Data Structures and Algorithms

HUS - 2023 HKII

Assignment 1

Lecturer: Vũ Đức Minh - Trần Bá Tuấn

§ Độ phức tạp thuật toán §

Phần 1: Reinforcement

R-4. 1. *Graph the functions $8n$, $4nlogn$, $2n^2$, n^3, and 2^n using a logarithmic scale for the $x-$ and $y-axes$; that is, if the function value $f(n)$ is y, plot this as a point with x-coordinate at $logn$ and y coordinate at $logy$.*

R-4. 2. *The number of operations executed by algorithms A and B is $8nlogn$ and $2n^2$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.*

R-4. 3. *The number of operations executed by algorithms A and B is $40n^2$ and $2n^3$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$.*

R-4. 4. *Give an example of a function that is plotted the same on a log-log scale as it is on a standard scale.*

R-4. 5. *Explain why the plot of the function n^c is a straight line with slope c on a $log - log$ scale.*

R-4. 6. *What is the sum of all the even numbers from 0 to $2n$, for any integer $n \geq 1$?*

R-4. 7. *Show that the following two statements are equivalent:*

(a) The running time of algorithm A is always $O(f(n))$.

(b) In the worst case, the running time of algorithm A is $O(f(n))$.

R-4. 8. *Order the following functions by asymptotic growth rate.*

$$\begin{array}{ccc} 4nlogn + 2n & 2^{10} & 2^{logn} \\ 3n + 100logn & 4n & 2^n \\ n^2 + 10n & n^3 & nlogn \end{array}$$

R-4. 9. *Give a big-Oh characterization, in terms of n, of the running time of the example1 method shown in Code Fragment 4.12.*

R-4. 10. *Give a big-Oh characterization, in terms of n, of the running time of the example2 method shown in Code Fragment 4.12.*

R-4. 11. *Give a big-Oh characterization, in terms of n, of the running time of the example3 method shown in Code Fragment 4.12.*

R-4. 12. *Give a big-Oh characterization, in terms of n, of the running time of the example4 method shown in Code Fragment 4.12.*

R-4. 13. *Give a big-Oh characterization, in terms of n, of the running time of the example5 method shown in Code Fragment 4.12.*

R-4. 14. *Show that if $d(n)$ is $O(f(n))$, then $ad(n)$ is $O(f(n))$, for any constant $a > 0$.*

R-4. 15. *Show that if $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then the product $d(n)e(n)$ is $O(f(n)g(n))$.*

R-4. 16. *Show that if $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n) + e(n)$ is $O(f(n) + g(n))$.*

Code Fragment 4.12: Some sample algorithms for analysis.

```java
// Returns the sum of the integers in given array.
 public static int example1(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        total += arr[j];
    return total;
 }

// Returns the sum of the integers with even index in given array.
 public static int example2(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j += 2) // note the increment of 2
        total += arr[j];
    return total;
 }

 // Returns the sum of the prefix sums of given array.
 public static int example3(int[ ] arr) {
    int n = arr.length, total = 0;
    for (int j=0; j < n; j++) // loop from 0 to n-1
        for (int k=0; k <= j; k++) // loop from 0 to j
            total += arr[j];
    return total;
 }

 // Returns the sum of the prefix sums of given array.
 public static int example4(int[ ] arr) {
    int n = arr.length, prefix = 0, total = 0;
    for (int j=0; j < n; j++) { // loop from 0 to n-1
         prefix += arr[j];
         total += prefix;
    }
    return total;
 }

 // Returns the number of times second array stores sum of prefix sums from
  ↪ first.
 public static int example5(int[ ] first, int[ ] second) {
 // assume equal-length arrays
    int n = first.length, count = 0;
    for (int i=0; i < n; i++) { // loop from 0 to n-1
        int total = 0;
        for (int j=0; j < n; j++) // loop from 0 to n-1
            for (int k=0; k <= j; k++) // loop from 0 to j
                total += first[k];
        if (second[i] == total) count++;
    }
    return count;
}
```

R-4. 17. *Show that if $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n) - e(n)$ is not necessarily $O(f(n) - g(n))$.*

R-4. 18. *Show that if $d(n)$ is $O(f(n))$ and $f(n)$ is $O(g(n))$, then $d(n)$ is $O(g(n))$.*

R-4. 19. *Show that $O(max\{f(n), g(n)\}) = O(f(n) + g(n))$.*

R-4. 20. *Show that $f(n)$ is $O(g(n))$ if and only if $g(n)$ is $\Omega(f(n))$.*

R-4. 21. *Show that if $p(n)$ is a polynomial in n, then $logp(n)$ is $O(logn)$.*

