

PUC-Rio
Departamento de Informática
Profs. Marcus Vinicius S. Poggi de Aragão
Período: 2008.2
Horário: 2as-feiras e 4as-feiras 17-19
28 de outubro de 2008
Data da Entrega: 5 de dezembro de 2008

ESTRUTURAS DISCRETAS (INF 1631)

2º Trabalho de Implementação

ENTREGA DO TRABALHO

O trabalho pode ser feito em grupos de até **3 (três)** alunos. O trabalho entregue deve conter:

- Um documento contendo uma análise do problema e do algoritmo proposto para a sua resolução. Discuta sobre possíveis alternativas. Apresente comentários e análises sobre a implementação e os testes realizados (**PAPEL**).
- A impressão do código fonte (papel).
- Um e-mail para **poggi@inf.puc-rio.br** contendo um arquivo ZIP (renomeado para .zzz) contendo o código fonte e o executável (por isso o .zzz) correspondente. **(É obrigatório o uso do ASSUNTO (ou SUBJECT) ED082T2). O NÃO CUMPRIMENTO DESTES REQUISITOS IMPLICA NA NOTA ZERO.**

OPÇÃO I

Descrição

Este trabalho prático consiste em desenvolver um algoritmo de enumeração (implícita) e o código correspondente. O objetivo é o desenvolvimento de algoritmos para enumeração de estruturas combinatorias. O desenvolvimento do código e a análise deve seguir o seguinte roteiro:

- Descrever o algoritmo 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.
- 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.

1. O problema

O problema que o algoritmo deve resolver é:

(K-Color):

Instância: Dado um grafo $G=(V,E)$ e um inteiro K , $0 < K < |V|$.

Questão: Existe uma atribuição $c : V \rightarrow \{1, \dots, K\}$ tal que $\forall (v, w) \in E, c(v) \neq c(w)$?

2. O algoritmo

O algoritmo deve apresentar uma atribuição que verifica que a instância dada pode ser “colorida” com K cores ou demonstrar que tal atribuição não existe.

Para esta segunda parte é necessário testar todas as possibilidades de atribuir K valores aos vértices do grafo o que, entretanto, não obriga que todas as possíveis atribuições sejam testadas. Veja, por exemplo, o algoritmo de *BackTrack* apresentado na seção 7.3 do livro Horowitz e Sahni, “Fundamentals of Computer Algorithms”.

Um algoritmo deste tipo funciona da seguinte forma: para cada vértice armazena-se o conjunto das cores que este pode receber (inicialmente $\{1, \dots, K\}$). A cada iteração um vértice recebe uma cor deste conjunto, operação que retira esta cor dos conjuntos de cores possíveis de todos os vértices a ele adjacentes. Esta operação se repete até que todos os vértices tenham recebido uma cor e o algoritmo termina (visto que nesse momento uma atribuição válida foi encontrada), ou até que todos os vértices ainda sem cor atribuída tenham seu conjunto de cores possíveis vazio. Neste, faz-se um *backtrack*. Ou seja, o último vértice que recebeu uma cor troca sua cor por outra do seu conjunto de cores possíveis que ainda não recebeu. Observe que esta troca de cor modifica os conjuntos de cores possíveis de todos os seus vizinhos. Caso todas as cores do seu conjunto tenham sido testadas, faz-se mais um nível de *backtrack* deixando este vértice sem cor atribuída (e atualizando os conjuntos de cores possíveis dos vizinhos) e refazendo esta mesma operação para o vértice que havia sido colorido imediatamente antes.

Sugere-se que esta implementação seja feita utilizando recursão, para reduzir o esforço de codificação.

Finalmente, caso se *aposte* que existe uma atribuição válida, pode-se tentar encontrá-la por métodos heurísticos, isto é, sem garantia de sucesso e que quando não encontram uma atribuição válida, não geram nenhuma informação com relação a sua existência.

Uma heurística simples e frequentemente eficaz é aplicar uma busca em profundidade que atribui aos vértices visitados sempre a menor cor possível.

3. Aplicação e Testes

Uma aplicação do algoritmo proposto acima é jogo conhecido como SUDOKU. Neste jogo um mesmo grafo aparece com alguns vértices já coloridos e o objetivo é encontrar uma atribuição de 9 cores aos vértices ainda sem cor atribuída.

