Lista de Exercícios Avaliativa 6

MC458 - 2s2020 - Tiago de Paula Alves - 187679

---

1. A solução será dada por uma lista $C = [C_1, \ldots, C_n]$, em que cada elemento é uma tupla $C_i = (c_{i,0}, \ldots, c_{i,k})$ de peças $0 \leq c_{i,j} \leq t_j$ tal que a soma dos comprimentos dos segmentos não ultrapasse o comprimento de um trilho, ou seja,

$$\sum_{j=0}^{k} 2^j \cdot c_{i,j} \leq M \qquad \text{para todo } 1 \leq i \leq n$$

Além disso, a lista como um todo deve suprir os segmentos necessários, então,

$$\sum_{i=1}^{n} c_{i,j} = t_j \qquad \text{para todo } 0 \leq j \leq k$$

Teorema (subestrutura ótima)**.** *Seja $C = [C_1, C_2, \ldots, C_n]$ uma lista de cortes com número de trilhos $n > 0$ mínimo. Então, a sublista $C/C_1 = [C_2, \ldots, C_n]$ tem número de trilhos mínimo para os segmentos não tratados em C_1.*

Demonstração. Seja $S = [S_1, \ldots, S_m]$ uma solução ótima para os segmentos $t'_i = t_i - c_{1,i}$, para $i \leq k$. Suponha, por contradição, que $m < n - 1$. Logo, podemos gerar uma solução $C' = [C_1, S_1, \ldots, S_m]$ com um número de trilhos $m + 1 < n$. Como n é ótimo para o problema original, isso é impossível. Portanto, $m \geq n - 1$ e, como C/C_1 tem $n - 1$ trilhos, ela deve ser uma solução ótima. ■

Teorema (escolha gulosa)**.** *Seja $D = (d_0, \ldots, d_k)$ uma tupla de segmentos onde*

$$d_i = \max \left\{ 0 \leq q \leq t_i \;\middle|\; q \cdot 2^i + \sum_{j=i+1}^{k} d_j \cdot 2^j \leq M \right\}$$

Se existir um elemento positivo em D, então existe uma lista de cortes ótima que contém D.

Demonstração. Suponha que D tem algum elemento positivo e seja C uma lista de cortes ótima. Se C contém D, não resta nada a provar, então suponha que C não contém D. Note que como existe um $d_i > 0$, então também existe um $t_i > 0$. Portanto, $C = [C_1, \ldots, C_n]$ é não-vazio.

Considere que $C^{(j)}$ é alguma solução ótima tal que $c^{(j)}_{1,s} = d_s$ para todo $j \leq s \leq k$. Suponha que existe algum $C^{(j+1)}$. A partir disso, vamos provar que existe $C^{(j)}$.

Caso 1: $c_{1,j}^{(j+1)} > d_j$. Por definição, d_j é o maior valor possível e, como $C_1^{(j+1)}$ tem o elementos finais iguais a D, isso implica que $c_{1,j}^{(j+1)} > t_j$ ou $\sum_{i=j}^{k} c_{1,i}^{(j+1)} \cdot 2^i > M$. Isso contradiz a suposição de que $C^{(j+1)}$ é uma solução, portanto esse caso é impossível.

Caso 2: $c_{1,j}^{(j+1)} = d_j$. Etão, temos $C^{(j)} = C^{(j+1)}$, como esperado.

Caso 3: $c_{1,j}^{(j+1)} < d_j$. Seja q_1 o maior inteiro tal que $q_1 \cdot 2^j \leq \sum_{i=0}^{j-1} c_{1,i}^{(j+1)} \cdot 2^i$. Como os comprimentos são potências de 2, então podemos juntar segmentos menores que j em q_1 grupos que somam 2^j e trocar cada um com outras tuplas de $C^{(j+1)}$ mantendo a soma dos comprimentos de cada tupla. Seja C' a solução gerada nesse processo.

Para os $q_2 = d_j - c_{1,j}^{(j+1)} - q_1$ segmentos de tamanho j restantes, temos que

$$(q_2 - 1) \cdot 2^j + \sum_{i=0}^{k} c'_{1,i} \cdot 2^i < \sum_{i=0}^{k} c'_{1,i} \cdot 2^i + q_2 \cdot 2^j - \sum_{i=0}^{j-1} c'_{1,i} \cdot 2^i \leq M$$

Então, se $q_2 = 0$, temos $C^{(j)} = C'$ como solução. Caso contrário, podemos trocar os segmentos menores por um segmento j de outra tupla e absorver outros $q_2 - 1$ segmentos j, mantendo a soma dos comprimentos de C'_1 menor ou igual a M. Assim, teremos C'' como uma nova solução, tal que $c''_{1,j} = d_j$ e, portanto, $C^{(j)} = C''$, ainda com n tuplas.

