# Soluções prova algoritmos e complexidade

**Questão 1**
A abordagem simples ("força bruta")

```
for i ∈ [n]
  for j ∈ [n]
    for k ∈ [n]
      if i ≠ j e i ≠ k e k ≠ j e a_i + a_j = a_k
        return (i, j, k)
return ''Não existe tal tripla''
```

possui complexidade $O(n^3)$. Existem diversas soluções de complexidade assintoticamente menor. Por exemplo, podemos ordenar os números em ordem crescente em $O(n \log n)$ e depois testar, para todo par $(i, j)$ com $i \neq j$ se existe o número $a_i + a_j$ na sequencia. Como a sequencia é ordenada isso é possível com uma busca binária em tempo $O(\log n)$, i.e. esta abordagem possui complexidade total $O(n^2 \log n)$.

**Questão 2**
Temos as seguintes comlexidades:

| | |
|---|---|
| Emparelhamento estável | $O(n^2)$ |
| Algoritmo de Kruskal | $O(n^2 \log n)$ |
| Algoritmo de Strassen | $O(n^{\log_2 7})$ |
| Transformada rápida de Fourier | $O(n \log n)$ |
| Multiplicação de números binários | $O(n^{\log_2 3})$ |
| Algoritmo de Edmonds-Karp | $O(n^5)$ |
| Algoritmo Húngaro | $O(n^4)$ |
| Algoritmo de Prim | $O(n^2 \log n)$ |

(Observações: A complexidade do algoritmo de Kruskal supõe uma implementação do "union-find" em $O(\log n)$, a complexidade do algoritmo de Prim que as operações da fila de prioridades precisam tempo $O(\log n)$.)

**Questão 3**
O trabalho por chamada do algoritmo $R$ é $\Theta(n)$ e temos duas chamadas recursivas com a metade dos elementos, logo

$$T(n) = 2T(n/2) = \Theta(n) = O(n \log n).$$

(De fato para um $n$ impar a duas chamadas recursivas são ligeiramente desbalanceadas, mas isso não afeta a complexidade assintótica, como o método de Akra-Bazzi mostra.)

**Questão 4**
O problema pode ser resolvido usando programação dinâmica. Seja $v_{ij}$ o valor na célula $(i, j)$ e $o_{ij}$ o maior valor possível partindo da célula $(i, j)$. Os valores $o$ satisfazem

$$o_{ij} = \begin{cases} 0 & \text{caso } i > n \text{ ou } j > n \\ v_{ij} + \max\{o_{i+1,j}, o_{i,j+1}\} & \text{caso contrário} \end{cases}.$$

logo uma programação dinâmica com tempo e espaço $O(n^2)$ é uma varredura da matriz $o$ no final resolve o problema. Para obter o melhor caminho podemos armazenar – como sempre – ainda a informação sobre qual das duas possibilidades gerou o máximo na segunda linha da recorrência, e extrair este caminho no final.