1 What is the asymptotic running time of the code above for searching one array, as a function of the array length n?

$$\begin{aligned} &for\ i := 1\ to\ n\ do \\ &\quad if\ A[i] = t\ then \\ &\qquad\qquad\qquad\qquad return\ TRUE \\ &\quad return\ FALSE \end{aligned}$$

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. $O(n^2)$

(3) is correct. The algorithm in the worst case does n comparisons to evaluate whether in fact t does exist in the array. The algorithm is order of at most n and hence $O(n)$.

2 What is the asymptotic running time of the code above for searching two arrays, as a function of the array lengths n?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. $O(n^2)$

(3) is the correct answer. Given two n lenght arrays the number of comparison will be $2 \cdot n$. The order of growth is then at most order of n and hence $O(n)$.

3 What is the asymptotic running time of the code above for checking for a common element, as a function of the lenghts n?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. $O(n^2)$

(4) is the correct answer. $n \cdot \sum_{i=1}^{n} 1 = n \cdot n = n^2$ which is order of at most n^2 hence we have $O(n^2)$.

4 What is the asymptotic running time of the code above for checking for duplicates, as a function of the array length n?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. $O(n^2)$

(4) is correct.

$$\begin{aligned}\sum_{i=1}^{n}(n-i) &= n \cdot \sum_{i=1}^{n} 1 - \sum_{i=1}^{n} i \\ &= n^2 - \frac{n^2}{2} - \frac{n}{2} \\ &= \frac{n^2}{2} - \frac{n}{2}\end{aligned}$$

Which is order of at most n^2 and hence $O(n^2)$.

5 Let $T(n) = \frac{1}{2} \cdot n^2 + 3 \cdot n$. Which of the following statements are true?

1. $T(n) = O(n)$
2. $T(n) = \Omega(n)$
3. $T(n) = \Theta(n^2)$
4. $T(n) = O(n^3)$

(2), (3) and (4) are correct. We first show that (2) is correct. $\frac{1}{2} \cdot n^2 + 3 \cdot n \geq n, n \geq 1$ hence $\Omega(n)$. We next show that (3) is correct. $n^2 \leq \frac{1}{2} \cdot n^2 + 3 \cdot n \leq \frac{7}{2} \cdot n^2, n \geq 1$ hence $\Theta(n^2)$. Finally we show that (4) is correct. $\frac{1}{2} \cdot n^2 + 3 \cdot n \leq \frac{7}{2} \cdot n^2, n \geq 1$, hence $T(n) = O(n^3)$.

6 show that $T(n) = 2^{n+10} \implies T(n) = O(2^n)$

$2^{n+10} = 2^{10} \cdot 2^n \leq 2^{11} \cdot 2^n, n \geq 1$ hence $O(2^n)$.

7 If $T(n) = 2^{10 \cdot n} \implies T(n)$ is not $O(2^n)$

We proof this fact by contradiction. Suppose that in fact $2^{10 \cdot n}$ is indeed $O(2^n)$. Then there exists constants c and n_0 such that $2^{10 \cdot n} \leq c \cdot 2^n, n \geq n_0$. Dividing by 2^n this inequality becomes $2^{9 \cdot} \leq c, n \geq n_0$ which is clearly false hence we have shown that $T(n)$ could not be $O(2^n)$.

8 Let f and g denote functions from the positive integers to the nonnegative real numbers, and define

$$T(n) \quad = \quad max\{f(n), g(n)\}, n \geq 1.$$

show that $T(n) = \Theta(f(n) + g(n))$.

We show first that the function $T(n)$ is $O(f(n) + g(n))$. $max\{f(n), g(n)\} \leq f(n) + g(n), n \geq 1$ hence $O(f(n) + g(n))$. We next show that $T(n)$ is $\Omega(f(n) + g(n))$.

$$\begin{aligned} 2 \cdot max\{f(n), g(n)\} &\geq f(n) + g(n) \\ max\{f(n), g(n)\} &\geq \frac{1}{2} \cdot (f(n) + g(n)), n \geq 1 \end{aligned}$$

Hence $\Omega(f(n) + g(n))$. Since the function is $\Omega(f(n) + g(n))$ and $O(f(n) + g(n))$ then it must be $\Theta(f(n) + g(n))$.

