Prova - Alair 3ª

Questão 1 - Sejam s e t sequências de fragmentos e sejam f e g fragmentos quaisquer. Sejam dadas funções w definidas sobre pares...

a) $\varphi(\lambda, s) = \varphi(t, \lambda) = 0$

b) $\varphi(sf, tg) = \max \begin{cases} \varphi(s,t) + w(f,g) \\ \varphi(sf, t) \\ \varphi(s, tg) \\ B \end{cases}$

$B = \max\{\varphi(s',t') + \alpha(s'', t'') \mid sf = s's'',\ tg = t't''\}$

Usando programação dinâmica, escreva em pseudo código uma função que recebe as sequências s e t de comprimentos n e m. Em sua estrutura de dados, pode utilizar vetores e matrizes indexados por strings (s[i..j] denota substring de s com as letras das posições de i a j)

Tradução da recorrência

a) $\varphi(s[1..0], t[1..j]) = 0$

$\varphi(s[1..i], t[1..0]) = 0$

b) $\varphi(s[1..i], t[1..j]) = \max \begin{cases} \varphi(s[1..i-1], t[1..j-1]) + w(s[i], t[j]) \\ \varphi(s[1..i], t[1..j-1]) \\ \varphi(s[1..i-1], t[1..j]) \\ \max\{\varphi(s[1..a], t[1..b]) + \alpha(s[a+1..i], t[b+1..j])\} \\ 0 \le a \le i \\ 0 \le b < j \end{cases}$

```
FUNCAO(s, n, t, m)
  Para j ← 1 até m
      φ[s[1..0], t[1..j]] ← 0
  Para i ← 1 até n
      φ[s[1..i], t[1..0]] ← 0
  Para i ← 1 até n
      Para j ← 1 até m
          opt4 ← 0
          opt1 ← φ[s[1..i-1], t[1..j-1]] + w(s[i], t[j])
          opt2 ← φ[s[1..i], t[1..j-1]]
          opt3 ← φ[s[1..i-1], t[1..j]]
          Para a ← 0 até i
              Para b ← 0 até j
                  opt4l ← φ[s[1..a], t[1..b]] + α(s[a+1..i], t[b+1..j])
                  se opt4 < opt4l
                      então opt4 ← opt4l
          φ[s[1..i], t[1..j]] ← max{opt1, opt2, opt3, opt4}
  devolva φ[s[1..n], t[1..m]]
```

Questão 2 - Achar o maior peso, denotado por w(i,j) de um caminho qualquer de (0,0) a (i,j) em G. Escreva um conjunto de recorrências para calcular w(i,j)

w[i,j] = maior peso alcançado no caminho de (0,0) a (i,j)

$$w[i,j] = \max \begin{cases} w[i,j-1]+1 \\ w[i-1,j-1]+1 \\ w[j-1,j]+1 \end{cases} \text{ para } i \neq 0 \text{ e } j \neq 0$$

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8
5	5	6	7	8	9

$w[0,0] = 0$

$w[0,j] = w[0,j-1]+1$ para $j > 0$

$w[i,j] = w[i-1,0]+1$ para $i > 0$

Subestrutura ótima

Teorema: Seja K uma solução ótima para a instância (n,m) com peso k

1- Se $(n-1,m) \in K$ então implica que $K_{n-1,m}$ é solução ótima para a instância (n-1,m) com k-1 de peso

2- Se $(n,m-1) \in K$ então implica que $K_{n,m-1}$ é solução ótima para a instância (n,m-1) com k-1 de peso

3- Se $(n-1,m-1) \in K$ então implica que $K_{n-1,m-1}$ é solução

ótima para a instancia $(n-1, m-1)$ com $K-1$ de peso

Prova

1- Suponha que $K^*_{n-1,m}$ não seja solução ótima para $(n-1, m)$ então existe uma outra solução para $(n-1, m)$ com $x^* > x-1$ de peso, com elemos que ao ser adicionado o peso da laminha de $(n-1, m)$ para (n, m) a solução ótima será melhor que K com peso maior que x, uma contradição.

2- Suponha que $K^*_{n,m-1}$ não seja solução ótima para $(n, m-1)$ então existe uma outra solução para $(n, m-1)$ com $K^* > x-1$ de peso, dessa forma temos que ao ser adicionado o peso da laminha de $(n, m-1)$ a (n, m) a solução ótima será melhor que K com peso maior que x, uma contradição.

