PUC-Rio
Departamento de Informática
Prof. Marcus Vinicius S. Poggi de Aragão
Horário: 4as-feiras de 13 às 16 horas - Sala 511 RDC
16 de junho de 2017
Data da Entrega: 13 de julho de 2017
Período: 2017.1

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

2^o 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:

- Um e-mail para poggi@inf.puc-rio.br (é obrigatório o uso do ASSUNTO (ou SUBJECT) PAA171T2 deve ser enviado contendo os arquivos correspondentes ao trabalho. O NÃO ENVIO DESTE E-MAIL IMPLICA QUE O TRABALHO NÃO SERÁ CONSIDERADO.
- 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.
- Obrigatoriamente apresente tabelas contendo três de colunas para cada algoritmo aplicado às instâncias, uma com o valor da complexidade teórica, uma com o tempo de CPU utilizado e uma com a razão destes dois valores. Cada linha da tabela é associada a uma instância e contém a identificação da mesma. Nesta tabela coloque as instâncias em ordem crescente de tamanho.

A corretude código será testada sobre um conjunto de instâncias que será distribuido. 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).
- O trabalho pode ser feito em grupo de até 4 alunos.

O objetivo do 2^o ***Trabalho é a implementação e a avaliação experimental algoritmos para um problema NP-difícil. Isto é, um problema de otimização cuja versão de decisão é um problema NP-completo. O objetivo prioritário é provar que soluções são ótimas, ou seja, que dado o valor da solução ótima, que não existe solução de valor inferior. Eventualmente, soluções ótimas de instâncias do problema proposto podem ser encontradas e provadas ótimas. Quando não se consegue provar que uma solução é ótima, deseja-se conhecer o melhor(maior) limite inferior para o valor ótimo (problema de minimização). Este é o foco destes trabalho.***

- **Problema da Árvore Geradora com Restrições de Capacidade (CAP-MST):**

  Dado um grafo $G = (V \bigcup \{r\}, E)$, custos $c(i, j) \geq 0$ associados às arestas (i, j) em E, demandas $q(v)$ associadas aos vértices v em V, e uma capacidade Q. Deseja-se encontrar uma árvore geradora de G, cujas sub-árvores enraizadas no vértice r possuam a soma das demandas dos seus vértices inferiores ou iguais à Q, cujo custo total de suas arestas seja **mínimo**.

- As instâncias estão no link CAP-MST da página do curso.

- **Aplicação:** Um contexto onde este problema é importante, considere que no vértice r estão situados roteadores onde cada um dos seus *slots* tem uma capacidade, C MB/s por exemplo. Assim, a soma da demanda de todos os clientes na árvore conectada em cada *slot* não pode ultrapassar a capacidade de tráfego do *slot*, no caso C MB/s.

- **Algoritmo:** A apresentação do seu algoritmo deve conter uma descrição de cada um dos elementos em um *branch-and-bound*, são eles:

  - Critério de particionamento do espaço de soluções. Sugere-se que o critério utilizado seja o de particionar em dois conjuntos: um onde a árvore geradora contém obrigatoriamente uma dada aresta; e outro onde esta aresta não está na árvore.
  - Um critério para percorrer os subconjuntos do espaço de soluções. Para facilitar a implementação, sugere-se o critério de busca em profundidade.
  - Uma relaxação do problema (uma função que gera sempre um valor inferior ou igual ao da solução ótima para o subconjunto de soluções considerado). Neste item, mostre como a sua relaxação é modificada quando se considera apenas um subconjunto do espaço de soluções (ou seja, quando um elemento da solução é fixado, neste caso isto corresponde a uma aresta ter que estar obrigatoriamente na árvore geradora ou obrigatoriamente fora dela). Uma relaxação simples é a árvore geradora mínima que não considera a restrição de capacidade.
  - Um método para obter uma "boa" solução viável (que seria a melhor solução conhecida antes de iniciar o *branch-and-bound*). Esta solução pode ser obtida por um método guloso, por exemplo, que inicie com cada vértice em uma sub-àrvore e proceda reunindo pares de sub-árvores que reduzem ao máximo o custo da árvore geradora corrente e não violem a capacidade da sub-árvore.
  - Critério de seleção do particionamento. Uma possibilidade é ordenar as arestas pelos seus respectivos custos. Outras ordens podem ser tentadas.

- **Resultados:** Apresente sempre a sua melhor solução (lista das arestas com seus respectivos valores) e o seu valor total. Caso o tempo de CPU ultrapasse 1 hora, faça com que seu algoritmo termine imprima a melhor solução encontrada até então. Apresente sempre, também, o valor do limite inferior para a solução ótima no momento em que a enumeração foi interrompida. No caso de um branch-and-bound, o limite inferior é o limite de menor valor entre os ramos abertos da enumeração.

  Deverá ser apresentado um relatório sobre as experiências computacionais comentando os resultados obtidos. Este relatório deverá conter:

  - Uma tabela com o valor da melhor solução obtida, indicando se é ótima (provado pelo seu algoritmo) ou não, o tempo de cpu total utilizado na resolução, o valor do limite inferior obtido no nó raiz e o limite inferior obtido ao final da execução, e o valor da solução ótima

(ou melhor conhecida) que serão fornecidos. A tabela deverá ter uma linha para cada uma das 39 instâncias contidas no arquivo no link;

- Uma análise dos resultados com relação à complexidade assintótica do algoritmo implementado. Mostrar que o crescimento do tempo é exponencial em função do tamanho da instância;
- Uma análise separada das diferentes etapas do algoritmo.

- **Relaxações**
  - A relaxação mais simples é a árvore geradora mínima (AGM) do grafo;
  - A melhoria da relaxação ajuda muito a resolução de um problema desse tipo. Uma forma de tornar a relaxação dada pela AGM mais forte é determinar o número mínimo de arestas n_a que precisam estar conectadas na raiz (Quantas são?) e, caso a AGM tenha menos que n_a arestas, calcula-se todas as possíveis AGMs com n_a arestas ligadas à raiz. O menor dos valores será o novo limite inferior para a CAP-MST;
  - Existem mais formas de reforçar essa relaxação. Existem também outras relaxações.