UFLA – Universidade Federal de Lavras
Departamento de Ciência da Computação
COM167 – Teoria da Computação
Prof. Rudini Sampaio
Monitor: Rodrigo Pereira dos Santos

Terceira Lista de Exercícios – **2005/1**

....

Exercício 1 Mostre que se ***P*** = ***NP***, então existe um algoritmo polinomial que, dado um grafo *G*, **obtém vértices** de *G* que formam uma clique máxima. **Dica:** Obtém o tamanho da clique máxima e depois vai removendo vértices cuja remoção não diminui esse tamanho.

RESPOSTA =

CLIQUE = {<G, K> | G tem uma clique de tamanho K}.

Vamos provar que CLIQUE Є NP. Para isso, vamos construir um verificador polinomial para a linguagem CLIQUE.

V = "Com entrada <G,K,C>, onde G é um grafo, V(G) é seu conjunto de vértices e E(G) é seu conjunto de arestas. C é o conjunto de vértices a ser verificado e K é o tamanho da clique.

1. Para todo v Є C, verifique se v possui arestas aos demais vértices de C.
2. Se verdadeiro, então:

   2.1 Se $|C| \leq K$, ACEITE.

   2.2 Senão, REJEITE.
3. REJEITE."

Como a realização de comparações e testes correspondem a operações em tempo polinomial, o verificador é polinomial. Logo, CLIQUE Є NP.

Se P = NP e CLIQUE Є NP, temos que CLIQUE Є P. Então, existe uma M.T. determinística M que decide a CLIQUE em tempo polinomial, onde M recebe como entrada um grafo G e um número K (<G, K>).

N = "Com entrada <G>:

1. K = 0.
2. Para i de 1 até n, onde n é o número de vértices de G, faça:

   2.1 Simule M com entrada <G, i>.

   2.2 Se M aceita, K = i.

   2.3 Se M rejeita, abandone o "para".
3. Para i de 1 até n, onde n é o número de vértices de G, faça:

   3.1 "Arranque" o vértice i de G.

   3.2 Simule M com entrada <G,K>.

   3.3 Se M rejeita, "devolva" o vértice i ao grafo G.
4. Retorne <G, K>."

■

....

Exercício 2 [7.21 Sipser] *HALF-CLIQUE* = {<*G*> | *G* é um grafo que possui uma clique com mais da metade dos vértices de *G*}. Mostre que a linguagem *HALF-CLIQUE* é NP-Completa.
Dica: Provar que *CLIQUE*. $\leq_P$ *HALF-CLIQUE*.

RESPOSTA =

HALF-CLIQUE = {<G> | G é um grafo que possui uma clique com pelo menos a metade do número de vértices de G}.

DICA: CLIQUE $\leq_p$ HALF-CLIQUE.

Primeiramente, vamos provar que HALF-CLIQUE Є NP. Para isso, basta construir um verificador polinomial para HALF-CLIQUE.

V = "Com entrada <G, C>, onde G é um grafo e C é o conjunto de vértices a ser verificado.

1. Se |C| < (n/2), onde n é o número de vértices total de G, REJEITE.
2. Senão, se todos os vértices C_i Є C possuem arestas ligando-os aos outros vértices de C, ACEITE.
3. REJEITE."

Agora, vamos provar que CLIQUE $\leq_p$ HALF-CLIQUE. Para tal, vamos construir uma função de redução F de CLIQUE para HALF-CLIQUE.

F = "Com entrada <G, K>, onde K é o tamanho da clique a ser testada.

1. Se K < (n/2), onde n é o número de vértices de G.

1.1 Adicione (n – 2K) vértices ao grafo G, com arestas destes para todos os demais vértices de G, ou seja, adicione vértices a G até que n ≤ 2K.

2. Retorne G."

Como HALF-CLIQUE Є NP e CLIQUE $\leq_p$ HALF-CLIQUE (CLIQUE Є NP-Completo), HALF-CLIQUE Є NP-Completo.

■

....

