Lista de soluções 1

Exercício 1 (KT 2.5)
Sabemos que existem c, n_0 tal que $f(n) \leq cg(n)$ para todo $n \geq n_0$. Em todos casos queremos encontrar c', n_0' tal que $h(f(n)) \leq c'h(g(n)$ (*) para todo $n \geq n_0'$, com $h(n) = \log_2 n$, $h(n) = 2^n$ e $h(n) = n^2$. Todas funções h são monotônicas em $\mathbb{N}$, i.e., para $x \leq y$ temos $h(x) \leq h(y)$. Logo sabemos que em todos casos $h(f(n)) \leq h(cg(n))$ para $n \geq n_0$.

a) Falso. Um contra-exemplo é $f(n) = 2$ e $g(n) = 1$: $f(n) \leq 1/2g(n)$, mas $\log_2 f(n) = 1 \not\leq c \log_2 g(n) = 0$.

b) Falso. Um contra-exemplo é $f(n) = n$ e $g(n) = 2n$: $2^n \not\leq c2^{2n} = 4^n$.

c) Verdadeiro. Temos $f(n)^2 \leq (cg(n)^2 = c^2 g(n)^2$. Logo para $n_0' = n_0$ e $c' = c^2$ a condição (*) é satisfeita.

Exercício 2 (KT 2.6)
O algoritmo executa

$$\begin{aligned}
\sum_{i\in[n]} \sum_{i<j\leq n} j - i + 1 &= \sum_{i\in[n]} \left(\sum_{j\in[n-i]} j + 1 \right) \\
&= \sum_{i\in[n]} \left(n - i + \sum_{j\in[n-i]} j \right) \\
&= \sum_{i\in[n]} \left(n - i + (n-i)(n-i+1)/2 \right) \\
&= \sum_{i\in[n]} \left(i - 1 + (i-1)i/2 \right) \\
&= 1/2 \sum_{i\in[n]} \left(i^2 + i - 2 \right) \\
&= 1/2 \left((n(n+1)(2n+1)/6 + n(n+1)/2 - 2n \right) \\
&= 1/6(n^3 + 3n^2 - 4n)
\end{aligned}$$

operações. Logo podemos escolher $f(n) = n^3$.

a) Temos $1/6(n^3 + 3n^2 - 4n) \leq n^3$, logo para $n_0 = 1$ e $c = 1$ a condição (*) é satisfeita.

b) Temos $1/6(n^3 + 3n^2 - 4n) \geq 1/6(n^3 - 4n)$ e como $4n \leq 1/4n^3$ para $n \geq 4$ temos $1/6(n^3 + 3n^2 - 4n) \geq 1/6(n^3 - 1/4n^3) = 1/8n^3$. Logo para $c = 1/8$ e $n_0 = 4$ a condição do Ω é satisfeita.

c) Podemos calcular primeiramente as somas parciais, e depois os elementos da matriz B.

```
1  P[0] := 0
2  for i ∈ [n] do
3     P[i] := P[i − 1] + A[i]
4  end for
5  { Garantia: P[i] = Σ_{1≤j≤i} A[i] }
6  for i ∈ [n] do
7     for j ∈ {i + 1, . . . , n do
8        B[i, j] := P[j] − P[i − 1]
9     end for
10 end for
```

Este algoritmo executa

$$1 + n + \sum_{i\in[n]} n - i = 1 + n + \sum_{i\in[n]} i - 1 = 1 + \sum_{i\in[n]} i = 1 + n(n+1)/2 = 1/2(n^2+n+2)$$

operações. Com

$$\frac{1/2(n^2+n+2)}{1/6(n^3+3n^2-4n)} \leq \frac{2n^2}{1/8n^3} = 16/n$$

para $n \geq 4$ temos

$$\lim_{n\to\infty} \frac{1/2(n^2+n+2)}{1/6(n^3+3n^2-4n)} = 0.$$

Exercício 3 (KT 1.1)
Falso. Um par desses não precisa existir. Considere homens h e h' e mulheres m e m' com preferências (do maior para menor) $h : m, m'$, $h' : m', m$, $m : h', h$, e $m' : h, h'$. Nenhum par saitsfaz as condições da afirmação.

