MAC 338 – Anáise de Algoritmos
Primeiro Semestre de 2008
Segunda Prova – 5 de junho

Nome do aluno: _____ **Curso**: _____

Assinatura: _____

No. USP: _____ **Professor**: _____

Instruções

1. Não destaque as folhas deste caderno.
2. A prova pode ser feita a lápis.
3. A legibilidade também faz parte da nota!
4. A prova consta de 4 questões, mas você deve resolver apenas 3 delas, a sua escolha. (Sua prova valerá 10,5.) Verifique antes de começar a prova se o seu caderno de questões está completo.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é necessário apagar rascunhos no caderno de questão mas especifique qual é a resposta e qual é o rascunho.
7. A prova é sem consulta.

Não escrever nesta parte da folha

Questão	Nota	Observação
1		
2		
3		
4		
Total		

Boa Sorte !

1. [2,0 + 1,5 pontos]
   Dado um algoritmo linear "caixa-preta" que devolve a posição de uma mediana de um vetor, dê um algoritmo simples, linear, que resolve o problema do k-ésimo mínimo. Assuma que o protótipo deste algoritmo é ou **Mediana** (v, n) ou **Mediana** (v, p, r). Ao apresentar o seu algoritmo, diga o que ele faz em função dos parâmetros, justifique porque seu algoritmo produz a resposta correta, e analise cuidadosamente o tempo consumido por ele, mostrando que de fato ele é linear.

2. [2,5 + 1,0 pontos]
Escreva uma versão recursiva com memoização do algoritmo descrito em aula e apresentado abaixo, que determina o custo de uma árvore de busca binária ótima. Quanto tempo consome a sua versão memoizada do algoritmo para calcular $c(1, n)$?

```
ABB-ÓTIMA (a, n)
 1  s[0] ← 0
 2  para i ← 1 até n faça
 3     s[i] ← s[i−1] + a[i]
 4  para i ← 1 até n+1 faça
 5     c[i][i−1] ← 0
 6  para ℓ ← 1 até n faça
 7     para i ← 1 até n−ℓ+1 faça
 8        j ← i+ℓ−1
 9        c[i][j] ← c[i+1][j]
 9        para k ← i+1 até j faça
10           se c[i][k−1] + c[k+1][j] < c[i][j]
11              então c[i][j] ← c[i][k−1] + c[k+1][j]
12        c[i][j] ← c[i][j] + s[j] − s[i−1]
13  devolva c[1, n]
```

3. [1,5 + 2,0 pontos]
Dado n e uma cadeia de n caracteres $s[1\,..\,n]$ que você acredita ser um texto corrompido, em que toda a pontuação foi removida (de modo que pareça com alguma coisa assim... "eraumavezumgatoxadrez..."). Você deseja reconstruir o documento, usando um dicionário, que está disponível na forma de uma função booleana `dict(.)`: para cada cadeia de caracteres w,

$$\texttt{dict(.)} \;=\; \begin{cases} \texttt{true} & \text{se } w \text{ é uma palavra válida} \\ \texttt{false} & \text{caso contrário.} \end{cases}$$

Escreva um algoritmo de programação dinâmica que, dado n e uma cadeia de caracteres $s[1\,..\,n]$, determina se s pode ser reconstituída como uma seqüência de palavras válidas. O seu algoritmo deve consumir tempo $O(n^2)$. Justifique porque ele funciona (por exemplo explicando a validade da recorrência de onde ele foi derivado) e porque o seu consumo de tempo é $O(n^2)$.

4. [1,0 + 2,5 pontos]
(Esta questão é um exercício de uma das listas reescrito.) Seja $x_1, x_2, \ldots, x_n$ uma seqüência de números, onde n é par. Um *pareamento* de $x_1, x_2, \ldots, x_n$ é uma partição do (multi)conjunto $\{x_1, x_2, \ldots, x_n\}$ em pares. Se P é um pareamento de $x_1, x_2, \ldots, x_n$, então a *altura* de P é o valor $\max x_i + x_j : \{x_i, x_j\}$ é um par de P. Um pareamento de $x_1, x_2, \ldots, x_n$ é ótimo se tem altura mínima.

Escreva um algoritmo que, dado n e os números $x_1, x_2, \ldots, x_n$, encontra um pareamento ótimo de $x_1, x_2, \ldots, x_n$. Seu algoritmo deve consumir tempo $O(n \lg n)$. Justifique que ele de fato produz uma resposta correta, e que o seu consumo de tempo é de fato $O(n \lg n)$.