Exercício 3 [7.27 Sipser] *SET-SPLITTING* = {<*S*, *C*> | *S* é um conjunto finito e *C* = {C_1, C_2, ..., C_k} é uma coleção de subconjuntos de *S* tais que os elementos de *S* podem ser coloridos de *vermelho* e *azul* de forma que nenhum C_i tenha todos os seus elementos coloridos com a mesma cor}. Mostre que a linguagem *SET-SPLITTING* é NP-Completa.

Dica: Provar que *VERT-COLOR* $\leq_P$ *SET-SPLITTING*

RESPOSTA =

SET-SPLITTING = {<S, C> | S é um conjunto finito e C = {C_1, C_2, ..., C_k} é uma coleção de subconjuntos de S tais que os elementos de S podem ser coloridos de vermelho e azul de forma que nenhum C_i tenha todos os seus elementos coloridos com a mesma cor}.

DICA: VERT-COLOR $\leq_p$ SET-SPLITTING.

Primeiramente, vamos provar que SET-SPLITTING Є NP, construindo um verificador polinomial para SET-SPLITTING.

V = "Com entrada <S, C, SOL>, onde S e C são tais como descrito acima, e SOL é o conjunto formado por duplas da forma (V_i(G), cor), onde V_i(G) é cada um dos vértices de G, associado à sua cor. SOL deve ser testado.

1. Para i de 1 até k, onde k = |C|, faça:

1.1 Usando o conjunto de duplas SOL, verifique se C_i Є C possui elementos (vértices) coloridos com duas cores.

1.2 Se C_i não segue essa regra, REJEITE.

2. Se todos os C_i Є C seguem a regra, ACEITE.
3. REJEITE."

Agora, vamos provar que VERT-COLOR $\leq_p$ SET-SPLITTING, construindo uma função de redução F de VERT-COLOR para SET-SPLITTING.

F = “Com entrada <G, K>, onde G é um grafo e K é o número de cores para se colorir G.
1. Considere K = 2.
2. Construa o conjunto S, onde cada um de seus elementos é um vértice de G.
3. Construa o conjunto C, onde cada um de seus elementos é um conjunto formado por um subgrafo distinto de G, ou seja, C é formado por uma coleção de conjuntos formados pelos vértices ligados por uma aresta.
4. Retorne S e C.”

Como SET-SPLITTING Є NP e VERT-COLOR $\leq_p$ SET-SPLITTING (VERT-COLOR Є NP-Completo), SET-SPLITTING Є NP-Completo.

■

....

Exercício 4 Prove que o problema *CAIXEIRO* = {Dado um grafo *G* completo, com pesos positivo nas arestas, obtenha o tamanho mínimo de um ciclo que passe por todos os vértices}. Mostre que *CAIXEIRO* é NP-Difícil. **Dica:** Provar que HAMPATH $\leq_P$ CAIXEIRO

RESPOSTA =

CAIXEIRO = {Dado um grafo G completo, com pesos positivo nas arestas, obtenha o tamanho mínimo de um ciclo que passe por todos os vértices}

DICA: HAMCYCLE $\leq_p$ CAIXEIRO.

A redução de um problema NP-Completo para o CAIXEIRO nos prova que o CAIXEIRO é NP-Difícil. Vamos provar que HAMCYCLE $\leq_p$ CAIXEIRO, através de uma função de redução F de HAMCYCLE para o CAIXEIRO.

F = “Com entrada <G>, onde G é um grafo.
1. Crie um grafo G’, com os mesmos vértices de G.
2. A cada aresta de G, adicione uma aresta com peso 1 em G’.
3. A cada par de vértices em G, que não possua arestas entre eles, adicione uma aresta com peso 2 em G’.
4. Retorne G’.”

Se o caixeiro encontrar um ciclo com peso n = |V(G’)|, isso significa que nenhuma das arestas “extras” (de peso 2) inseridas no grafo original G foram usadas e existe um ciclo hamiltoniano em G. Senão, o ciclo terá peso n > |V(G’)|, o que indica que arestas “extras” (de peso 2) foram utilizadas para completar o ciclo e, neste caso, não existe um ciclo hamiltoniano em G.