Exercício 4 (KT 1.2)
Verdadeiro. Dado um par (h, m) com preferência máxima mutual, remove o par da instância e encontra um emparelhamento estável dos restantes homens e mulheres. Este emparelhamento junto com o par (h, m) é estável: Supõe que temos uma instabilidade. O par (h, m) não pode fazer parte da instabilidade. Logo, o emparelhamento sem o par (h, m) não é estável, uma contradição.

Exercício 5 (KT 1.3)
Caso a rede A possui dois shows com avaliação 3 e 6 a rede B dois shows com avaliação 5 e 2 não existe um par estável de sequenciamentos.

Exercício 6 (KT 2.7)
Um maneira para codificar isso num script sucinto é

```
1  v_1 := ''Primeira linha''
2  v_2 := ''Segunda linha''
3  ...
4  v_k := ''Última linha''
5  for i ∈ [n] do
6    for j ∈ [i] do
7      imprime linha v_j
8    end for
9  end for
```

Dado que o canto possui n palavras, ele possui $k = \Theta(\sqrt{n})$ versos. O número de carácteres no script com k versos é limitado por ck com c uma constante suficientemente grande. Logo o número de carácteres em função do número de palavras em total é $f(n) = \Theta(\sqrt{n})$. Isso é assintoticamente ótimo, porque cada script precisa pelos menos k carácteres para k versos.
Para ver a primeira afirmação, seja $w(k)$ o número de palavras com k versos. Como cada verso possui entre 1 e c palavras, temos

$$\sum_{i\in[k]}\sum_{j\in[i]} 1 \leq w(k) \leq \sum_{i\in[k]}\sum_{j\in[i]} c \iff k(k+1)/2 \leq w(k) \leq ck(k+1)/2.$$

Logo com $n = w(k)$ palavras temos

$$k(k+1)/2 \leq n \leq ck(k+1)/2 \iff k^2/2 \leq n \leq ck^2$$

que é satisfeito para $k = \Theta(\sqrt{n})$.

Exercício 7 (KT 2.8)
Testando a primeira jarra na altura h, pode quebrar, de forma que temos com $k-1$ jarras encontrar o degrau mais alto até $h-1$. Caso não quebra, temos ainda k jarras para analisar $n-h$ degraus. Portanto $f_k(n)$ satisfaz

$$f_k(n) = 1 + \min_{1 \leq h \leq n} \max\{f_{k-1}(h-1), f_k(n-h)\}.$$

Para $k = 2$, por exemplo, podemos calcular os valores exatos como

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$f_2(n)$	0	1	2	2	3	3	3	4	4	4	4	5	5	5	5	5	6

e ver que

$$f_2(n) = i, \qquad \text{caso } (i-1)i/2 < n \leq i(i+1)/2$$

com solução exata

$$f_2(n) = \left\lceil \sqrt{2n + 1/4} \right\rceil - 1.$$

(Para provar isso, temos que provar por indução que essa solução satisfaz a recorrência acima.) Para $k > 2$ a solução exata é difícil obter, portanto vamos extrapolar o comportamento do caso $k = 2$. Para testar assintoticamente menos, queremos dividir o problema em n^a testes em intervalos de n^{1-a} com um a decrescente em função de k. Escolhendo $a = 1/k$ o problema é reduzido para um intervalo de $n^{1-1/k}$ com a primeira jarra, para um intervalo de $n^{1-2/k}$ com a segunda jarra, e no geral para um intervalo de $n^{1-i/k}$ com a i-ésimo jarra, tal que a última jarra procede em intervalos de 1. O número total de testes dessa abordagem é $f_k(n) = kn^{1/k}$ no pior caso, satisfazendo o critério que para k maior, o trabalho é assintoticamente menor.