R-4. 22. *Show that $(n + 1)^5$ is $O(n^5)$.*

R-4. 23. *Show that 2^{n+1} is $O(2^n)$.*

R-4. 24. *Show that n is $O(nlogn)$.*

R-4. 25. *Show that n^2 is $\Omega(nlogn)$.*

R-4. 26. *Show that $nlogn$ is $\Omega(n)$.*

R-4. 27. *Show that $\lceil f(n) \rceil$ is $O(f(n))$, if $f(n)$ is a positive non-decreasing function that is always greater than 1.*

R-4. 28. *For each function $f(n)$ and time t in the following table, determine the largest size n of a problem P that can be solved in time t if the algorithm for solving P takes $f(n)$ microseconds (one entry is already completed).*

	1 Second	1 Hour	1 Month	1 Century
$logn$	$\approx 10^{300000}$			
n				
$nlogn$				
n^2				
2^n				

R-4. 29. *Algorithm A executes an $O(logn)$-time computation for each entry of an array storing n elements. What is its worst-case running time?*

R-4. 30. *Given an n-element array X, Algorithm B chooses $logn$ elements in X at random and executes an $O(n)$-time calculation for each. What is the worst-case running time of Algorithm B?*

R-4. 31. *Given an n-element array X of integers, Algorithm C executes an $O(n)$-time computation for each even number in X, and an $O(logn)$-time computation for each odd number in X. What are the best-case and worst-case running times of Algorithm C?*

R-4. 32. *Given an n-element array X, Algorithm D calls Algorithm E on each element $X[i]$. Algorithm E runs in $O(i)$ time when it is called on element $X[i]$. What is the worst-case running time of Algorithm D?*

R-4. 33. *Al and Bob are arguing about their algorithms. Al claims his $O(nlogn)$ time method is always faster than Bob's $O(n^2)$-time method. To settle the issue, they perform a set of experiments. To Al's dismay, they find that if $n < 100$, the $O(n^2)$-time algorithm runs faster, and only when $n \geq 100$ is the $O(nlogn)$-time one better. Explain how this is possible.*

R-4. 34. *There is a well-known city (which will go nameless here) whose inhabitants have the reputation of enjoying a meal only if that meal is the best they have ever experienced in their life. Otherwise, they hate it. Assuming meal quality is distributed uniformly across a person's life, describe the expected number of times inhabitants of this city are happy with their meals?*

Phần 2: Creativity

C-4. 1. *Assuming it is possible to sort n numbers in $O(nlogn)$ time, show that it is possible to solve the three-way set disjointness problem in $O(nlogn)$ time.*

C-4. 2. *Describe an efficient algorithm for finding the ten largest elements in an array of size n. What is the running time of your algorithm?*

C-4. 3. *Give an example of a positive function $f(n)$ such that $f(n)$ is neither $O(n)$ nor $\Omega(n)$.*

C-4. 4. *Show that $\sum_{i=1}^{n} i^2$ is $O(n^3)$.*

C-4. 5. *Show that $\sum_{i=1}^{n} \frac{i}{2^i} < 2$.*

C-4. 6. *Determine the total number of grains of rice requested by the inventor of chess.*

C-4. 7. *Show that $log_b f(n)$ is $\Theta(logf(n))$ if $b > 1$ is a constant.*

C-4. 8. *Describe an algorithm for finding both the minimum and maximum of n numbers using fewer than $3n/2$ comparisons.*

C-4. 9. *Bob built a website and gave the URL only to his n friends, which he numbered from 1 to n. He told friend number i that he/she can visit the website at most i times. Now Bob has a counter, C, keeping track of the total number of visits to the site (but not the identities of who visits). What is the minimum value for C such that Bob can know that one of his friends has visited his/her maximum allowed number of times?*

C-4. 10. *Draw a visual justification of Proposition 4.3 analogous to that of Figure 4.3(b) for the case when n is odd.*

C-4. 11. *An array A contains $n-1$ unique integers in the range $[0, n-1]$, that is, there is one number from this range that is not in A. Design an $O(n)$-time algorithm for finding that number. You are only allowed to use $O(1)$ additional space besides the array A itself.*

C-4. 12. *Perform an asymptotic analysis of the insertion-sort algorithm given in Section 3.1.2. What are the worst-case and best-case running times?*

