Nome: Patricia da Silva Rodrigues

Nota da Prova 2: 8.2

Q1 (valor 4.0): nota 2.2

- (penalidade: -0.1) Item (c): justificativa se refere ao 3-SAT
  O Teorema de Cook e Levin provou que o problema SAT é NP-completo.
- (penalidade: -0.5) Item (b): resposta incorreta ou inexistente
  A afirmação é verdadeira.
- (penalidade: -0.5) Item (d): resposta incorreta ou inexistente
  A afirmação (d) é falsa. Como visto em aula, se um grafo bipartido não tem um emparelhamento perfeito, existe um certificado sucinto deste fato. A saber, um conjunto de vértices de um dos lados da bipartição que tem menos vizinhos do outro lado.
- (penalidade: -0.5) Item (e): resposta incorreta ou inexistente
  A afirmação (e) é verdadeira.
- (penalidade: -0.0) Item (f): resposta incorreta mas justificativa correta
  Sua justificativa implica que a afirmação não tem resposta conhecida.
  Obs: Justificativa não exatamente certa, mas na direção certa.
- (penalidade: -0.2) Item (g): justificativa incompleta
  A afirmação (g) é falsa, pois NP não é a classe dos problemas "não polinomials" e também porque o problema de encontrar um emparelhamento perfeito não é um problema de decisão, condição necessária para que um problema esteja em NP.

Q2 (valor 3.0): nota 3.0

Q3 (valor 3.0): nota 0.0

- (penalidade: -3.0) Questão não escolhida
  Atenção: isso não se trata de penalidade! Mas o sistema que uso para gerar os relatórios descreve qualquer coisa como penalidade de forma "hard-coded".

Q4 (valor 3.0): nota 3.0

MAC0338 Análise de Algoritmos: Prova II

NOME COMPLETO: Patricia S. Rodrigues

NUSP: 11315590

ASSINATURA: Patricia S. Rodrigues

ATENÇÃO

O exercício 1 é obrigatório. Dentre os exercícios 2, 3 e 4, você deve escolher **EXATAMENTE DOIS** para resolver. **CIRCULE** os números dos dois exercícios escolhidos; somente estes serão corrigidos.

Siga as instruções abaixo **RIGOROSAMENTE**. Qualquer desvio acarretará na penalização em 1,0 ponto.

1. Escreva todas as suas iniciais no canto inferior direito de **TODAS** as folhas de prova.
2. Mantenha suas respostas **dentro das bordas** de cada espaço reservado para elas. (Sua prova será escaneada; conteúdos fora das bordas serão perdidos.)
3. A prova pode ser escrita a lápis ou caneta. Sua escrita deve ser legível e firme para ser visível para o scanner.

Iniciais: P.S.R

Exercício 1. Seguem abaixo várias afirmações referentes a complexidade computacional. Para cada uma, escreva se ela é verdadeira, falsa, ou sem resposta conhecida. No caso de uma afirmação falsa, você deve especificar **todas** as partes da afirmação que são falsas e explicar o que há de errado em cada parte; pode haver mais de uma parte falsa em uma afirmação. [4,0 pontos]

✓ (a) Não existe um algoritmo polinomial para o problema 3-SAT se e somente se $P \neq NP$.

(b) Se Π e Π' são problemas de decisão tais que $\Pi \leq_P \Pi'$ e Π' está em P, então Π também está em P.

Foi o SAT ← ✗ (c) O Teorema de Cook e Levin estabeleceu o problema de encontrar um ciclo hamiltoniano num grafo como o primeiro problema conhecidamente NP-completo.

(d) O seguinte problema de decisão não está em coNP: dado um grafo bipartido, existe um emparelhamento perfeito?

(e) O problema de decidir se um número inteiro $n > 1$ é composto está na classe NP, pois um divisor d de n tal que $1 < d < n$ pode ser utilizado como certificado polinomial da resposta SIM.

✗ (f) Todo problema de decisão que admite verificadores polinomiais para as respostas SIM e NÃO pode ser resolvido em tempo polinomial.

NP ≠ ñ pol ✗ (g) A classe NP dos problemas não polinomiais inclui o problema de encontrar um emparelhamento perfeito em um grafo bipartido.

✗ (h) Todo problema NP-difícil é NP-completo.

RESPOSTA

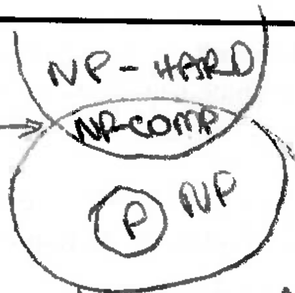
(C) Na verdade o Teorema estabeleceu (na década de 70) que o problema 3-SAT era NP-completo. Como sabemos, se um problema $\Pi \geq_P \Pi'$, e $\Pi \in$ NP-completo, então Π' é NP-completo também. Nesse caso, é possível reduzir 3-SAT para HC e portanto assim podemos constatar que HC também é NP-completo, mas o primeiro problema constatado foi o 3-SAT.

(F) A f está incorreta, pois há problemas NP que também são problemas que retornam SIM ou NÃO e que através de um certificado de tamanho polinomial, usam verificadores em tempo polinomial para retornar SIM ou NÃO para um dado certificado, mas alguns não são capazes (até onde sabemos) de resolver um problema em tempo polinomial (exemplo: NP-comple-

 Iniciais: PSR

RESPOSTA

TO). O diagrama ilustra isso



Portanto um contra-exemplo da afirmação são os NP-completos, pois possuem verificador polinomial para sim, e NÃO, são problemas de decisão, mas não podem ser resolvido em tempo polinomial (até o momento não sabemos se P=NP.).

