Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Programa de Pós-Graduação em Ciência da Computação
Projeto e Análise de Algoritmos - 1ºsemestre de 2010
Professor: David Menotti

Trabalho Prático 2

Para cada um dos exercícios propostos abaixo (exceto o primeiro), além da implementação de algoritmos exigida, deve-se:

1. Executar os algoritmos para 5 instâncias de tamanhos e valores diferentes. Procure utilizar instâncias que levem os algoritmos a exceder os limites de espaço (memória física de seu computador) e tempo (mais de 10 minutos).

2. Comparar os resultados obtidos analisando-os e discusando sobre eles.

Valor: 1,0 pontos (10% da nota total) Data de Entrega: 18/05/2010

1. (Tentativa e Erro) Um calouro do curso de Ciência da Computação da Universidade Federal de Ouro Preto, Tibúrcio Monte Carlo, resolveu comemorar o final do semestre escolar. Em seu bar preferido no centro, o famoso buteco Barroco, ele consumiu vasta quantidade de cerveja, e logo após estava bastante embriagado. No momento de sair, lembrou que seus companheiros de república estavam nas ruas, prontos para trotear os calouros bêbados. Apesar do estado inebriante de sua mente, ele percebeu que poderia evitar o encontro de seus companheiros enquanto voltava para casa.

   Sua estratégia para tentar chegar em casa foi a seguinte: ao chegar em uma esquina, ele olharia em direção aos cruzamentos seguintes, um de cada vez, e prosseguiria em direção à primeira esquina onde não houvesse sinal de seus companheiros. Com um pouco de esperteza, ele percebeu que poderia caminhar em ziguezague, mas de forma que não voltasse a um mesmo cruzamento que já tivesse visitado. Chegando a um cruzamento, se a única rua não bloqueada fosse exatamente aquela por onde havia vindo, ele retornaria por esta rua e continuaria o algoritmo a partir do cruzamento anterior.

   Apesar dele pertencer a uma república numerosa que contava com cerca de 50 membros e estes estavam quase que em sua maioria nas ruas, os membros conseguiam cobrir somente 40% dos cruzamentos. Isto é, a probabilidade de que Tibúrcio Monte Carlo possa prosseguir em direção a qualquer esquina deve ser $p = 0.6$, e a probabilidade de que um de seus companheiros estejam em

qualquer esquina é $p = 0.4$. Após sua saída, seus companheiros chegaram ao Barroco. Caso ele retornasse pelo caminho utilizado até o buteco, seria pego e levaria um trote daqueles.

(a) Elabore um mapa do centro histórico de Ouro Preto, em formato de grafo, com pelo menos 50 cruzamentos, criando a localização do Barroco, da casa de Tibúrcio Monte Carlo, e a casa de sua namorada, lugar igualmente considerado seguro dos trotes de seus veteranos.

(b) Implemente um algoritmo[1] usando o paradigma de **tentativa e erro** que encontre um caminho para Tibúrcio chegar em sua casa (ou de sua namorada). Este algoritmo deve apresentar o número de ocasiões que Tibúrcio teve que retornar por não ter conseguido um novo caminho para chegar em um lugar seguro.

2. (Divisão-e-Conquista) O uso mais comum das transformadas de Fourier, e consequentemente da transformada rápida de Fourir (FFT - Fast Fourier Transform), é no processamento de sinais. Um sinal é dado no **domínio do tempo** como uma função de mapeamento de tempo para amplitude. A análise de Fourier nos permite expressar o sinal como uma soma ponderada de senóides de frequências variadas deslocadas em fase. Os pesos e as fases associados com as frequências caracterizam o sinal no **domínio de frequência**. Análises em domínio na frequência permitem realizar filtragens não factíveis no domínio do tempo.

   Implementações ingênuas desta transformada tem ordem de complexidade quadrática, *i.e.*, $O(n^2)$.

   - Implemente um algoritmo usando o paradigma de **divisão-e-conquita** para computar a FFT de um sinal discreto unidimensional, cujo tamanho é uma potência 2, com complexidade $O(n \lg n)$.
   - Implemente também um algorithmo para computar a inversa da FFT.