Os testes devem ser feitos sobre um conjunto de instâncias de grafos com valores de K indicados e um subconjunto dos vértices com as respectivas cores atribuídas. Em particular haverá uma série

de instâncias do SUDOKU. Nestes caso, como sabe-se que existe uma atribuição deve-se explicitar quando foi usada uma heurística e quando não.

As instâncias estão disponíveis na página do curso.

OPÇÃO II

SEGURANÇA ESTATÍSTICA DE DADOS - Descrição

O problema de Segurança Estatística de Dados está presente quando um órgão público de levantamento de dados, como o IBGE por exemplo, divulga relatórios onde além de dados agregados divulga também alguns dados individuais, mas sem revelar outros dados.

De forma mais precisa, considere que uma tabela $T(n, m)$ de n colunas e m linhas armazena as informações em questão, que no caso são valores inteiros positivos. As somas dos valores nas linhas e nas colunas tem que ser divulgados e além deles o máximo de valores individuais, desde que estes valores divulgados não permitam inferir outros valores da tabela.

Por exemplo, cada coluna pode se referir a uma cidade e cada linha a um setor de atividade do governo. Os valores podem ser os gastos do governo com a atividade associada à linha: saúde, educação, infra-estrutura, etc. Pode ser que os valores destes gastos sejam "confidenciais", mas que alguns tenham que ser revelados, além dos totais por cidade e por atividade para um conjunto de cidades, do mesmo estado por exemplo. O órgão deve divulgar os valores totais por cidade e por atividade e alguns exemplos de gastos com certas atividades em certas cidades (mas só os que agradam!). De forma alguma, os dados divulgados devem permitir que sejam inferidos outros valores (o que permitiria deduzir que o gasto com saúde em uma dada cidade foi exageradamente baixo, por exemplo).

Assim, para uma tabela $n \times m$ de elementos t_{ij} positivos ou nulos, são dadas as somas das linhas r_i para $i = 1, \dots, m$ e as somas das colunas c_j para $j = 1, \dots, n$; e uma lista de p posições da tabela $\{(i_1, j_1), (i_2, j_2), \dots, (i_p, j_p)\}$ que tem seus valores associados $t_{i_k j_k}$, $k = 1, \dots, p$, divulgados, deseja-se determinar se alguma outra posição na tabela também fica determinada.

ALGORITMO PARA RESOLUÇÃO

Para determinar se uma dada posição está com o seu valor determinado construa um grafo bipartido $G = (L, C, E)$ onde L é o conjunto de vértices associado às linhas da matriz, C é o conjunto de vértices associado às colunas da matriz, e E o conjunto de arcos composto dos arcos que saem de cada vértice de L para todos os vértices de C .

Associe a cada vértice em L um fluxo fixo de valor r_i que entra no vértice e a cada vértice em C um fluxo fixo de valor c_j que sai do vértice. Observe que a conservação de fluxo (tudo que entra tem que ser igual a tudo que sai) nos vértices do grafo correspondem exatamente às somas dos valores das linhas (vértices em L) e às somas dos valores das colunas (vértices em C).

Em seguida, para cada valor divulgado $t_{i_k j_k}$, $k = 1, \dots, p$, retire o arco $(i_k j_k)$ do grafo e subtraia $t_{i_k j_k}$ do fluxo que entra no vértice i_k em L e do fluxo que sai de j_k em C . Represente por r'_i e c'_j os

valores dos fluxos que entram e que saem dos vértices de L e C , respectivamente, após a retirada dos valores conhecidos.

Obtenha para o grafo resultante $G' = (L, C, E')$ um **fluxo viável** onde os fluxos fixos tem valor de entrada r' para os vértices de L e onde fluxos fixos tem valor de saída c' para os vértices de C . Isto é, valores positivos ou nulos x_{ij} para cada arco em E' que satisfaçam a conservação de fluxo em cada vértice.