3- Simétrico aos anteriores

```
MAIORCAMINHO(G, n, m, i, j)
  se i > m ou j > m
     então devolva -1
  W[0,0] ← 0
  Para x ← 1 até n
       W[x,0] ← W[x-1,0] + 1
  Para y ← 1 até m
       W[0,y] ← W[0,y-1] + 1
  Para x ← 1 até i
       Para y ← 1 até j
            opt1 ← W[x,y-1] + 1
            opt2 ← W[x-1,y] + 1
            opt3 ← W[x-1,y-1] + 1
            W[x,y] ← max{opt1, opt2, opt3}
```

devolva $W[i,j]$

Tamanho da Instância (n,m)

No pior caso quando $i = n$ e $j = m$ temos um consumo de $O(n.m)$

O Algoritmo devolve -1 caso i, j não sejam válidos ou o maior caminho de $(0,0)$ a (i,j) se i, j válidos

Questão 3 -

$|A_i| = 2^i$ $\quad$ nº de elementos nos K vetores $= \sum_{i=0}^{K-1} n_i 2^i = n$

$|A_{K-1}| = 2^{K-1}$

a)

$n_0 2^0 + n_1 2^1 + n_2 2^2$

$1 + 2 + 4 + \ldots = n$

```
BUSCA(A, K, n, x)
1  Para i ← 0 até K-1
2      se n_i = 1
3          então se BUSCABIN(A_i, 0, 2^i - 1, x) ≠ -1   BUSCABIN
4                então devolva 1
5  devolva -1

BUSCABIN(A, p, r, x)
1  se p = r + 1
2     então devolva r
3     senão q ← ⌊(p+r)/2⌋
4           se x ≤ A[q]
5              então devolva BUSCABIN(A, p, q-1, x)
6              senão devolva BUSCABIN(A, q+1, r, x)
```

BUSCABIN tem consumo $\log 2^i$

BUSCA tem no máximo consumo linha 1: $O(K)$ e linha 3 $O(K).O(\log 2^K) = O(K.K) = O(K^2)$ $\quad$ $= O((\log(n-1)^2)^2) = O(\log^2 n)$

como $K = \lfloor \log(n+1) \rfloor$

b) Página 89

Questão 4 - Tarefas (escalonamento)

```
MIN-MULTA(p, m, n)
1   p ← ordena por multa decrescentemente
2   i ← 1                          inicializa e ← ∞  multa ← 0
3   enquanto i ≤ n
4       d ← p[i]
5       v ← p[i]
6       enquanto e[d] ≠ ∞ e d > 0
7           d ← d - 1
8       se d > 0
9           então e[d] ← i
10          senão u ← n.           /* tem multa */
11          enquanto e[u] ≠ ∞ e u > v
12              u ← u - 1
13          e[u] ← i
14          multa ← multa + m[i]
```

Leva no máximo tempo $O(n^2)$ pois no pior caso, todas as tarefas tem como prazo o 1º dia então será necessário percorrer todo o vetor.

Linha
1 - nlgn
2 - O(1)
3-5 - O(n)
6-7 - O(1)
8-10 O(1)
11-13 O(n).O(n)
= total = $O(n^2)$

Correção pag 52

Questão 5 - Calcule a complexidade computacional Θ do algoritmo em função de n.

```
Potencia(x, n):
1 se n = 0
2     devolva 1
3 se n for múltiplo de 3
4     devolva Potencia(x*x*x, n/3)
5 senão
6     devolva x * POTENCIA(x, n-1)
```

$T(n) = 1$ se $n = 0$

$T(n) = \log_3 n + T(n/3) + T(n-1)$

x^n

Potencia(2,15)
(2,5) -
(2,4) -
(2,3) -
(2,1)
(2,0)
$\log_3 15 = 2$ op = 5

$1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$1\ 2\ 2\ 3\ 4\ 5\ 6\ 7\ 3$

$\sum_{i=1}^{\log_3 n} i + \sum_{i=1}^{n} i - \lfloor \lg i \rfloor - \sum_{i=1}^{\log_3 n} 3^i - i$

Potencia(2,27)
Potencia(2^3, 9)
Potencia(2^{3^3}, 3)
Potencia(2^{3^3}, 1)
(Potencia 2^{3^3}, 0) = $2^{3^3} = 2^n$
$\log_3 27 = 3$ op = 4

Potencia(2,5)
Potencia(2,4) 2*
Potencia(2^2, 3) 2*2*
Potencia(2^3, 1) $2^2 2^3$
Potencia(2^3, 0) 2^5
$\log_3 5 = 1$ op = 4

