Discrete Structures

Algorithms Practice Problems

Name and ID: ....

1. Fill in the following table with one of the three: O, Ω, Θ.

   Remark: If $f = \Theta(g)$ then $f = O(g)$ and $f = \Omega(g)$ are wrong answers.

	$f(n)$	???	$g(n)$
a	n		n^2
b	n^2		n
c	$2n$		$5n$
d	$1000000n$		$(1/100000)n$
e	$\log_2(n)$		$\log_2^2(n)$
f	$\log_2(n)$		$\log_{10}(n)$
g	$n\log_2(n)$		$n/\log_2(n)$
h	2^n		n^{100}
i	2^n		3^n
j	2^n		$n!$

2. Which of the following 10 functions are $O(n)$? Which are $\Omega(n)$? Which are $\Theta(n)$?

   Remark: If $f = \Theta(n)$ then $f = O(n)$ and $f = \Omega(n)$ are wrong answers.

	$f(n)$	???
a	2^n	
b	n^2	
c	$2n$	
d	$n/\log_2(n)$	
e	$\log_2(n)$	
f	$100\log_2(n)\log_2(n)$	
g	$n\log_2(n)$	
h	$10^{10}n/100^{100}$	
i	n^{π}	
j	$n!$	

3. Match the following 8 functions as 4 pairs: if $f(n)$ is paired with $g(n)$ then $f(n) = \Theta(g(n))$.

   Justify your pairings.

$$\underline{2^{n+1}}\ ;\ \underline{n}\ ;\ \underline{\log_2(n^2)}\ ;\ \underline{n^2}\ ;\ \underline{2^n}\ ;\ \underline{\log_2(n)}\ ;\ \underline{100n^2 - 500n}\ ;\ \underline{\log(2^n)}$$

4. Let P be a problem whose input is an array of size n for $n \geq 1$. Order the following ten algorithms from the most efficient to the least efficient. Justify your answer.
   - Algorithm A solves P with complexity $\Theta(n)$.
   - Algorithm B solves P with complexity $\Theta(2^n)$.
   - Algorithm C solves P with complexity $\Theta(n \log(n))$.
   - Algorithm D solves P with complexity $\Theta(n!)$.
   - Algorithm E solves P with complexity $\Theta(n^2)$.
   - Algorithm F solves P with complexity $\Theta(1)$.
   - Algorithm G solves P with complexity $\Theta(n^n)$.
   - Algorithm H solves P with complexity $\Theta(n^{100})$.
   - Algorithm I solves P with complexity $\Theta(\log(n))$.
   - Algorithm J solves P with complexity $\Theta(\sqrt{n})$.

5. For each of the following four parts, give an example of a function that satisfies the criteria or state that none exist. Justify your answers.
   (a) A function that is $O(n/2)$ and also $\Omega(2n)$.
   (b) A function that is both $\Omega(10n)$ and $O(n^2/100)$.
   (c) A function that is $O(5n)$ but not $\Theta(n/3)$.
   (d) A function that is $\Omega(2^n)$ but not $\Theta(2^n)$.

6. A problem P has an **upper bound** complexity $O(n^2)$ and a **lower bound** complexity $\Omega(n)$. Justify your answers to the following three questions.
   (a) Could someone design an algorithm that solves the problem with complexity n^3?
   (b) Could someone design an algorithm that solves the problem with complexity $0.5n$?
   (c) Could someone design an algorithm that solves the problem with complexity $100 \log(n)$?

7. Express the value of c when each of the following procedures terminates with the Θ-notation.

   Bonus: Try to find the exact value of c when each of the following procedures terminates.

   Justify your answers.

   (a) $f(n)$ (* $n = k^2$ is a positive square integer *)
   $c = 0$
   for $i = 1$ **to** n **do**
   if i is a square number
   then $c := c + 1$

   (b) $f(n)$ (* $n > 1000$ is a power of 2 *)
   $c = 0$
   while $n > 512$ **do**
   $n := n/2$
   $c := c + 1$

8. Consider the following procedure:

   $f(x, y)$ (* a positive multiple of 3 integer x and a positive integer y *)
   $c = 0$
   for $i = 1$ **to** $x/3$ **do**
   for $j = 1$ **to** $6y^2$ **do**
   then $c := c + 1$

   (a) As a function of x and y, what is the **exact** value of c when the program terminates?
   (b) Define x and y as functions of n such that $c = \Theta(n^3)$ when the program terminates.

