The University of Michigan
Electrical Engineering & Computer Science
EECS 281: Data Structures and Algorithms
Fall 2024

Lab 3: Complexity Analysis, Amortization, and Strings

10/10

Instructions:
Work on this document with your group, then enter the answers on the canvas quiz.

Note:
Be prepared before you meet with your lab group, and read this document so that you know what you must submit for full credit. You can even start it ahead of time and then ask questions during any lab section for help completing it.

You MUST include the following assignment identifier at the top of every file you submit to the autograder as a comment. This includes all source files, header files, and your Makefile (if there is one). If there is no autograder assignment, you may ignore this.

Project Identifier: 5AE7C079A8BF493DDDB6EF76D42136D183D8D7A8

The Autograder has changed from previous semesters. If you are taking this class for the second time, you may need to alter the code that you have from the last time you took the class.

---

1 Logistics

1. What is the due date of lab 2 quiz and AG?
    A. 9/21/2024
    B. 9/23/2024
    C. 9/19/2024
    D. 9/25/2024
2. What is the due date of lab 3 handwritten?
    A. 9/24/2024
    B. 9/27/2024
    C. 9/23/2024
    D. 9/26/2024
3. What is the due date of lab 3 quiz and AG?
    A. 9/28/2024
    B. 9/26/2024
    C. 10/04/2024
    D. 9/30/2024
4. What is the due date of Project 2?
    A. 10/6/2024
    B. 10/10/2024
    C. 10/12/2024
    D. 10/14/2024

2 Recurrence Relations and Complexity Analysis

5. What is the best approach for deducing the following relation, given $T(1) = 1$?

$$T(n) = 8T(\frac{n}{2}) + 3n$$

A. The Substitution Method
B. The Master Theorem

6. What is the best approach for deducing the following relation, given $T(1) = 1$?

$$T(n) = 2T(n-1) + 2$$

A. The Substitution Method
B. The Master Theorem

7. Given the function below, calculate the recurrence relation. Assume bar(n) runs in $\Theta(n)$ time.

```
void foo(int n){
    if (n == 1)
        return;

    foo(n / 4);

    int k = n * n;

    for (int i = 0; i < k; ++i)
        for (int j = 0; j < n; ++j)
            bar(n);

    for (int i = 0; i < n; ++i)
        foo(n / 2);

    bar(k);
} // foo()
```

A. $T(n) = T(\frac{n}{4}) + n^3 \log n + T(\frac{n}{2}) + n^2$
B. $T(n) = T(\frac{n}{4}) + n^3 \log n + nT(\frac{n}{2}) + n^2$
C. $T(n) = T(\frac{n}{4}) + n^4 + nT(\frac{n}{2}) + n^2$
D. $T(n) = T(\frac{n}{4}) + n^4 \log n + nT(\frac{n}{2}) + n^2$

8. Which ordering lists $\Theta(n^{100}), \Theta(100^n), \Theta(n!), \Theta(\log(n!))$ in order of increasing complexity? *Hint*: $\Theta(\log(n!))$ can be simplified to a complexity class that you've likely seen before. Think about any mathematical identities that may help you perform this simplification.

   A. $\Theta(n^{100}), \Theta(100^n), \Theta(\log(n!)), \Theta(n!)$
   B. $\Theta(100^n), \Theta(n^{100}), \Theta(\log(n!)), \Theta(n!)$
   C. $\Theta(\log(n!)), \Theta(n^{100}), \Theta(n!), \Theta(100^n)$
   D. $\Theta(\log(n!)), \Theta(n^{100}), \Theta(100^n), \Theta(n!)$
   E. $\Theta(\log(n!)), \Theta(100^n), \Theta(n^{100}), \Theta(n!)$

$\log(n!) = \sum_{i=1}^{n} \log i = O(n \lg n)$

9. Solve the following recurrence relation:

$$T(n) = \begin{cases} 281, & \text{if } n = 1 \\ T(n-1) + 370, & \text{if } n > 1 \end{cases}$$

   A. 281n + 370
   B. 281n + 651
   C. 370n + 281
   D. 370n - 89
   E. 370n - 459

10. Consider the following recurrence relation:

$$T(n) = \begin{cases} c_0, & \text{if } n = 1 \\ \sqrt{2}T(\frac{n}{2}) + c_1, & \text{if } n > 1 \end{cases}$$

What is the tightest complexity class that you can attribute to this recurrence relation?

   A. $T(n) = \Theta(\sqrt{n})$
   B. $T(n) = O(\sqrt{n})$
   C. $T(n) = \Omega(\log_2 n)$
   D. $T(n) = \Theta(n \log_2 n)$
   E. $T(n) = \Theta(n^2)$

$T(n) = \sqrt{2}\,T(\frac{n}{2}) + c_1$

$T(\frac{n}{2}) = \sqrt{2}\,T(\frac{n}{4}) + c_1$

$\vdots$

$T(2) = \sqrt{2}\,T(1) + c_1$

$T(n) = (\sqrt{2})^{\log_2 n} T(1) + c_1 \log_2 n$

$= O\left((2^{\frac{1}{2}})^{\log_2 n}\right) = O(n^{\frac{1}{2}}) = O(\sqrt{n})$

11. What is the worst-case complexity of this function?

```
void foo(vector &v) {
    int n = static_cast(v.size());
    int rt = static_cast(floor(sqrt(n)));

    for (int i = 0; i < rt; ++i) {
        for (int j = 0; j < rt * rt; j += rt) {
            cout << v[j + i] << " ";
        }
        cout << endl;
    }
}
```

A. $\Theta(\sqrt{n})$
B. $\Theta(n)$
C. $\Theta(n\sqrt{n})$
D. $n \log n$
E. n^2

12. What is the worst-case time complexity of this function?

```
void potato(int n) {
    for (int i = 1; i < n; i *= 2) {
        for (int j = 1; j < n; ++j) {
            for (int k = 1; k < j; ++k) {
                cout << "I'm a smart potato! o(^-^)o \n";
}
```

A. $\Theta(n^2)$
B. $\Theta(n^2 \log n)$
C. $\Theta(n \log^2 n)$
D. $\Theta(n^2 \log^2 n)$
E. $\Theta(n^3)$

3 Vectors and Strings

13. What is the worst-case time complexity of the `push_back` operation discussed in the lab slides, given a vector of size n?

    A. $\Theta(1)$

    B. $\Theta(\log n)$

    C. $\Theta(n)$

    D. $\Theta(n^2)$

14. What is the amortized time complexity of the `push_back` operation discussed in the lab slides, given a vector of size n?

    A. $\Theta(1)$

    B. $\Theta(\log n)$

    C. $\Theta(n)$

    D. $\Theta(n^2)$

15. Consider the following snippet of code:

```
char str1[] = "BING";
cout << strlen(str1);
cout << sizeof(str1);

string str2("BING");
cout << str2.length();
cout << str2.size();
```

    What is the output?

    A. 4444

    B. 4455

    C. 4544

    D. 5544

    E. None of the above is correct.

16. Consider the following snippet of code:

```
const char *s = "hello";
char ss[20];
int length = strlen(s);
for (int i = 0; i < length; ++i)
    ss[i] = s[length - i];
printf("%s", ss);
```

    What is the output?

    A. hello

    B. olleh

    C. olle

    D. segmentation fault

    E. no output is printed