3. (Programação Dinâmica) Seja

$$M = M_1 \times M_2 \times ... \times M_n.$$

   onde M_i é uma matriz com $d_{i..1}$ linhas e d_i colunas, $2 \leq i \leq n$. Saiba que a ordem da multiplicação pode ter um efeito enorme no número total de operações de adição e multiplicação necessárias para obter M. Considere que o produto de uma matriz $p \times q$ por outra matriz $q \times r$ requer $O(pqr)$ operações. Tentar todas as ordens possíveis de multiplicação de matrizes para minimizar o número de operações $f(n)$ para multiplicar as n matrizes é exponencial em n, onde $f(n) \geq 2^{n-2}$.

   - Implemente um algoritmo usando o paradigma de **programação dinâmica** para calcular a ordem de multplicação das n matrizes que gera o número mínimo de operações, cuja ordem de complexidade seja $O(n^3)$.

[1] Interfaces gráficas ilustrando Tibúrcio voltando para um lugar seguro valem até 0,3 pontos extra.

4. (Algoritmos Gulosos) Considere o problema inteiro da mochila:

   - Um ladrão acha n itens numa loja.
   - Item i vale v_i unidades (dinheiro, *e.g.*, R$, US$, etc).
   - Item i pesa p_i unidades (kg, etc).
   - v_i e p_i são inteiros.
   - Consegue carregar P unidades no máximo.
   - Deseja carregar a "carga" mais valiosa.

   - Implemente três algoritmos usando abordagens **gulosas** para calcular os itens que podem ser carregados na mochila respeitando a capacidade P.

5. (Algoritmos Aproximados) Uma instância do problema de soma de subconjuntos é um par (S, t), onde S é um conjunto $\{x_1, x_2, ..., x_n\}$ de inteiros positivos e t é um inteiro positivo. Esse problema de decisão pergunta se existe um subconjunto de S cuja soma seja exatamente t. Esse problema é NP-completo.

   O problema de otimização associado a esse problema de decisão surge em aplicações práticas. No problema de otimização, desejamos encontrar um subconjunto de $\{p_1, p_2, ..., p_n\}$ cuja soma seja tão grande quanto possível, mas não maior que t. Por exemplo, podemos ter um caminhão que não pode transportar mais de t quilogramas, e ter n caixas diferentes para transportar, das quais a i-ésima caixa pesa p_i quilogramas.

   - Implemente um algoritmo **aproximado** para encher o caminhão com a carga mais pesada possível, sem exceder o limite de peso dado.

O que deve ser entregue

- Código fonte dos programas em C ou C++ (bem identado e comentado).
- Documentação do trabalho.

Entre outras coisas, a documentação deve conter:

1. Introdução: descrição de cada problema a ser resolvido.
2. Implementação: descrição sobre a implementação do programa. Muito importante: os códigos utilizados nas implementações devem ser inseridos na documentação.
3. Análise de Complexidade: estudo da complexidade de tempo e espaço das funções implementadas.
4. Análise de Resultados: comparar os dados obtidos e discussar sobre estes.
5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.

6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sítio da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto onde é utilizada.

7. Em LaTeX: A documentação deve ser elaborada obrigatoriamente em LaTeX. Veja modelo de como fazer o trabalho em latex: `http://www.decom.ufop.br/menotti/paa101/tps/modelo.zip`

8. Formato final: mandatoriamente em PDF (`http://www.pdf995.com/`).

Como deve ser feita a entrega

A entrega DEVE ser feita via Moodle (`www.decom.ufop.br/moodle`) na forma de um único arquivo zipado, contendo o código fonte, arquivos diversos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

Comentários Gerais

- A maioria destes problemas foram extraídos de [2, 1, 3, 4] e as vezes de alguma forma alterados;
- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero. Devido a recorrentes problemas com cópias de trabalhos (plágios), os autores de trabalhos copiados também terão a maior nota dentre os testes teóricos levada a zero, como forma de punição e coação ao plágio acadêmico;
- Trabalhos entregues em atraso terão descontados 0,1 pontos por hora;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Referências

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press/McGraw-Hill, 2nd edition, 2001.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press/McGraw-Hill, 3rd edition, 2009.

[3] N. Ziviani. *Projeto de Algoritmos: com implementações em Pascal e C*. Cengage Learning (Thomson / Pioneira), São Paulo, 2nd edition, 2004.

[4] N. Ziviani. *Projeto de Algoritmos com implementações em Java e C++*. Cengage Learning (Thomson / Pioneira), São Paulo, 1st edition, 2006.