G) NP é <u>diferente</u> de não polinomial. O que define um problema NP é possuem verificadores que, através de um certificado de tamanho polinomial, retornam Sim ou NÃO em tempo polinomial.

H) A afirmação está incorreta, pois são coisas diferentes. Todo NP-completo é NP-HARD, mas nem todo NP-HARD é NP-completo. A definição de NP-HARD é dado um problema X, se qualquer problema Y ∈ NP puder ser reduzido a X, então X é NP-HARD. Isso é equivalente a dizer que um problema NP-HARD é tão difícil quanto o problema NP mais difícil.
Já a definição de NP-completo é primeiro que ele é NP, ou seja possui verificador que retorna sim ou não em tempo polinomial para um dado certificado de tamanho polinomial, além disso para mostrarmos que ele é completo, reduzimos um NP-completo já conhecido a ele.

 Iniciais: PSR

Exercício 2. Descreva um algoritmo eficiente para o seguinte problema. Dado um vetor $A[1\,..\,n]$ com n números racionais, determina um menor conjunto de intervalos fechados de comprimento 1 (ou seja, subconjuntos de $\mathbb{R}$ da forma $[x, x+1]$ para algum $x \in \mathbb{R}$) que contém todos os números de $A[1\,..\,n]$. [3,0 pontos]

Você deve escrever o pseudocódigo do seu algoritmo e descrever resumidamente seu funcionamento e consumo de tempo. Você também deve provar a corretude do seu algoritmo.

RESPOSTA

```
ESCALONA (B)
ordena de modo que A_1 ≥ A_2 ≥ ... ≥ A_n e k = 1
PARA i de 1 até n FAÇA:
    se |A_k - A_i| = 1 FAÇA:
        S = S ∪ {A_i}
        k = i
DEVOLVA S.
```

Prova:

Seja T_k o conjunto de intervalos imediatamente após a inclusão do elemento x_k em S_k. Seja x_y o próximo elemento que atende ao requisito, ou seja $|x_k - x_y| = 1$, portanto ele deveria ser o próximo elemento a ser incluído em S_k e que $x_k > x_y$.

Vamos supor que x_y não faz parte de nenhuma solução ótima e retirado de S_k. Desta forma estamos supondo que $\exists$ um S'_k que também seria uma solução ótima tal que $S'_k = S \setminus \{x_y\}$ e ela não inclui o próximo elemento que atende ao requisito

Iniciais: P. S. R

RESPOSTA

Seja esse elemento x_w portanto $|x_R - x_w| = 1$. Como ordenamos o vetor de modo decrescente, sabemos que $x_y > x_w$.

Se $|x_R - x_w| = 1$ e $|x_R - x_y| = 1$ e $x_R > x_w$ e $x_R > x_y$ logo temos uma contradição, pois $x_w = x_y$, então a escolha do escalona para o conjunto T_R é ótima.

Consumo de tempo

ordenar: $O(n \log n) \rightarrow$ merge sort.

o loop (linha 2) será $O(n)$

$3-6 = 1 \cdot n \rightarrow$ loop

logo o tempo total $T(n)$ será $O(n \log n)$ dominado pela ordenação

Iniciais: P.S.R.

Exercício 3. Seja $G = (V, E)$ um grafo conexo em que cada aresta $e \in E$ tem um custo c_e. [3,0 pontos] Seja T uma MST de (G, c) e seja $f \in T$. Prove que existe um corte de G em que f tem custo mínimo; ou seja, prove que existe $S \subseteq V$ tal que $f \in \delta(S)$ e f tem custo mínimo dentre as arestas de $\delta(S)$.

(Para mostrar existência de tal $S \subseteq V$, você deve especificar precisamente como construir tal conjunto.)

Dica: Da mesma forma que, para qualquer aresta e fora de T, existe um único circuito em $T + e$, temos que, para qualquer aresta e' de T, existe um único corte de G disjunto de $T - e'$.

RESPOSTA

Iniciais: P.S.R

RESPOSTA

Exercício 4. Uma sequência de n operações é executada em uma estrutura de dados. Para cada i em $1, 2, \ldots, n$, a i-ésima operação custa i se i é uma potência de 2, e 1 caso contrário. Determine o tempo amortizado por operação; justifique a sua resposta. [3,0 pontos]

RESPOSTA

$$\text{custo} = \begin{cases} i, & \text{se } i \text{ é potência de } 2 \\ 1, & \text{c.c} \end{cases}$$

Tempo amortizado

$$= \sum_{i=1}^{\log n} 2^i + \left(\sum_{i=1}^{n} 1\right) - \log n$$

$$\boxed{I = 2^0 + 2^1 + \dots + 2^{\log n} = 2^{\log n + 1} - 1.}$$

$$= 2^{\log_2 n+1} - 1 + n - \log n$$

$$= 2^{\log_2 n} \cdot 2 - 1 + n - \log n = 2 \cdot 2^{\log_2 n} - 1 + n - \log n$$

$$= 2 \cdot n - 1 + n - \log n = 3n - \log n - 1 = T(n) = O(n)$$

Como o termo $3n$ dominará, temos $T(n) \approx 3n$. Por operação será

$$\frac{3n}{n} = \boxed{3}$$

Justificativa: como de 1 até n termos log n termos i, podemos somar essas potências de 1 até log n. e depois somarmos os termos 1 que serão n − log n (porque log n desses n termos já somamos). Depois basta juntar tudo e teremos o custo total. Para saber por operação, divi-

Iniciais: P.S.R

RESPOSTA

dividimos esse total pelo número de elementos.
No caso, n elementos

Iniciais: P.S.R

RESPOSTA

Iniciais: P. S. R