Q1

Melhor caso, seria todos os valores do vetor, iguais. Sendo assim, haveria apenas 2 atribuições. O menor e maior números, no início do programa, na posição 1 do vetor. No total, seriam 2 atribuições e ‘n – 2’ comparações.

O melhor caso seria f(n) = 2 + n – 2 = n ou ‘f(n) = n’.

Pior caso, seria o vetor com valores crescentes em posições pares e valores decrescentes em posições ímpares ou vice-versa. Desta forma, haveria 1 atribuição para cada posições do vetor. No total, seriam 2 atribuições no início, ‘n – 2’ comparações e ‘n - 2’ atribuições.

O pior caso seria f(n) = 2 + (n – 2) + (n – 2) = 2 + 2(n – 2) = 2 + 2n – 4 = -2 + 2n ou ‘f(n) = 2n – 2’.

Caso médio, seria o vetor com valores em ordem crescente ou decrescente. As atribuições seriam feitas em posições alternadas. Todas as atribuições ocorreriam em posições pares ou ímpares. No total, seriam 2 atribuições no início, ‘n – 2’ comparações e ‘(n – 2) / 2’ atribuições.

O caso médio seria f(n) = 2 + n – 2 + (n – 2) / 2 = n + (n – 2) / 2 = n + n / 2 – 1 ou f(n) = ‘n + n / 2 - 1’

Q2

a) Somatório de i^4, sendo ‘i’ de 1 até ‘n’ e incrementando ‘i’. Ex: $1^4 + 2^4 + 3^4 + \ldots + n^4$.

b) Atribuição.

c) Sim. Como não existe condicionais, todas as vezes que o loop executar, a atribuição é feita.

d) n vezes. 1 atribuição no início do programa. n – 1 atribuições no loop, pois vai de 1 até n.

e) O(n).

Q3

a) Ordena os valores do vetor ‘A’ com ‘n’ valores, em ordem crescente.

b) Pior caso é quando o vetor está em ordem decrescente. Seria necessário realizar a troca em todas as iterações. Seria $O(n^2)$, por ter um laço dentro do outro.

c) Atribuição ‘k = i’ ‘n – 2’ vezes t(n – 2). Comparação ‘a[j] < a[i]’ ‘$n^2 – 1$’ vezes t($n^2 – 1$. Atribuição ‘k = j’ ‘$n^2 – 1$’ vezes t($n^2 – 1$). Função ‘troca(a[k], a[i])’ ‘n – 2’ vezes t(n – 2).

Maior custo t(n^2). Resultado = t(n^2).

Q4

n – 1 asteriscos. O loop vai de 1 (intervalo fechado) até n (intervalo aberto). Ou seja, ‘n’ não é contemplado.

Q5

Sim. Nas propriedades da notação assintótica, as constantes multiplicativas podem ser omitidas. Outras operações básicas (adição, subtração, divisão) com constantes também podem omitidas pois influenciam pouco no resultado final da função. Assim como um número infinito que é multiplicado por 2. Não vai fazer diferença. Ele continuará sendo infinito.

Q6

a)

a = 14.

Para n = 16

$7T(16/2) + 16^2 = 7T(8)+256 = T(56) + 256$

$aT(16/4) + 16^2 = aT(4)+256 = T(4a) + 256$

Para que $4a = 56 \rightarrow a = 54/4 \rightarrow a = 14$

Para n = 8

$7T(8/2) + 8^2 = 7T(4) + 64 = T(28) + 64$

$aT(8/4) + 8^2 = aT(2) + 64 = T(2a) + 64$

Para que $2a = 28 \rightarrow a = 28/2 \rightarrow a = 14$

Então, para qualquer ‘n > 0’, $T(n) = aT(n/4) + n^2$ será assintoticamente mais rápido que $T(n) = 7T(n/2) + n^2$ para ‘a = 14’.