9. Let $A = A[1] < A[2] < \cdots < A[n]$ be a sorted array containing n distinct negative and positive integers.

   Describe an efficient algorithm that finds, if it exists, an index $1 \le i \le n$ such that $A[i] = i$. What is the complexity of your algorithm?

   Justify the correctness of your algorithm and the correctness of your complexity claim.

10. For $n \ge 1$, let A be an array of size n for which the first k entries contain positive integers and the rest of the array is all zeros. The value of n is **known** but the value of k, which can be any number between 0 and n, is **unknown**.

    Examples:

    - $[34, 13, 21, 0, 0, 0, 0, 0]$: $k = 3$ in this array of length 8.
    - $[0, 0, 0, 0, 0, 0, 0]$: $k = 0$ in this array of length 7.
    - $[55, 8, 34, 13, 21, 89]$: $k = 6$ in this array of length 6.

    Describe an efficient algorithm that determines the value of k which is the number of positive integers in A. What is the complexity of your algorithm?

    Justify the correctness of your algorithm and the correctness of your complexity claim.

11. Let $A = A[1] \le A[2] \le \cdots \le A[n]$ be a sorted array of n integers. Let k be an integer.

    Describe an efficient algorithm that finds the number of times k appears in the array. What is the complexity of your algorithm?

    Justify the correctness of your algorithm and the correctness of your complexity claim.

12. For $n \ge 2$, let $A = A[1] < A[2] < \cdots < A[n]$ be a sorted array with n distinct positive integers from the range $1, 2, \ldots, n+1$. That is, exactly one of the integers from this range is missing in the array A.

    Examples: The missing integer in the array $[1, 2, 3, 4, 5, 7, 8, 9]$ is 6, the missing integer in the array $[1, 2, 4, 5]$ is 3, the missing integer in the array $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]$ is 12, the missing integer in the array $[2, 3, 4, 5, 6, 7]$ is 1, and the missing integer in the array $[1, 2, 3, 4, 5, 6, 7, 8, 9]$ is 10.

    Describe an efficient algorithm that finds the missing integer. The only questions about the integers in the arrays that your algorithm may ask are of the type "**is $\mathbf{A[i] = i}$?**" for some integer $1 \le i \le n$.

    What is the worst-case complexity (number of questions asked) of the algorithm?

    Justify the correctness of your algorithm and the correctness of your complexity claim.

13. For $n \ge 2$, let $A = A[1], \ldots, A[n]$ be an array of n positive integers. Let the sum of all the integers in the array be $M = A[1] + \cdots + A[n]$. For $1 \le i \le n$, let $S[i]$ be the sum of all the numbers in the array except $A[i]$.

    $$S[i] = M - A[i] = A[1] + \cdots + A[i-1] + A[i+1] + \cdots + A[n]$$

    Example: Let $A = [16, 2, 128, 64, 1, 8, 32, 4]$. Then $M = 255$ and $S = [239, 253, 127, 191, 254, 247, 223, 251]$.

    Design a linear time algorithm ($\Theta(n)$) to compute $S[1], \ldots, S[n]$ **only with plus operations** (you are not allowed to use minus operations).

    What is the **exact** number of plus operations used by your algorithm?

    Justify the correctness of your algorithm and the correctness of your complexity claim.

14. For $n \ge 3$, let $A = A[1] < A[2] < \cdots < A[n]$ be a sorted array containing n distinct positive integers.

    Describe an efficient algorithm that finds two distinct integers from the array, $A[i]$ and $A[j]$ (for $1 \le i \ne j \le n$) whose some $A[i] + A[j]$ is even.

    What is the complexity of the algorithm?

    Justify the correctness of your algorithm and the correctness of your complexity claim.