9 Let f and g be non-decreasing real-valued functions defined on the positive integers. with $f(n)$ and $g(n)$ at least 1 for all $n \geq 1$. assume that $f(n) = O(g(n))$ and let c be a positive constant. Is $f(n) \cdot log_2(f(n)^c) = O(g(n) \cdot log_2(g(n))$

1. Yes, for all such f, g and c
2. Never, no matter what f, g and c are
3. Sometimes yes, sometimes no, depending on the constant c
4. Sometimes yes, sometimes no, depending on the function f and g

(1) is correct

$$\begin{aligned} f(n) &\leq c_1 g(n) \\ f(n^c) &\leq (c_1 g(n))^c \\ log_2(f(n^c)) &\leq log_2(c_1 g(n))^c \\ f(n) \cdot log_2(f(n^c)) &\leq g(n) \cdot log_2(c_1 \cdot g(n))^c) \\ f(n) \cdot log_2(f(n^c)) &\leq g(n) \cdot log_2 c1 + c \cdot g(n) \cdot log_2(g(n)) \\ f(n) \cdot log_2(f(n^c)) &\leq g(n) \cdot log_2 c1 \cdot log_2(g(n)) + c \cdot g(n) \cdot log_2(g(n)) \\ f(n) \cdot log_2(f(n^c)) &\leq (log_2 c1 + c) \cdot g(n) \cdot log_2(g(n)), n \geq 1 \end{aligned}$$

10 Assume again two positive non-decreasing function f and g such that $f(n) = O(g(n))$. Is $2^{f(n)} = O(2^{g(n)})$?

1. Yes, for all such f and g

2. Never, no matter what f and g are

3. Sometimes yes, sometimes no, depending on the functions f and g

4. Yes, whenever $f(n) \leq g(n)$ for all sufficiently large n

(3) and (4) are correct. To see why we first disprove (1) by contradiction. Let $f(n) = 2 \cdot n$ and $g(n) = n$ since $f(n) = O(g(n))$ then $2^{2 \cdot n} \leq 2^n$ but $2^{2n} = 4^n$ is clearly not $O(2^n)$ hence (1) is incorrect. (2) is also incorrect because if $f(n) = g(n)$ then $2^{f(n)} \leq 2^{g(n)}$ which is true. Hence we have that (3) is correct, the correctness of the argument depends on the functions f and g. We now show that (4) is correct. Since

$$\begin{aligned} 2^{f(n)} &\leq c \cdot 2^{g(n)} \\ 1 &\leq c \cdot 2^{g(n)-f(n)} \end{aligned}$$

This holds when $g(n) - f(n) \geq 0$ hence we have $g(n) \geq f(n)$. Thus (4) is correct.

11 Arrange the following function in order of increasing grow rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

1. $\sqrt{n}$
2. 10^n
3. $n^{1.5}$
4. $2^{\sqrt{log_2 n}}$
5. $n^{\frac{5}{3}}$

The list is as followed $2^{\sqrt{log_2 x}}, \sqrt{x}, x^{1.5}, x^{\frac{5}{2}}, 10^x$

12 Arrange the following functions in order of increasing growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

1. $n^2 log_2 n$
2. 2^n
3. 2^{2^n}
4. $n^{log_2 n}$
5. n^2

The list is as followed n^2,$n^2 log_2 n$,$n^{log_2 n}$ 2^n, 2^{2^n}

13 Arrange the following functions in order of increasing growth rate, with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

1. $2^{log_2 n}$
2. $2^{2^{log_2 n}}$
3. $n^{\frac{5}{2}}$
4. 2^{n^2}
5. $n^2 log_2 n$

$2^{log_2 n}$,$n^2 \cdot log_2 n$,$n^{\frac{5}{2}}$,$2^{2^{log_2 n}}$,2^{n^2}