50.004 – Algorithms

Jan–Apr Term, 2022

Homework Set 1

Due by: Week 2 Monday (31 Jan 2022) 1pm.
Please submit your homework online via eDimension.

Note: In this homework set, log denotes the natural logarithm[1], i.e. the logarithm in base e.

Question 1. Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ be algorithms, each of whose required input is an array of any length. Given an input array of length $n \geq 0$, suppose that the running times of $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are denoted by the functions $f(n)$, $g(n)$, $h(n)$ respectively. You may assume that the running times are positive. For each statement listed below, indicate whether it is **true (T)** or **false (F)**, and **justify your answer with details**. [5 points]

(i) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$.
(ii) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.
(iii) Whatever $f(n)$ is, we know that the running time of algorithm $\mathcal{A}$ is always at least $\Omega(1)$.
(iv) If $f(n) = \Theta(n)$ and $g(n) = \Theta(2n)$, then for all input sizes $n \geq 1$, algorithm $\mathcal{A}$ must always run faster than algorithm $\mathcal{B}$.
(v) If $f(n) = O(n)$ and $g(n) = O(\log n)$, then for all input sizes $n \geq 0$, algorithm $\mathcal{B}$ must always run faster than algorithm $\mathcal{A}$.

Question 2. Each of the items (i)–(v) below has only one correct option, among the possible options (a), (b), (c), (d). Please clearly indicate your answers. You do not have to justify your answers. [5 points]

(i) Consider the function $f(n) = \sum_{i=0}^{n} i^3$.

- Adam says "$f(n)$ is $\Theta(n^4)$."
- Beatrice says "$f(n)$ is $\Theta(n^5)$."
- Colin says "$f(n)$ is $O(n^5)$."
- Diana says "$f(n)$ is $\Omega(n^3)$."

Who is correct?
(a) Only Adam.
(b) Only Beatrice.
(c) Adam, Colin, and Diana, but not Beatrice.
(d) Beatrice, Colin, and Diana, but not Adam.

(ii) Algorithm $\mathcal{A}$ takes $n \log_2 n$ microseconds to solve a problem with input size n, while algorithm $\mathcal{B}$ takes n^2 microseconds to solve the same problem with input size n. What is the approximate time needed for algorithms $\mathcal{A}$ and $\mathcal{B}$ to solve the same problem with input size 10 million? (Note: 1 microsecond equals 10^{-6} second.)
(a) A few seconds and a few minutes, respectively.
(b) A few minutes and a few hours, respectively.
(c) A few minutes and a few days, respectively.
(d) A few minutes and a few years, respectively.

[1]The logarithm notation is not consistently used throughout various academic disciplines, so what "log" (with no subscript) means would depend on the context. For example, in mathematics (especially in functional analysis and analysis-related areas), log by default is natural logarithm. In information theory, log is commonly used to denote base 2 logarithm. In some engineering areas (e.g. communication theory), log is sometimes in base 10, but sometimes in base 2. More theoretical engineering papers would use log to mean natural logarithm. The common good practice in research papers is to specify which base is being used in logarithms, since there is actually no "universally accepted convention".

(iii) Let A be a given unsorted array of $n \geq 2$ distinct non-zero integers. Suppose we want to arrange the entries so that every negative integer always appears before (i.e. to the left of) any positive integer. In the worst case scenario, what is the least number of pairs of entries that have to be swapped, so that the desired arrangement property is satisfied?
  (a) $n - 1$ pairs to be swapped.
  (b) $\lfloor \frac{n}{2} \rfloor$ pairs to be swapped.
  (c) $\frac{n(n-1)}{2}$ pairs to be swapped.
  (d) None of the above.

(iv) Assume we have a database with 10^k records, where k is some unspecified integer. Suppose there are two machines $\mathcal{A}$ and $\mathcal{B}$, where machine $\mathcal{A}$ takes $0.0001n^2$ microseconds to process n records, while machine $\mathcal{B}$ takes $10n\log_{10} n$ microseconds to process n records. What is the smallest possible value for k such that machine $\mathcal{A}$ would have a higher efficiency over machine $\mathcal{B}$?
  (a) 5.
  (b) 6.
  (c) 7.
  (d) 8.

