Assignment 2

Problems identified by x.y(z) denote the problem “y”, in chapter “x” of the textbook, with part “z”. If “z” is not noted, then the entire problem is required.

Assignment 2a

- 2.5(a, c, e) Use the master theorem, show work.
- Solve recurrence relation \( T(n) = 2T(n/3) + n \). Use the master theorem, show work.

Assignment 2b

- 2.5(b, d) Use the master theorem, show comparison.
- Solve recurrence relation \( T(n) = 8T(n/3) + n^2 \). Use the master theorem, show work.
- 2.5(g) Use the substitution method. Show the pattern and determination of \( k_{\text{max}} \).
- Write the function \( \text{unsigned int binary_search( const std::vector< int > &data, int value )} \). Verify that the function will correctly find the index of \( \text{value} \) within \( \text{data} \). You may assume that \( \text{value} \) is present, and \( \text{data} \) is already sorted in ascending order. At the top of your source, include a comment with your estimated Big-Oh complexity of the algorithm.

Assignment 2c

- 2.5(f, h) Use the substitution method. Show the pattern and determination of \( k_{\text{max}} \).
- 2.16 Find an algorithm, give pseudo-code, argue correctness, analyze the runtime, showing it is \( \text{O}(\text{log}(n)) \). The values stored are integers, not necessarily positive Hint: You should know how to find items in a sorted array in \( \text{O}(\text{log}(n)) \).
- Write the function \( \text{unsigned int ternary_search( const std::vector< int > &data, int value )} \). Verify that the function will correctly find the index of \( \text{value} \) within \( \text{data} \). You may assume that \( \text{value} \) is present, and \( \text{data} \) is already sorted in ascending order. At the top of your source, include a comment with your estimated Big-Oh complexity of the algorithm. \( \text{ternary_search} \) divides its input array into 3 equally sized groups, in the same way that \( \text{binary_search} \) divides into 2 equally sized groups.

Assignment 2d

- 2.5(i, j) Use the substitution method. Show the pattern and determination of \( k_{\text{max}} \).
- 2.19 Analyze the complexity of the algorithm for part (a). Provide your divide and conquer solution and its complexity analysis for part (b).
- Time \( \text{binary_search} \) and \( \text{ternary_search} \) on vectors of sizes \( 2^0, 2^1, ... , 2^{30} \). Be sure to do correct statistical data collection. Submit a table of the data collected, and declaration of which appears to be faster.

Assignment 2e

- 2.5(k) Use the substitution method. Show the pattern and determination of \( k_{\text{max}} \).
- 2.22 Find an algorithm, give pseudo-code, argue correctness, analyze the runtime, showing it is \( \text{O}(\text{log}(m + \text{log}(n))) \).
- If one algorithm is \( \text{O}(\text{log}(m+n)) \), another is \( \text{O}(\text{log}(m) + \text{log}(n)) \), which is more efficient? Give your proof.
- Chart the normalized runtimes of \( \text{binary_search} \) and \( \text{ternary_search} \), along with \( N^{1/2}, N^{1/3}, \text{LOG}_2(N), \text{LOG}_3(N) \) and 1. Submit the chart, and a statement discussing which algorithm has better Big-Oh, and which algorithm is faster.

Assignment 2f, Due Never (optional)

- 2.14 Find a divide-and-conquer algorithm, write the recurrence relation, solve it.
- 2.34 Find a divide-and-conquer algorithm, write the recurrence relation, solve it. The book says “linear”. We are not as optimistic. Any polynomial divide-and-conquer algorithm is acceptable.

Assignment 2z, Due Never (optional)

- 2.4(A) Write down the recurrence relation. Solve it.
- 2.4(B) Write down the recurrence relation. Solve it.
- 2.4(part C) Write down the recurrence relation. Solve it.
- 2.4 Which would you choose?
• 2.25(a) Fill in the missing code, give a recurrence relation, and solve it.
• 2.25(b) Fill in the missing code, give a recurrence relation, and solve it.
• 2.17 Find an algorithm, prove the runtime is O(log(n)).

Submission

• Submit your solutions by the due date and time. For written problems, your work and answers as a PDF. For code, submit the source code. For tables and graphs, submit a PDF.