ACH-2002 - Introdução à Análise de Algoritmos - Prova rec – 28.01.2016 – Prof. Fábio Nakano

Nome: **nusp**

Orientações

- Duração: 1h45min. Entregar a avaliação ao professor antes deste anunciar o final do tempo. Em seguida o prof. deve apontar a entrega na lista de presença. Caso o professor precise ir ao aluno recolher a avaliação será atribuída nota ZERO. Avaliações que não forem entregues receberão nota ZERO.
- Estratégia sugerida: leia todas as questões e resolva-as procurando MAXIMIZAR sua nota. Considere o tempo para ENTENDER (não só ler) o enunciado, E o que você é capaz de responder gastando pouco tempo.
- Escrever seu nome e número USP em todas as folhas. Quando houver local indicado, usá-lo.
- Preencher seu nome e número USP nesta folha.
- Entregar esta folha junto com as folhas de resposta.
- Colocar as folhas de resposta e esta uma dentro da outra de forma que formem um único bloco.
- É proibida qualquer consulta, por exemplo (não limitado a) colegas, livros, anotações feitas antes da avaliação e anotações de colegas.
- Mostrar o encadeamento lógico das idéias e conceitos é essencial nas respostas.
- As questões sobre a linguagem de programação DEVEM ser respondidas USANDO-A. Caso não haja menção sobre linguagem a usar, é permitido usar qualquer uma, inclusive pseudo-código.
- Escrever a lápis ou tinta, como preferir
- Indicar claramente a que questão refere-se a resolução
- Apresentar a resolução na ordem que preferir. (o enunciado deve ser lido sequencialmenete ;-)
- A avaliação contém 11 pontos. Você pode escolher não responder alguma questão ou ítem pois serão considerados até 10 pontos.

---

1. **(1pt)** Explique qual a utilidade da análise de algoritmos.

2. **(2pt)** Dado o código-fonte recursivo, qual a sequência usual para chegar a sua complexidade de tempo?

3. **(1pt)** No caso ideal de tabelas de endereçamento **direto**, dê as operações de dicionário e sua complexidade de tempo em notação assintótica.

4. O código abaixo refere-se a uma tabela de endereçamento aberto.

```
1  class P2EnderecamentoAberto {
2     final int m=10;
3     String[] Memo= new String[m];
4     int h1 (String chave) {
5        // as chaves tem dois ou mais caracteres;
6        return chave.charAt(0)+17*chave.charAt(1);
7     }
8     boolean sonda (String chave, int tentativa) {
9        if (Memo[(h1(chave)+tentativa)%m]==null) {
10          return true;
11       }
12       return false;
13    }
14    boolean insere (String chave) {
15       int i=0;
16       while ((!sonda (chave, i))&&(i<m)) {
17          i++;
18       }
19       if (i>=m) return false; // tabela cheia;
20       Memo[(h1(chave)+i)%m]=chave;
21       return true;
22    }
23    void print () {
24       System.out.println ("Memo=[");
25       for (int i=0;i<Memo.length;i++)
26          System.out.println (Memo[i]);
27       System.out.println ("]");
28    }
29    public static void main (String[] args) {
30       P2EnderecamentoAberto p=new P2EnderecamentoAberto();
31       String nomesParaInserir[]={"cimento", "giz", "cintila",
32          "jamanta", "cervo", "jaba"};
33       for (int i=0;i<nomesParaInserir.length;i++) {
34          p.insere(nomesParaInserir[i]);
35       }
36       p.print();
37    }
38 }
39
```

(a) **(2pt)** Faça o teste de mesa para o exemplo dado, contando quantas sondagens são necessárias a cada inserção. **Deixe as contas indicadas**. Use o espaço abaixo para mostrar o estado final da memória relativa à tabela e a tabela para o número de sondagens. Considere os caracteres em ordem alfabética com código $'a' = 0, 'b' = 1, 'c' = 2, 'd' = 3, ...$

Memo.....		Sondagens.....	
0		cimento	
1		giz	
2		cintila	
3		jamanta	
4		cervo	
5		jaba	
6			
7			
8			
9			

5. Para o código abaixo:

```
1 class p1 {
2   int p (int a, int b) {
3     if (b==0) return 1;
4     int c= p(a, b/2) ;
5     if ((b%2)==0) return c * c;
6     return c * c * a;
7   }
8   public static void main (String[]args){
9     p1 calc = new p1();
10     System.out.println (calc.p(4,0));
11    System.out.println (calc.p(4,3));
12    System.out.println (calc.p(4,4));
13  }
14 }
```

   (a) **(1pt)** Extraia do método p a recorrência que descreve sua complexidade de tempo ;

   (b) **(2pt)** Apresente e demonstre a que classe de complexidade ($O(2^n)$, $O(n*lg(n))$, ...) pertence a recorrência do ítem anterior aplicando o Teorema Mestre;

6. **(2pt)** Explique por quê no atual paradigma de computação não é possível algoritmo de ordenação por comparação com complexidade de tempo menor que $\Theta(n * lg(n))$

Teorema Mestre:

Dada a recorrência $T(n) = aT(\frac{n}{b}) + f(n)$

caso 2: Se $f(n) \in \Theta(n^{log_b(a)}) \Rightarrow T(n) \in \Theta(n^{log_b(a)} * lg(n))$

caso 1: Se $f(n) \in O(n^{log_b(a)-\epsilon}) \Rightarrow T(n) \in \Theta(n^{log_b(a)})$

caso 3: Se $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ ...
e $a * f(\frac{n}{b}) \leq c * f(n); 0 < c < 1 \Rightarrow T(n) \in \Theta(f(n))$

Aproximação de Stirling:

$$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + O(\frac{1}{n}))$$

Escrito usando Windows8.1, Notepad++, TeXworks (Latex) e j2html código testado usando Oracle JDK1.8.0-25