Rio de Janeiro, 2 de Abril de 2012.
PROVA 1 DE ANÁLISE DE ALGORITMOS
PROFESSOR: EDUARDO SANY LABER
DURAÇÃO: 1:50h

1 (2.0pt). Considere o problema $\mathcal{Q}$ definido da seguinte forma: *Entrada*: Inteiro $N \geq 3$; *Saída*: SIM se N é composto; NÃO, caso contrário.

a) Qual é o *tamanho da entrada* deste problema em função de N.

b) Seja $T(N)$ a complexidade de pior caso do algoritmo abaixo. Encontre uma função $f(N)$ tal que $T(N) = \Theta(f(N))$. Assuma que o teste N é múltiplo de I custa $\Theta(\log N)$.

```
I ← 2
While I * I ≤ N faça
        If  N é múltiplo de I then
                Return  SIM, N é composto
        End If
        I ← I + 1
End While
Return  NÃO, N é primo
```

2. Considere o problema $\mathcal{P}$ definido da seguinte forma: Entrada: uma palavra P com 3 caracteres e um texto T com n caracteres. Saída: SIM se **alguma permutação** da palavra P aparece no texto T e NÃO, caso contrário.

Escreva o pseudo-código de um algoritmo para resolver este problema e analise a sua complexidade de pior caso em função de n. Caso seja necessário, utilize $T[i]$ e $P[i]$ para acessar o i-ésimo caracter do padrão P e do texto T, respectivamente.

3. Considere a lista de números $(125, 24, 467, 456, 1236, 245, 8, 7091, 884, 15, 224, 321)$. Explique como o algoritmo RADIX-SORT apresentado em aula ordenaria essa lista. Em particular, mostre como a lista fica depois de cada uma das iterações do loop principal do algoritmo.

4. Para cada um dos somatórios abaixo encontre uma função $f(n)$ tal que $T(n) = \Theta(f(n))$

a) $T(n) = \sum_{i=1}^{n} n^3/3^n$

b) $T(n) = 4T(n/2) + n$ se $n > 1$; $T(1) = 1$, caso contrário.

5. (2.0pt) Seja S um conjunto de n inteiros no conjunto $\{1, ..., U\}$. Considere o seguinte procedimento

Para i=1,...,n^2
    Remova o menor elemento de S
    $x \leftarrow Rand(1, U)$ [1].
    Insira x em S.
Fim Para

a) Analise a complexidade do algoritmo acima quando S esta armazenado como um heap binário. Note que após cada inserção e remoção devemos restaurar a ordenação do heap.

b) Assuma agora que $1 \leq U \leq 5$. Como poderiamos armazenar S de modo a melhorar a complexidade do procedimento? Qual a complexidade obtida?

[1] RAND(1,U) retorna em tempo constante um inteiro aleatório entre 1 e U