antes de enviá-lo.
- Apenas a documentação deve ser entregue impressa na secretaria do DCC. Não coloque nos escaninhos dos professores, entregue para a secretária para que sua documentação seja colocada no envelope de AEDS3. A documentação impressa pode ser entregue no dia útil seguinte r submissão digital. **Trabalhos que não tiverem a documentação entregue na secretaria, dentro do prazo de entrega, receberao nota 0.**
- Será postada uma planilha no Moodle sobre a entrevista do trabalho, leia-a e siga as orientações para o agendamento da sua entrevista.
- Será adotado média harmônica entre a pontuação obtida na execução e na documentação do TP, o que implica em valor zero caso alguma das partes não seja apresentada.
- A política para desconto por atraso de entrega do trabalho prático considera a fórmula:

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

onde d é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.

Observações

- O formato de submissão **DEVE** ser respeitado: *tp2_NomeSobrenome.zip*. Por favor, **NÃO** submetam trabalhos em outros formatos (.rar, .tar, .tar.gz). Certifique-se que seu nome completo foi incluído no nome do arquivo.
- **EXCLUA** o arquivo executável do arquivo compactado a ser submetido no *minha.ufmg*.
- **INCLUA** seu email na capa da documentação.

Referências

Projeto de Algoritmos com implementação em Pascal e C. Nívio Ziviani - *http://www.dcc.ufmg.br/algoritmos/*.

Introduction to Algorithms. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. MIT Press, 3rd edition, 2009.

Algorithms in C - capítulo 24. Robert Sedgewick. Addison Wesley Professional, 1st edition, 2001.

Algorithm Design. Jon Kleinberg, and Éva Tardos. Addison Wesley, 1st edition, 2005.