Como HAMCYCLE Є NP-Completo e HAMCYCLE $\leq_p$ CAIXEIRO, CAIXEIRO Є NP-Difícil.

■

....

Exercício 5 Prove que *PATH* = {<*G*, *s*, *t*> | o grafo *G* possui um caminho entre os vértices *s* e *t*} está em ***P*** e em ***PSPACE***. **Dica:** Inicie em *s* e vai marcando quem pode ser atingido.

RESPOSTA =

PATH = {<G, s, t> | o grafo G possui um caminho entre os vértices s e t}.

A construção de uma M.T. determinística que decide uma linguagem em tempo e espaço polinomiais nos dá a prova de que PATH Є P e de que PATH Є PSAPACE, respectivamente.

M = “Com entrada <G, s, t>, onde G é um grafo e s e t são vértices de G.
1. Marque o vértice s.
2. Para cada um dos V_i Є V(G) vértices marcados, faça:
2.1 Marque todos os vértices que possuem arestas ligando-os ao vértice V_i.

2.2 Repita até que todos os vértices estejam marcados ou se nenhum novo vértice possa ser marcado.
3. Se t está marcado, ACEITE.
4. Senão, REJEITE."

Como o "loop" marca os vértices uma única vez, o algoritmo tem custo n de tempo. Logo como a complexidade é polinomial e o algoritmo é determinístico, PATH Є P.

Como as operações são realizadas em cima de um tipo de estrutura de dados, um vetor de tamanho n, o espaço também tem custo n. Logo, a complexidade é polinomial em espaço e PATH Є PSPACE.

■

....

Exercício 6 **(6.1)** Estabeleça, na forma de uma pergunta, a definição dos problema de decisão correspondentes às linguagens das questões anteriores. **(6.2)** Sobre o problema CAIXEIRO, **(a)** defina seu tipo (min-max), **(b)** seu conjunto de instâncias, **(c)** o conjunto de soluções para uma dada instância, **(d)** qual o custo de uma solução, **(e)** o que é uma solução ótima e **(f)** qual o pior resultado de um algoritmo de aproximação 5/4, se o ótimo é 100. Justifique.

RESPOSTA =

6.1)

<u>QUESTÃO 1:</u> Dado um grafo G e um inteiro K, G possui uma clique de tamanho máximo?
<u>QUESTÃO 2:</u> Dado um grafo G, G possui uma clique com pelo menos a metade dos vértices de G?
<u>QUESTÃO 3:</u> Dado um conjunto finito S e uma coleção de subconjuntos C, os elementos de S podem ser coloridos de vermelho e azul de forma que nenhum C_i Є C tenha todos os seus elementos coloridos com a mesma cor?
<u>QUESTÃO 4:</u> Dado um grafo G completo com pesos positivos nas arestas e um número K, G possui um ciclo que passa por todos os vértices de comprimento menor ou igual a K?
<u>QUESTÃO 5:</u> Dado um grafo G e dois vértices <u>s</u> e <u>t</u> de G possui um caminho de <u>s</u> a <u>t</u>?

6.2)

a) Problema de minimização.
b) Grafo G completo com pesos positivos nas arestas.
c) Dada uma instância, a solução é uma permutação de vértices.
d) Soma dos pesos das arestas entre os vértices adjacentes na permutação.
e) É o ciclo cujo custo da solução é mínimo.
f) $\alpha = 5/4$ e $Opt(x) = 100 \rightarrow C(x) \leq Opt(x) . \alpha \rightarrow C(x) \leq 100 . 5/4 \rightarrow C(x) \leq 125$

O pior caso é C(x) = 125. Isso ocorre pois, para um problema de minimização, mesmo que não se chegue à solução ótima, pelo menos se limita um valor superior, para evitar que o resultado seja muito ruim, ou longe demais de um valor plausível.