Para isso associe aos arcos em E' capacidades de fluxo da seguinte forma: o arco (i, j) terá capacidade de fluxo máxima u_{ij} igual ao menor valor entre r'_i e c'_j . Assim, $0 \leq x_{ij} \leq u_{ij}$ para todo arco de E' . Siga o algoritmo abaixo:

1. Faça inicialmente $x_{ij} = 0$ para todo arco em E' . Seja x esse fluxo.
2. Determine o grafo $G'(x)$, a rede residual, onde para cada arco (i, j) de E' existirá em $G'(x)$ um arco (i, j) com capacidade $u_{ij} - x_{ij}$, se $x_{ij} < u_{ij}$, e um arco (j, i) com capacidade x_{ij} , se $x_{ij} > 0$.
3. Se existir pelo menos um vértice em L e pelo menos um vértice em C onde ainda não há conservação de fluxo, vá para 4. Senão, x é um fluxo viável. FIM.
4. Encontre um caminho em $G'(x)$ de um vértice em L (ainda sem conservação de fluxo) a um vértice em C (também ainda sem conservação de fluxo). Seja u_{min} a menor capacidade entre os arcos deste caminho. Aumente de u_{min} o valor de x_{ij} de todos os arcos do caminho, se o arco do caminho for (j, i) subtraia u_{min} de x_{ij} . Obtem-se assim um novo fluxo x . Volte para 2.

Para determinar se um dado valor desconhecido t_{ij} está determinado, basta verificar na rede residual do fluxo viável x , $G'(x)$, se existe caminho de i para j e de j para i , onde esse caminho não pode usar os arcos (i, j) e (j, i) . Se existir um dos dois caminhos, o valor NÃO está determinado. Se nenhum dos dois existirem, o valor estará determinado e x_{ij} será este valor.

EXPERIMENTAÇÃO

1. Programe o algoritmo obtido descrito acima.
2. Execute o algoritmo para o arquivo de dados deste 2o. Trabalho (disponível na página *web* do curso). A especificação dos arquivo de entrada e as saídas desejadas encontram-se mais abaixo.
3. Determine o tempo de CPU gasto para cada caso teste no arquivo de dados (utilize o mesmo procedimento do 1o. Trabalho).
4. BONUS: Quando o valor da posição pedida não pode ser obtido, é possível determinar um intervalo para este valor. Será concedido um bonus de 50% (ou seja o trabalho vale 15!) para os trabalhos que não somente determinarem se os valores pedidos podem ser inferidos, mas que forneçam os intervalos possíveis para todos as posições pedidas das tabelas.
5. Ainda no bonus, determine os tempos de CPU de cada caso teste.

CONCLUSÕES

Escreva suas conclusões analisando os resultados obtidos e o tempos de CPU. Em que condições os valores das tabelas podem ser estimados com uma boa precisão ? Você considera que o algoritmo implementado é eficiente ?

Descrição do Arquivo de Entrada

A primeira linha contém um inteiro N contendo o número de casos teste. Cada caso teste possui na sua primeira linha um inteiro m . A linha seguinte possui m inteiros correspondendo às somas r_i das linhas da matriz (desconhecida). Na próxima linha está um inteiro n que é o número de colunas da matriz. Segue uma linha com os n inteiros correspondentes às somas dos valores nas colunas. Logo abaixo está um número p que é o número de posições reveladas da matriz. Cada uma das p linhas que seguem possuem 3 inteiros: i , j e t . Onde t é o valor da posição (i, j) que é revelada, sendo $1 \leq i \leq m$ e $1 \leq j \leq n$. Finalmente, a linha seguinte contém um inteiro q que define o número de posições em que deverá ser verificado se o seu valor pode ser obtido a partir das informação anteriores. Estas posições aparecem nas q linhas que seguem onde cada uma possui dois inteiros i e j indicando a posição. O arquivo termina com um 0(zero). Um exemplo com 1 caso teste segue.

```
1
4
29 31 28 13
4
26 22 14 39
5
2 2 5
2 4 21
3 1 3
3 4 12
4 3 1
3
1 3
2 1
4 4
0
```

Descrição do Arquivo de Saída

O arquivo de saída contém o número do caso teste e as resposta sim ou não para cada posição que deseja-se saber se pode ser obtida. No caso afirmativo, apresente o valor.

No caso da versão BONUS, apresente para o caso negativo os intervalos menores possíveis que podem ser inferidos.