IME 04-10820 - *Algoritmos e Estruturas de Dados I* – Prova 1 – 10/01/2013 - GABARITO

Questão 1. Imediata

Questão 2, – *Seqüência Simples* – prova 1

a)

	2	3								
	f f_2	r								
4 < 6	2	3	4							
	f		f_2 r							
6 < 8	2	3	4	6						
		f	f_2	r						
8 < 9	2	3	4	6	8					
		f			f_2 r					
9 < 16	2	3	4	6	8	9				
			f		f_2	r				
12 < 16	2	3	4	6	8	9	12			
				f	f_2		r			
16 < 18	2	3	4	6	8	9	12	16		
				f				f_2 r		
18 < 32	2	3	4	6	8	9	12	16	18	
					f			f_2	r	
24 < 32	2	3	4	6	8	9	12	16	18	24
						f		f_2		r

b) Algoritmo:

```
SS(n):
    Esvazia Q;   Enfila(2);   Enfila(3);   f2 ← 1;
    Para i de 1 a n-2:
        Se (3*Q[f] < 2*Q[f2]) Então
            Enfila(3*Q[f]);   Desenfila;
        Senão
            Enfila(2*Q[f2]);   f2 ← r;
        Fs;
    Fp;
Fim;
```

c) A complexidade do algoritmo é O(n), pois a instrução mais executada (dentro do loop) só é executada uma vez a cada iteração.

Questão 2, – Seqüência Simples – prova 2

a)

2				
3				

2 < 3

4				
3	6			

2									

3 < 4

4				
6	9			

2	3								

4 < 6

8				
6	9	12		

2	3	4							

6 < 8

8				
9	12	18		

2	3	4	6						

8 < 9

16				
9	12	18	24	

2	3	4	6	8					

9 < 16

16				
12	18	24	27	

2	3	4	6	8	9				

12 < 16

16				
18	24	27	36	

3	3	4	6	8	9	12			

16 < 18

32				
18	24	27	36	48

2	3	4	6	8	9	12	16		

18 < 32

32				
24	27	36	48	54

2	3	4	6	8	9	12	16	18	

24 < 32

32				
27	36	48	54	72

2	3	4	6	8	9	12	16	18	24

b) Algoritmo:

```
SS(n):
    Esvazia Q1;   Esvazia Q2;  Enfila1(2);   Enfila2(3);
    Para i de 1 a n:
        Se (Q1[f1] < Q2[f2]) Então
            Enfila1(2*Q1[f1]);   Enfila2(3*Q1[f1]);  V[i] ← Desenfila1;
        Senão
            Enfila2(3*Q2[f2]);    V[i] ← Desenfila2;
        Fs;
    Fp;
Fim;
```

c) A complexidade do algoritmo é $O(n)$, pois a instrução mais executada (dentro do loop) só é executada uma vez a cada iteração.

Questão 3 – Algoritmo da Beatriz – prova 1

```
AB(s);
        Para i de 1 a n-1;
                d ← s – V[i];
                c ← i+1;   f ← n;
                Enquanto (c ≤ f):
                        j ← ⌊(c+f)/2⌋;
                        Se (V[j] < d) Então c ← j+1
                        Senão Se (V[j] > d) Então f ← j-1
                        Senão Parar loop;
                Fe;
                Se (V[j] = d) Então Retornar "S";
        Fp;
        Retornar "N";
Fim;
```

Questão 3 – Algoritmo da Beatriz – prova 2

a) uma maneira simples de obter a complexidade é a seguinte.
A instrução mais executada é qualquer uma interna à Pesquisa Binária. A Pesquisa Binária é executada em até n-1 iterações e, cada vez que é executada, as instruções internas são executadas no máximo $(\log_2 n + 1)$ vezes. Logo, o número máximo de execuções é $(n-1)(\log_2 n + 1)$, que é menor que $(n \log_2 n + n)$. Para $n > 2$ esse valor é menor que $2n \log_2 n$. Logo a complexidcade é $O(n \log n)$.

b) Uma maneira mais exata é considerar que, na primeira iteração, o loop é executado no máximo $\log_2 (n-1)$, na segunda, $\log_2 (n-2)$, e assim sucessivamente. O total de execuções é, no máximo:
$\log_2 (n-1)+ \log_2 (n-2)+...+1 = \log_2 ((n-1)!)$. Pela aproximação de Stirling, $\log n! < (n+1)\log(n+1)-n < 2n \log n$, para $n > 10$. Portanto a complexidade é $O(n \log n)$;

Questão 4 - Lista Encadeada Autoorganizável

```
Busca(k);
        p ← cab^.prox;  ant ← cab;   pont ← Nulo;
        Enquanto (p ≠ Nulo):
                Se (p^.c = k) Então
                        pont ← p;  p ← Nulo;
                Senão
                        ant ← p;  p ← p^.prox;
        Fe;
        Se (pont ≠ Nulo) Então
                ant^.prox ← pont^.prox;
                pont^.prox ← cab^.prox;
                cab^.prox ← pont;
        Fs;
Fim;
```