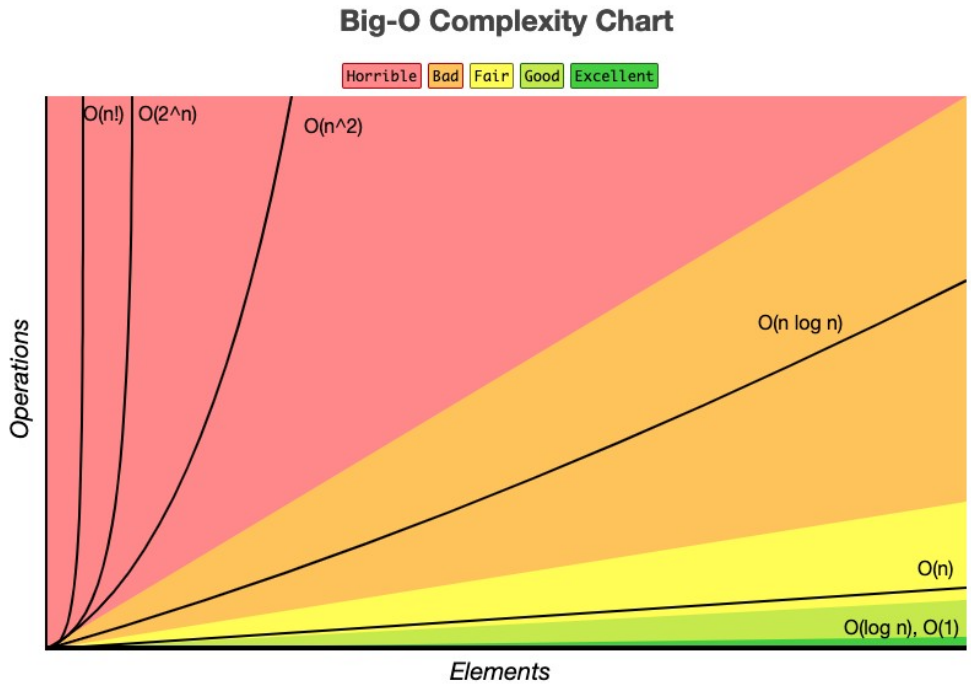
b) Sim.

a = 4; b = 2; n^2lgn é $O(n^2lgn)$; d = 2; $\log^4_2 = 2$.

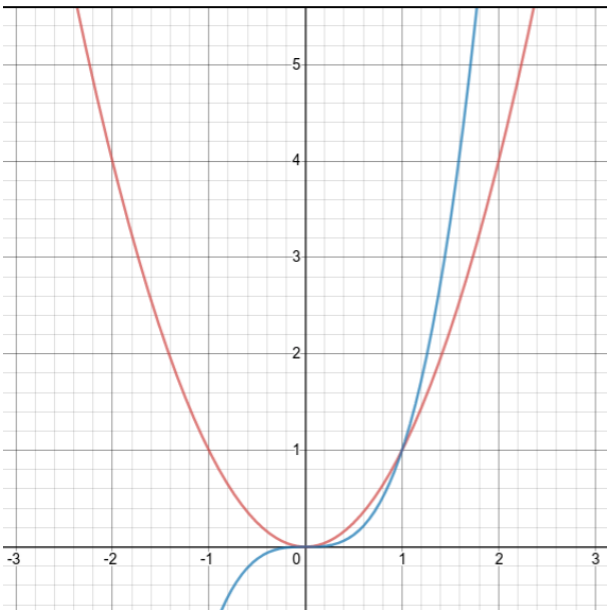
As condições são: a > 0; b > 0; d >=0; $d = \log^a_b$, para O(n^d*lgn).

Portanto, todas as condições são satisfeitas.

c)



Escolheria o segundo algoritmo. Teria muita vantagem por atuar em uma progressão linear O(n), ainda mais por ter mais chance de ocorrer. Mesmo no pior caso, a diferença entre a complexidade quadrática e cúbica é proporcionalmente pequena em relação à vantagem da progressão linear, como mostra no seguinte gráfico e comparando com o primeiro. A linha azul é a função cúbica e a vermelha é quadrática.