Por fim, podemos tomar $C^{(k+1)} = C$ por vacuidade. Pela demonstração anterior, temos que existe algum $C^{(0)}$, em que $c_{1,i}^{(0)} = d_i$ para todo $0 \leq i \leq k$, ou seja, $C_1^{(0)} = D$, como proposto. ■

O valor de q inteiro que maximiza a expressão é $q = \left\lfloor \frac{M}{2^i} \right\rfloor$. No entanto, pelas restrições temos que limitar q até t_i, como é feito no algoritmo abaixo.

```
SEGMENTOS-ÓTIMOS(t, k, M)
Seja D[0, ..., k] um vetor.

positivo ← FALSO
para i ← k descendo até 0
    D[i] = min(⌊M/2^i⌋, t[i])
    M ← M − D[i] · 2^i

    se D[i] > 0 então
        positivo ← VERDADEIRO

se positivo então
    retorna D
senão
    retorna NULO
```

```
MÍNIMO-TRILHOS(t, k, M)
Seja C uma lista vazia
n ← 0

faça D ← SEGMENTOS-ÓTIMOS(t, k, M)
     se D ≠ NULO então
          n ← n + 1
          C[n] ← D

          para i = 0 até k
               t[i] ← t[i] − D[i]
enquanto D ≠ NULO

retorna (C, n)
```

A função SEGMENTOS-ÓTIMOS tem complexidade $\Theta(k)$, devido ao laço interno. Além disso, podemos ver que o laço da MÍNIMO-TRILHOS executa $n+1$ vezes. Portanto, o algoritmo por completo tem tempo $\Theta(nk)$. Note que n não é entrada do algoritmo, mas pode ser limitada pela entrada.

Um solução para este problema poderia ser comprar um trilho de tamanho M para cada um dos S segmentos necessários. Essa solução é ótima em pouquíssimos casos, entrentanto, nenhuma solução ótima é pior que essa, já que a solução estaria desperdiçando trilhos completos. Por outro lado, o mínimo de peças seria o comprimento total dos segmentos dividido pelo tamanho de um trilho. Logo, os limites de n são:

$$\frac{S}{M} = \frac{1}{M}\sum_{i=0}^{k} t_i \leq \frac{1}{M}\sum_{i=0}^{k} t_i \cdot 2^i \leq n \leq \sum_{i=0}^{k} t_i = S$$

Portanto, $T(t,k) \in \Theta(Sk)$. A complexidade de espaço também é $\Theta(Sk)$, já que essa é a memória necessária para a solução.

2.

Teorema (subestrutura ótima). *Sejam k e $S = (s_1, \ldots, s_n)$ uma solução de desequilíbrio mínimo. Considere também que $p_{\min}$ e $p_{\max}$ são o mínimo e máximo de P. Então, para todo $1 \le i \le n$,*

$$s_i = \begin{cases} p_i + k & se\ |p_i - p_{\max}| > |p_i - p_{\min}| \\ p_i - k & se\ |p_i - p_{\max}| < |p_i - p_{\min}| \\ p_i \pm k & caso\ contrário \end{cases}$$

Teorema (escolha gulosa). *Seja $p_{\text{inf}} = \max\{p_i \mid p_{\max} - p_i > p_i - p_{\min}\}$ e $p_{\text{sup}} = \min\{p_i \mid p_{\max} \le p_i - p_{\min}\}$. Então, com*

$$k = \left\lceil \frac{\min\left\{p_{\text{sup}} - p_{\min}, p_{\max} - p_{\text{inf}}\right\}}{2} \right\rceil$$

Pode-se gerar uma sequência de desequilíbrio mínimo.

```
DESEQUILÍBRIO-MÍNIMO(P, n)
p-min ← P[1]
p-max ← P[1]
para i ← 2 até n
    p-min ← min(p-min, P[i])
    p-max ← max(p-max, P[i])

p-inf ← p-min
p-sup ← p-max
para i ← 1 até n
    se p-max − P[i] > P[i] − p-min então
        p-inf ← max(p-inf, P[i])
    senão
        p-sup ← min(p-sup, P[i])

max-diff ← min(p-sup − p-min, p-max − p-inf)
retorna ⌈max-diff/2⌉
```

Podemos ver que todos os laços na função executam no máximo n vezes. Então, a complexidade de tempo do algoritmo é $T(n) \in \Theta(n)$. Como não é usado nenhum armazenamento adicional, o espaço tem complexidade constante, $E(n) \in \Theta(1)$.

2. Sequência de deslocamentos $T = \langle t_1, \ldots, t_n \rangle$ tal que $t_i = -1$ ou $t_i = +1$ para todo $1 \leq i \leq n$.

Sequência $S_{P,T,k} = \langle s_1, \ldots s_n \rangle$ construída a partir da sequência P, do deslocamento T e de um natural k é tal que $s_i = p_i + t_i k$ para todo $1 \leq i \leq n$.

Valores mínimo $p_{\min}$ e máximo $p_{\max}$ de P:

$$p_{\min} = \min_{1 \leq i \leq n} p_i \quad \text{e} \quad p_{\max} = \max_{1 \leq i \leq n} p_i$$

Subsequência $S = \langle s_1, \ldots, s_m \rangle$ é tal que existem $j < l \leq n$ em que $s_i = p_j$ e $s_{i+1} = p_l$ para todo $i < m$.

