

PUC-Rio
Departamento de Informática
Prof. Marcus Vinicius S. Poggi de Aragão
Horário: 4as-feiras de 13 às 16 horas - Sala 452L
25 de abril de 2007
Data da Entrega: 7 de junho de 2007
Período: 2007.1

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

1º Trabalho de Implementação

Descrição

Este trabalho prático consiste em desenvolver códigos para diferentes algoritmos e estruturas de dados para resolver os problemas descritos abaixo e, principalmente, analisar o desempenho das implementações destes algoritmos com respeito ao tempo de CPU. O desenvolvimento destes códigos e a análise devem seguir os seguintes roteiros:

- Descrever os algoritmos informalmente.
- Demonstrar o entendimento do algoritmo explicando, em detalhe, o resultado que o algoritmo deve obter e justificá-lo.
- Explicar a fundamentação do algoritmo e justificar a sua corretude. Apresentar e explicar a complexidade teórica esperada para cada algoritmo.
- Apresentar as tabelas dos tempos de execução obtidos pelos algoritmos sobre as instâncias testadas, comparando sua evolução com a evolução dos tempos seguindo a complexidade teórica correspondente.
- Documente o arquivo contendo o código fonte de modo que cada passo do algoritmo esteja devidamente identificado e deixe claro como este passo é executado.
- Para a medida de tempo de CPU das execuções utilize as funções disponíveis no link correspondente na página do curso, um exemplo de utilização é apresentado. Quando o tempo de CPU for inferior à 5 segundos, faça uma repetição da execução tantas vezes quantas forem necessárias para que o tempo ultrapasse 5 s (faça um while), conte quantas foram as execuções e reporte a média.

A corretude código será testada sobre um conjunto de instâncias que será distribuído. O trabalho entregue deve conter:

- Um documento contendo o roteiro de desenvolvimento dos algoritmos (e dos códigos), os itens pedidos acima, comentários e análises sobre a implementação e os testes realizados (papel).

- A impressão dos trechos relevantes dos códigos fonte (papel).
- Um e-mail para poggi@inf.puc-rio.br (é obrigatório o uso do ASSUNTO (ou SUBJECT) PAA071T1) deve ser enviado contendo os arquivos correspondentes ao trabalho.
- O trabalho pode ser feito em grupo de até 5 alunos.

0. Estruturas de Dados

O grupo deve implementar (ou usar códigos prontos) códigos para efetuar as seguintes operações nas estruturas de dados abaixo:

1. Árvore Balanceada (Árvore AVL, Árvore Vermelha-Preta, etc.)
2. *Heap* que permita a união de *heaps* (*Leftist Heap* do Knuth), com e sem operações *preguiçosas* (**LAZY**).

1. Problema da Árvore Geradora Mínima

1. Implementar o Algoritmo de Prim utilizando as estruturas de dados, listadas a seguir, para selecionar o vértice mais próximo da árvore corrente. Nestas estruturas, cada vértice tem como valor-chave o peso da menor aresta que o conecta à árvore corrente. (ver links na página do curso para textos sobre algoritmos para a MST, em especial para o algoritmo de Round-Robin, ver também links para instâncias do problema).

Lista de estruturas de dados a utilizar:

- (a) Árvore Balanceada de Busca
- (b) *Heap* sem *lazy*.
- (c) *Heap* com *lazy*.

A *Heap* sugerida para ser implementada é a *Leftist Heap* ver texto no link heap na página do curso.

2. Implementar o Algoritmo de Round-Robin (Tarjan) (equivalente ao algoritmo de Solin ou Borůvka) nesse algoritmo inicia-se com uma árvore associada a cada vértice (n árvores) armazenado-se numa *min heap* as arestas que ligam cada árvore ao restante do grafo. A cada iteração uma árvore é conectada a uma outra e suas *min heaps* combinadas. A ordem em que as árvores são combinadas segue o critério FIFO onde a ordem inicial é arbitrária (1,2,...,n por exemplo). Utilize as seguintes *heaps* com operação de união:

- (a) *Heap* sem *lazy*.
- (b) *Heap* com *lazy*.

2. Problema da Mochila Fracionária (pode-se colocar parte de um objeto na mochila)

1. Implementar os algoritmos colocam na mochila os itens de maior razão valor/peso com as seguintes complexidades teóricas em função do número n de itens candidatos a serem colocados na mochila:

(a) $O(n \log n)$

(b) $O(n)$

3. Encontrar os componentes fortemente conexos de um grafo orientado.

1. Implementar algoritmos com a seguinte complexidade::

(a) $O(n^2 + nm)$

(b) $O(|C|. (n + m))$ onde C é o conjunto de componentes fortemente conexos.

(c) $O(n + m)$

2. Utilize esse algoritmo para determinar se uma instância do problema 2-SAT pode ser satisfeita ou não.