Q7

a = 3; b = 2; d = 1
$\log^a_b = \log^3_2 = 1{,}58 > d$
Se '$\log^3_2 > d$', então $O(n^{\log b(a)})$

Portanto, T(n) é $O(n^{\log b(a)}) = O(n^{\log 2(3)}) = O(n^{1{,}58})$

Q8

a)

T(n) = T(n – 1) + (n – 1)
[T(n – 2) + (n – 1)] + (n – 1)
[T(n – 3) + (n – 2)] + (n – 1) + (n – 1)
…
T(n) = T(n – k) + (n – (k – 1)) + (n – (k – 2)) + … + (n – 1) + (n – 1) =
T(n – k) + (n – (k – 1)) + (n – (k – 2)) + … + 2(n – 1)

Se T(1) = 0, então n – k = 0 → n = k

T(n) = T(n – n) + (n – (n – 1)) + (n – (n – 2)) + … + 2(n – 1)
T(0) + (n – n + 1) + (n – n + 2) + … + 2n – 2
T(0) + 1 + 2 + … + 2n – 2

$T(n) = T(0) + \frac{n(1 + 2n - 2)}{2}$ => PA

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

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

Portanto, T(n) é $O(n^2)$

b)

T(n) = T(n – 1) + n
[T(n – 2) + (n – 1)] + n
[T(n – 3) + (n – 2)]+ (n – 1) + n
…
T(n) = T(n – k) + T(n – k) + (n – (k – 1)) + (n – (k – 2)) + … + (n – 1) + n

Se T(1) = 1, então n – k = 1 → n = k + 1 → n – 1 = k

T(n) = T(n – n – 1) + (n – (n – 1 – 1)) + (n – (n – 1 – 2)) + … + (n – 1) + n
T(-1) + (n – n + 1 + 1) + (n – n + 1 + 2) + … + (n – 1) + n
T(-1) + 2 + 3 + … + (n – 1) + n

$$T(n) = T(-1) + \frac{n(2 + n)}{2} \quad \Rightarrow PA$$

$$T(-1) + \frac{2n + n^2}{2}$$

Portanto, T(n) é $O(n^2)$

c)

a = 8; b = 2; d = 2
$\log^a_b = \log^8_2 = 3 > d$
Se '$\log^8_2 > d$', então $O(n^{\log b(a)})$

Portanto, T(n) é $O(n^{\log b(a)}) = O(n^{\log 2(8)}) = O(n^3)$

d)

a = 2; b = 2; d = 1
$\log^a_b = \log^2_2 = 1 = d$
Se '$\log^2_2 = d$', então $O(n^d \lg n)$

Portanto, T(n) é $O(n^d \lg n) = O(n \lg n)$

e)

$T(n) = 2T(N - 1) + 1$
$2[2T(N - 2) + 1] + 1 = 2^2T(N - 2) + 2 + 1$
$2^2[2T(N - 3) + 1] + 2 + 1 = 2^3T(N - 3) + 2^2 + 2 + 1$
…
$T(n) = 2^kT(N - k) + 2^{k-1} + 2^{k-2} + \ldots + 2 + 1$

Se T(1) = 1, então $N - k = 1 \rightarrow N = k + 1 \rightarrow N - 1 = k$

$T(n) = 2^{N-1}T(N - N - 1) + 2^{N-1} + 2^{N-2} + \ldots + 2 + 1$
$2^{N-1}T(- 1) + 2^{N-1} + 2^{N-2} + \ldots + 2 + 1$

$$T(n) = 2^{N-1}T(- 1) + \frac{1(2^{N-1} - 1)}{(2 - 1)} \quad \Rightarrow PG$$

$T(n) = 2^{N-1}T(- 1) + 1(2^{N-1} - 1)$
$2^{N-1}T(- 1) + 2^{N-1} - 1$

Portanto, T(n) é $O(2^N)$

f)

a = 2; b = 2; d = 1;
$\log^a_b = \log^2_2 = 1 = d$
Se '$\log^2_2 = d$', então $O(n^d \lg n)$

Portanto, $T(n)$ é $O((n-1)^d \lg n) = O((n-1) \lg n) = O(n \lg n)$