Potencia(2,10)
(2,9)
(2^3, 3)
(2^{3^3}, 1)
(2^{3^3}, 0) $2 * 2^{3^3} = 2^{10}$
$\log_3 10 = 2$ op = 4

Potencia(2,2)
(2,1)
(2,0)
$\log_3 2 = 0$ 2 op

Potencia(2,28)
(2,27) -
(2,9) <-
(2,3) -
(2,1) -
(2,0) -

i	ci
1	1
0+1	
2	2
3	2
4	3
5	4
6	5
7	6
8	7
9	3

ci = custo por operação

$$ci = \begin{cases} \log_3 i + 1 & \text{se } i \text{ múltiplo de } 3 \\ i - \lfloor \log_3 i \rfloor & \text{senão múltiplo de } 3 \\ 1 & \text{se } i = 0 \end{cases}$$

$$\sum_{i=0}^{n-1} c_i \le n + \sum_{k=0}^{\log_2 n} \left\lfloor \log_2 k + 1 \right\rfloor + \sum_{k=0}^{\log_2 n} \left[i - \lfloor \lg i \rfloor \right]$$

$$n + \sum_{k=1}^{\log_2 n} \log_2 k + \sum_{k=1}^{\log_2 n} 1 + \sum_{k=1}^{\log_2 n} k - \sum_{k=1}^{\log_2 n} \lfloor \lg k \rfloor$$

$$n + \frac{1}{2}(\log_2 n)\cdot(\log_2 n) + \log_2 n + \frac{1}{2}\log_2 n\,(1+\log_2 n) - \frac{1}{2}\log_2 n \cdot (\log_2 n)$$

$$n + \log_2 n + \frac{1}{2}\log_2 n + \frac{1}{2}\log_2^2 n$$

$$n + \frac{1}{2}\log_2^2 n + \frac{3}{2}\log_2 n$$

$$= n + \frac{1}{4} n + \frac{3}{2} n$$

$$= n + \frac{4}{2} n$$

$$= 2n$$

Acho que certo! / errado

$$\sum_{i=0}^{n-1} c_i \le \sum_{k=1}^{\log_2 2n} \left\lfloor \log_2 k + 1 \right\rfloor + \sum_{k=1}^{n} i - \log i$$

$$= \sum_{k=1}^{\log_2 n} \log_2 k + \sum_{k=1}^{\log_2 n} 1 + \sum_{k=1}^{n} i - \sum_{k=1}^{n} \log_2 i$$

$$= \frac{1}{2}\log_2 n\,(\log_2 n) + \log_2 n + n - \frac{1}{2} n (\log_2 n)$$

$$= \frac{1}{2}\log_2^2 n + \log_2 n + n - \frac{1}{2} n \log_2 n$$

$$= n \log_2 n \qquad \tfrac{1}{2} + 1 + 5 - \tfrac{1}{2}\ 5.1 \qquad 2.5$$

$$= O(n \log_2 n) \text{ para } n \text{ operações}$$

$$d(n)/n = \frac{n \log_2 n}{n} = \log_2 n \quad \text{// custo amortizado}$$

Para n múltiplo de 3

$t(n) = O(\log_3 n)$

Para não múltiplo

$t(n) = O(n - \log_3 n)$

custo total amortizado: $O(n \log_3 n)$

custo amortizado por operação $O(\log_3 n)$ //

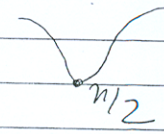
Prova - 2003/2

Questão 1 - $f(j) = (n-j)^2 + j^2$

$f'(j) = 2(n-j)\cdot(-1) + 2j = -2n + 2j + 2j = 4j - 2n$

$f'(j) = 0$

$4j - 2n = 0 \Rightarrow j = \frac{2n}{4} \Rightarrow j = n/2$



$f'(n/4) = 4\cdot\left(\frac{n}{4}\right) - 2n = -n$ negativo

$f'(n/2+1) = 4\cdot\left(\frac{n}{2}+1\right) - 2n = 2n + 4 - 2n = 4$ positivo

Para $j = n/4$

$j < n/2$ e $f'(j) < 0$ decrescente

Para $j = n/2 + 1$

$j > n/2$ e $f'(j) > 0$ crescente

Então $n/2$ é ponto mínimo, assim forma pontos próximos a 0 e a n maximizam a função