C-4. 13. *Communication security is extremely important in computer networks, and one way many network protocols achieve security is to encrypt messages. Typical cryptographic schemes for the secure transmission of messages over such networks are based on the fact that no efficient algorithms are known for factoring large integers. Hence, if we can represent a secret message by a large prime number p, we can transmit, over the network, the number $r = pq$, where $q > p$ is another large prime number that acts as the encryption key. An eavesdropper who obtains the transmitted number r on the network would have to factor r in order to figure out the secret message p. Using factoring to figure out a message is hard without knowing the encryption key q. To understand why, consider the following naive factoring algorithm:*

```
for (int p=2; p < r; p++)
    if (r % p == 0)
        return "The secret message is p!";
```

a. *Suppose the eavesdropper's computer can divide two 100-bit integers in μs (1 millionth of a second). Estimate the worst-case time to decipher the secret message p if the transmitted message r has 100 bits.*

b. *What is the worst-case time complexity of the above algorithm? Since the input to the algorithm is just one large number r, assume that the input size n is the number of bytes needed to store r, that is, $n = [(log_2 r)/8] + 1$, and that each division takes time $O(n)$.*

C-4. 14. *Al says he can prove that all sheep in a flock are the same color:*
Base case: One sheep. It is clearly the same color as itself.
Induction step: A flock of n sheep. Take a sheep, a, out. The remaining $n-1$ are all the same color by induction. Now put sheep a back in and take out a different sheep, b. By induction, the $n-1$ sheep (now with a) are all the same color. Therefore, all the sheep in the flock are the same color. What is wrong with Al's "justification"?

C-4. 15. *Consider the following "justification" that the Fibonacci function, $F(n)$ is $O(n)$:*
Base case ($n \leq 2$): $F(1) = 1$ and $F(2) = 2$.
Induction step ($n > 2$): Assume claim true for $n' < n$. Consider n. $F(n) = F(n-2) + F(n-1)$. By induction, $F(n-2)$ is $O(n-2)$ and $F(n-1)$ is $O(n-1)$.
Then, $F(n)$ is $O((n-2)+(n-1))$, by the identity presented in Exercise $R-4.16$.
Therefore, $F(n)$ is $O(n)$.
What is wrong with this "justification"?

C-4. 16. *Consider the Fibonacci function, $F(n)$ (see Proposition 4.20). Show by induction that $F(n)$ is $\Omega((3/2)^n)$.*

C-4. 17. *Let S be a set of n lines in the plane such that no two are parallel and no three meet in the same point. Show, by induction, that the lines in S determine $\Theta(n^2)$ intersection points.*

C-4. 18. *Show that the summation $\sum_{i=1}^{n} logi$ is $O(nlogn)$.*

C-4. 19. *Show that the summation $\sum_{i=1}^{n} logi$ is $\Omega(nlogn)$.*

C-4. 20. *Let $p(x)$ be a polynomial of degree n, that is, $p(x) = \sum_{i=0}^{n} a_i x^i$.*

a. *Describe a simple $O(n^2)$-time algorithm for computing $p(x)$.*

b. Describe an $O(nlogn)$-time algorithm for computing $p(x)$, based upon a more efficient calculation of x^i.

c. Now consider a rewriting of p(x) as $p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + ... + x(a_{n-1} + xa_n)...)))$, which is known as ***Horner's method****. Using the big-Oh notation, characterize the number of arithmetic operations this method executes.*

C-4. 21. *An evil king has n bottles of wine, and a spy has just poisoned one of them. Unfortunately, they do not know which one it is. The poison is very deadly; just one drop diluted even a billion to one will still kill. Even so, it takes a full month for the poison to take effect. Design a scheme for determining exactly which one of the wine bottles was poisoned in just one month's time while expending $O(logn)$ taste testers.*

C-4. 22. *An array A contains n integers taken from the interval $[0, 4n]$, with repetitions allowed. Describe an efficient algorithm for determining an integer value k that occurs the most often in A. What is the running time of your algorithm?*

C-4. 23. *Given an array A of n positive integers, each represented with $k = [logn]+1$ bits, describe an $O(n)$-time method for finding a k-bit integer not in A.*

C-4. 24. *Argue why any solution to the previous problem must run in $\Omega(n)$ time.*

C-4. 25. *Given an array A of n arbitrary integers, design an $O(n)$-time method for finding an integer that cannot be formed as the sum of two integers in A.*

Phần 3: Projects

P-4. 1. *Perform an experimental analysis of the two algorithms $prefixAverage1$ and $prefixAverage2$, from Section 4.3.3. Visualize their running times as a function of the input size with a log-log chart.*

P-4. 2. *Perform an experimental analysis that compares the relative running times of the methods shown in Code Fragment 4.12.*

P-4. 3. *Perform an experimental analysis to test the hypothesis that $Java'sArray.sort$ method runs in $O(nlogn)$ time on average.*

P-4. 4. *For each of the algorithms $unique1$ and $unique2$, which solve the element uniqueness problem, perform an experimental analysis to determine the largest value of n such that the given algorithm runs in one minute or less.*