....

Exercício 7 [8.4 Sipser] Mostre que a classe de linguagens ***PSPACE*** é fechada sob as operações de:

RESPOSTA =

Sejam L_1 e L_2 linguagens, tais que existem M.T.'s M_1 e M_2 que decidem L_1 e L_2, respectivamente, em espaço polinomial. Então, L_1,L_2 Є PSPACE. Sejam f(n) e g(n) funções de complexidade de espaço para as M.T.'s M_1 e M_2, respectivamente.

a. União

Vamos construir uma M.T. M_3 que decide a linguagem $L_3 = L_1 \cup L_2$.

M_3 = "Com entrada w:
1. Simule M_1 com entrada w.
2. Simule M_2 com entrada w.
3. Se M_1 ou M_2 aceitam, ACEITE.
4. Senão, REJEITE."

A complexidade de espaço de M_3 é f(n) + g(n), que é polinomial. Então, M_3 decide L_3 em espaço polinomial. Logo, $L_3 \in$ PSPACE e a classe de linguagens PSPACE é fechada sob a operação de união.

■

b. Intersecção

Vamos construir uma M.T. M_4 que decide a linguagem $L_4 = L_1 \cap L_2$.

M_4 = "Com entrada w:
1. Simule M_1 com entrada w.
2. Simule M_2 com entrada w.
3. Se M_1 e M_2 aceitam, ACEITE.
4. Senão, REJEITE."

A complexidade de espaço de M_4 é f(n) + g(n), que é polinomial. Então, M_4 decide L_4 em espaço polinomial. Logo, $L_4 \in$ PSPACE e a classe de linguagens PSPACE é fechada sob a operação de interseção.

■

c. Concatenação

Vamos construir uma M.T. M_5 que decide a linguagem $L_5 = L_1 \bullet L_2$.

M_5 = "Com entrada $w = w_1w_2w_3...w_n$, onde w_i é cada um dos caracteres de w:
1. Para i de 0 até n, faça:

1.1 Simule M_1 com entrada $w_1w_2...w_i$.

1.2 Simule M_2 com entrada $w_{i+1}...w_n$.

1.3 Se M_1 e M_2 aceitam, ACEITE.

2. Senão, REJEITE."

A complexidade de espaço de M_5 é (n + 1)*[f(n) + g(n)], que é polinomial. Então, M_5 decide L_5 em espaço polinomial. Logo, $L_5 \in$ PSPACE e a classe de linguagens PSPACE é fechada sob a operação de concatenação.

■

d. Estrela

Vamos construir uma M.T. M_6 que decide a linguagem $L_6 = L_1{}^*$.

M_6 = "Com entrada $w = w_1w_2w_3...w_n$, onde w_i é cada um dos caracteres de w:
1. Se $w = \varepsilon$, ACEITE.
2. Senão, para i de 0 até n faça:

2.1 Simule M_1 com entrada $w_1w_2...w_i$.

2.2 Simule M_6 com entrada $w_{i+1}...w_n$.

2.3 Se M_1 e M_6 aceitam, ACEITE.
3. REJEITE."

A complexidade de espaço de M_6 é n*f(n), que é polinomial. Então, M_6 decide L_6 em espaço polinomial. Logo, L_6 Є PSPACE e a classe de linguagens PSPACE é fechada sob a operação estrela.

■

e. Complementação

Vamos construir uma M.T. M_7 que decide a linguagem $L_7 = \neg L_1$ (complemento ou negação de L1).

M_7 = "Com entrada w:
1. Simule M_1 com entrada w.
2. Se M_1 aceita, REJEITE.
   Senão, ACEITE."

A complexidade de espaço de M_7 é f(n), que é polinomial. Então, M_7 decide L_7 em espaço polinomial. Logo, L_7 Є PSPACE e a classe de linguagens PSPACE é fechada sob a operação complemento.

■

....