Teorema. *Sejam k um inteiro positivo e T um deslocamento que geram um sequência de desequilíbrio mínimo a partir de P. Então, para toda subsequência $S = \langle p_{s_1}, \ldots, p_{s_m} \rangle$ que contém $p_{\min}$ e $p_{\max}$, o deslocamento $T' = \langle t_{s_1}, \ldots t_{s_m} \rangle$ com gera uma sequência de desequilíbrio mínimo para algum k' positivo.*

Demonstração. Se $p_{\min} = p_{\max}$, então o desequilíbrio é mínimo apenas quando todos os elementos de T são iguais. Então, T' terá todos os elentos iguais, gerando um $S' = \langle \ldots, p_{s_i} + t_{s_i} k, \ldots \rangle$, com $D(S') = 0$, isto é, mínimo. Vamos provar por indução em m que o terema vale para $p_{\min} < p_{\max}$.

Caso base: $m = 2$. Seja ■

3.

Teorema (subestrutura ótima)**.** *Seja n um inteiro com representação mínima $\langle t_0, t_1, \dots, t_k\rangle$, sendo $k > 0$. Então, $\langle t_1, \dots, t_k\rangle$ é uma representação mínima de* $\dfrac{n-t_0}{3}$.

***Demonstração*.** Considere $m = (n-t_0)/3$ e sua representação mínima $R = \langle r_0, r_1, \dots, r_{l^*}\rangle$ de tamanho l^*. Suponha agora uma representação $S = \langle s_0, \dots, s_l\rangle$ de m que não é ótima, isto é, $l > l^*$. Assim, temos que:

$$\begin{aligned} n &= 3m + t_0 \\ &= t_0 + 3\left(r_0 + r_1 \cdot 3 + \cdots + r_{l^*} \cdot 3^l\right) \\ &= t_0 + r_0 \cdot 3 + r_1 \cdot 3^2 + \cdots + r_{l^*} \cdot 3^{l^*+1} \end{aligned}$$

Então, temos a representação $R' = \langle t_0, r_0, \dots, r_{l^*}\rangle$ para n. Da mesma forma, temos que $S' = \langle t_0, s_0, \dots, s_l\rangle$ representando o mesmo número. Entretanto,

$$|R'| = |R| + 1 = l^* + 2 < l + 2 = |S| + 1 = |S'|$$

Ou seja, a representação S' não é ótima. Portanto, pela contrapositiva, para qualquer representação A de n, se ela for mínima, $A - \langle t_0\rangle$ será uma representação mínima de m. Como $\langle t_0, t_1, \dots, t_k\rangle$ é ótima, o teorema segue. ■

Teorema (escolha gulosa)**.** *Seja $\langle t_0, \dots, t_k\rangle$ uma representação mínima de n. Então, o primeiro dígito é $t_0 = (n+1) \bmod 3 - 1$.*

Demonstração. Seja $r = (n+1) \bmod 3 - 1$ e note que

$$n - t_0 = \sum_{i=0}^{k} t_i \cdot 3^i - t_0 = 3\left(\sum_{i=1}^{k} t_i \cdot 3^{i-1}\right)$$

Como $t_i \in \mathbb{Z}$ e $3^{i-1} \in \mathbb{Z}$ para $i > 0$, então $n - t_0 \equiv 0 \pmod 3$. Portanto, $n \equiv t_0 \pmod 3$.

Caso 1: $n \equiv 0 \pmod 3$. Então, $(n+1) \equiv 1 \pmod 3$ e $r = 0$. Como $-1 \not\equiv 0 \pmod 3$ e $1 \not\equiv 0 \pmod 3$, temos que $t_0 = 0 = r$.

Caso 2: $n \equiv 1 \pmod 3$. De forma similar, temos que $t_0 = 1 = r$.

Caso 3: $n \equiv 2 \pmod 3$. Note que $(n+1) \equiv 0 \pmod 3$, então $r = -1 = t_0$.

■

```
MÍNIMO-DÍGITOS(n)
se n == 0 então
    retorna (0, [0])

Seja T uma lista vazia
k ← 0
enquanto n > 0 faça
    T[k] ← ((n+1) mod 3) − 1
    n ← (n − T[k])/3
    k ← k+1

retorna (k, T)
```

Pelo laço da função MÍNIMO-DÍGITOS, temos que a complexidade de tempo é $T(n) \in \Theta(k)$. Com k dígitos, o menor número representável é com $t_i = -1$ e o maior é $t_i = +1$, para $i < k$. Para que n seja positivo e T seja a representação mínima, é necessário que $t_k = 1$. Logo,

$$3^{k-1} \leq \frac{3^k + 1}{2} = 3^k - \sum_{i=0}^{k-1} 3^i \leq n \leq \sum_{i=0}^{k} 3^k = \frac{3^{k+1} - 1}{2} \leq 3^{k+1}$$

Assim, podemos limitar k por $\log_3 n - 1 \leq k \leq \log_3 n + 1$. Portanto, $T(n) \in \Theta(\lg n)$. Devido ao armazenamento do resultado, a complexidade de espaço também é $\Theta(k) = \Theta(\lg n)$.