(v) Which of the following sequences of functions is in ascending order in terms of asymptotic complexity? (By ascending order, we mean that if $g(n)$ follows $f(n)$, then it should be the case that $f(n) = O(g(n))$.)
  (a) $\sqrt{\log n}$, $\log(\log n)$, $2^{\log n}$, $n \log n$
  (b) $(\log n)^{100}$, $\frac{n}{100}$, $2^{\log 2n^{\frac{3}{2}}}$, $n \log n$
  (c) $\sqrt{n}$, $\frac{n}{4}(\log n)^3$, n^2
  (d) $2^{\sqrt{\log n}}$, $\sqrt{n}$, $n(\log n)^2$, $n^{3/2}$

Question 3. Each of the items (i)–(v) below is a program written in either Python or C, for which you have to determine its running time complexity. You may assume that n is an unspecified positive integer that has already been defined. You do not have to justify your answers. [5 points]

(i) Determine the time complexity of the program below (in terms of n) using the Big-Θ notation [1 point]

```
for i in range(n**2):
    for j in range(i):
        print("Algorithm!")
```

(ii) Determine the time complexity of the program below (in terms of n) using the Big-Θ notation. [1 point]

```
def f(m):
    if m <= 0:
        return 0
    else:
        return f(m-1) + m
print(n)
```

(iii) Determine the time complexity of the program below (in terms of n) using the Big-Θ notation. [1 point]

```
for i in range(n):
    for j in range(n**2):
        print("Algorithm!")
```

(iv) Determine the time complexity of the program below (in terms of n and k) using the Big-Θ notation. Assume that $k > 1$ is a constant. [1 point]

```
i=1
while i < n:
    i *= k
```

(v) Determine the time complexity of the program below (in terms of n using the Big-Θ notation. [1 point]

```
int temp = 0
for (int i=n; i>0; i/=2){
    for (int j=0; j<i; j++){
        temp++;
    }
}
```

Question 4. In every row of the table below, there are two functions $f(n)$ and $g(n)$ given. Determine whether each of the three statements "$f(n) = O(g(n))$", "$f(n) = \Omega(g(n))$", "$f(n) = \Theta(g(n))$" is true or false for the given functions. You may indicate your answer as T/F in the blanks provided in the table. Please assume that k and c are both constants > 1. You do not have to justify your answers. [5 points]

$\boldsymbol{f(n)}$	$\boldsymbol{g(n)}$	$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
n^k	c^n			
$\log_2 n$	$\log_8 n$			
8^n	4^n			
3^n	$n2^n$			
$\log n!$	$n \log n$			

Question 5. Please answer the following questions with **detailed justification**.

(i) Let $f(n)$ and $g(n)$ be asymptotically non-negative functions. Prove that $f(n) + g(n) = \Theta(\max(f(n), g(n)))$ [1 point]

(ii) In the beginning, we have two algorithms $\mathcal{A}$ and $\mathcal{B}$. Given an input of size n, the running times of $\mathcal{A}$ and $\mathcal{B}$ are closely related: The running time of $\mathcal{A}$ is $f(n)$, and the running time of $\mathcal{B}$ is $n/f(n)$. Suppose we have a third algorithm $\mathcal{C}$ that runs $\mathcal{A}$ and $\mathcal{B}$ consecutively (i.e. when executing $\mathcal{C}$, the machine will run $\mathcal{A}$ first, and then run $\mathcal{B}$). How do we choose $f(n)$ so that our machine can finish running $\mathcal{C}$ in the shortest time? [1 point]
If we have another algorithm $\mathcal{D}$ that enables our machine to run $\mathcal{A}$ and $\mathcal{B}$ simultaneously, will the time complexity of $\mathcal{D}$ be different from the time complexity of $\mathcal{C}$? [1 point].

(iii) Suppose that we have an algorithm whose running time is $T(n) = 2T(n/2) + \Theta(n)$. This algorithm is recursive: It solves the whole computational problem by splitting the problem into two sub-problems, and then combining the two sub-problems, where this combination process runs in $\Theta(n)$ time. What is the time complexity $T(n)$ of the algorithm? Please provide detailed justification. [2 points]