Exercício 8 **(8.1)** Prove que se um problema APX-Completo tem um esquema de aproximação polinomial, então PTAS = APX (ou seja, todos os problemas em APX têm um esquema de aproximação polinomial). **(8.2)** Prove que $\leq_{\mathbf{PTAS}}$ é uma relação transitiva.

RESPOSTA =

8.1)

Seja P um problema de otimização APX-Completo. Logo, P Є APX e para todo A Є APX, A $\leq_{PTAS}$ P, ou seja, A tem uma redução PTAS para P.

Como A $\leq_{PTAS}$ P, existem algoritmos polinomiais M_1 e M_2 tais que para cada instância de A, M_1 produz uma instância $M_1(x)$ de P, e para cada solução s_P de $F(M_1(x))$, M_2 produz uma solução s_A de F(x).

Se P tem um esquema de aproximação polinomial, então existe um algoritmo polinomial que obtém uma solução de P.

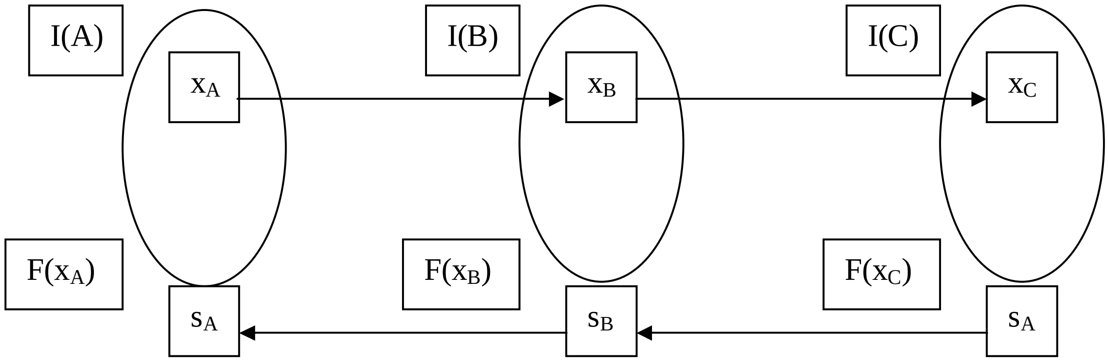
Logo, como A $\leq_{PTAS}$ P e P tem uma PTAS, então A tem uma PTAS também (transformação completamente polinomial). E como A é qualquer problema em APX, então todo problema em APX possui um esquema de aproximação polinomial.

Assim, para todo A Є APX, A tem um esquema de aproximação polinomial. Logo, APX = PTAS.

■

8.2)

Sejam A, B e C problemas tais que A $\leq_{PTAS}$ B e B $\leq_{PTAS}$ C. Logo, vale o seguinte diagrama:



- Como A $\leq_{PTAS}$ B, existem algoritmos polinomiais M_1 e M_2, tais que:

  $x_A \in I(A) \rightarrow x_B = M_1(x_A) \in I(B)$
  $s_B \in F(M_1(x_A)) = F(x_B) \rightarrow M_2(s_B) \in F(x_A)$

- Como B $\leq_{PTAS}$ C, existem algoritmos polinomiais M_1' e M_2', tais que:

  $x_B \in I(B) \rightarrow x_C = M_1'(x_B) \in I(C)$
  $s_C \in F(M_1'(x_B)) = F(x_C) \rightarrow M_2'(s_C) \in F(x_B)$

- Como $x_B = M_1(x_A)$ e $s_B = M_2'(s_C)$:

  $x_A \in I(A) \rightarrow x_C = M_1'(x_B) = M_1'(M_1(x_A)) \in I(C)$
  $s_C \in F(x_C) = F(M_1'(x_B)) = F(M_1'(M_1(x_A))) \rightarrow M_2(s_B) = M_2(M_2'(s_C)) \in F(x_A)$

Logo, como a composição de polinômios também é um polinômio, A $\leq_{PTAS}$ C. Isso prova que $\leq_{PTAS}$ é uma relação transitiva.

■

....