Exame de Qualificação - 11/12/2014

Profs. Álvaro Freitas Moreira e Luciana S. Buriol

Nome:

Dicas gerais:

- Leia todas as questões antes de começar e pergunte em caso de dúvidas;
- sempre justifique a sua resposta;
- responda a cada questão, ainda que a resposta não esteja completa.

Escolha apenas quatro questões para responder, sendo duas entre as três primeiras e duas das três últimas.

1. (2,5 pts) Escreva sobre a tese de Church -Turing. Em sua resposta , você deve:
   a) escrever o enunciado da tese
   b) explicar porque é uma tese e não um teorema (ou seja, explicar por que ela não pode ser provada)
   c) escrever sobre as evidências que suportam a sua verdade (mencionar duas dessas evidências)
   d) explicar a sua importância para a Ciência da Computação, em particular explicar porque ela é importante em provas de resultados negativos, como a indecidibilidade do Problema da Parada.

2. (2,5 pts) Faça um breve ensaio discorrendo sobre os seguintes conceitos relativos à Teoria da Complexidade
   a) complexidade de tempo e um algoritmo e complexidade de tempo de um problema algorítmico
   b) complexidade de tempo polinomial e complexidade de tempo exponencial
   c) problemas tratáveis e intratáveis (com exemplos de problemas tratáveis e intratáveis, e exemplos de problemas com "status" em relação a tratabilidade desconhecido).

3. (2,5 pts) A fim de entender melhor problemas algorítmicos com respeito ao uso de recursos, pesquisadores começaram a agrupá-lo de acordo com as suas semelhanças em relação a requisitos de tempo (e espaço). Esse estudo deu origem as classes de complexidade. Faça um breve ensaio sobre as classes

   PTIME , NPTIME , NPTIME- Complete, EXPTIME

   Descreva quais são as propriedades determinantes para um problema estar dentro de cada uma dessas classes. Complemente o ensaio com diagramas de Venn que mostram a relação entre essas classes. Discutir sobre a questão P = NP.

4. (2,5 pontos) **Análise de algoritmos.** Considere o pseudocódigo abaixo. O algoritmo recebe como entrada um nó $s \in V$ e um grafo não direcionado, conexo e pesado $G = (V, A, w)$ onde V representa o conjunto de nós (enumerados de 1 a $|V| = n$), E representa o conjunto de arestas ($|E| = m$), e $w(u, v) \in R^+$ são valores atribuídos a cada aresta $(u, v) \in E$.

```
INPUT: G=(V,A,w), s ∈ V
OUTPUT: π
for each v ∈ V do
    c[v] := ∞; insert(v,Q);
end
c[s] := 0; π[s] := null;
while Q ≠ ∅ do
    u := extract-min(Q);
    for each v adjacent to u do
        if v ∈ Q and c[v] > w(u,v) then
            c[v] := w(u, v);
            π[v] := u;
            update-key(Q,v);
        end
    end
end
return (π);
```

A função `insert(s,Q)` insere node s em Q; função `extract-min(s,Q)` remove o elemento i de Q com o menor valor c_i; e função `update-key(Q,v)` reorganiza Q de acordo com c caso necessário. As funções acima têm os seguintes custos de pior caso considerando Q as seguintes estruturas de dados:

	insert	extract-min	update-key
unordered array	O(1)	O(n)	O(1)
binary heap	O(log n)	O(log n)	O(log n)
Brodal Queue	O(1)	O(log n)	O(1)

a) Qual problema o algoritmo resolve?

b) Complete a tabela abaixo com a complexidade de pior caso, brevemente detalhando cada caso:

G \ Q	matriz de adjacência	lista de adjacência	
unordered array			
binary heap			
Brodal Queue			

É o algoritmo de Prim que resolve o problema da Árvore Geradora de Custo Mínimo.

G \ Q	matriz de adjacência	lista de adjacência
unordered array	O(n^2)	O(n^2)
binary heap	O($n^2 \log n$)	O($m \log n$)
Brodal Queue	O(n^2)	O($n \log n + m$)

5. (2,5 pontos) Apresente o pseudo-código de um algoritmo de divisão-e-conquista que, dado um vetor de dimensão n, o algoritmo retorna o valor do menor elemento deste vetor. Ainda, apresente a equação de recorrência do seu algoritmo, bem como analise a complexidade de tempo do mesmo.

```
FindMinDivideConquer(V, i, f)
INPUT: vetor V, indices i e f indicando o início e fim a ser considerada do vetor if f==i then
  | return(V[i]);
end
avg := ⌊(f-i)/2⌋ ;
v1 := FindMinDivideConquer(V, i, avg) ;
v2 := FindMinDivideConquer(V, avg+1, f) ;
if v1 ≤ v2 then
  | return(v1);
  | else return(v2);
end
```

Equação de recorrência: T(n)=2T(n/2) +O(1) e a complexidade correspondente é O(n).

6. (2,5 pontos) O problema da mochila é um dos 21 problemas classificados como NP-Completos por Karp em 1972. É dado um conjunto de n itens, sendo que cada item i possui associado um peso w_i e um valor v_i. A versão de maximização do problema da mochila 0-1 consiste em encontrar o subconjunto de itens cuja soma de valor é máxima, e a soma dos pesos dos itens selecionados não ultrapassa a capacidade K da mochila. Seja x_i uma variável que indica se o item i está ou não na solução, o problema em notação matemática:

$$\textbf{min} \quad \sum_{i=1}^{i=n} x_i v_i \tag{1}$$

$$\textbf{s.a} \quad \sum_{i=1}^{i=n} x_i w_i \leq K \tag{2}$$

$$x_i \in \{0,1\} \quad for \quad i = 1, ..., n \tag{3}$$

$$\tag{4}$$

Resolva este problema via programação dinâmica. Para tanto:

a) apresente o pseudo-código de um algoritmo de PD para resolver o problema;

b) analise a complexidade de tempo e espaço do seu algoritmo.