CS 3510: Advanced Algorithms/Data Structures
Spring 2019 Assignment 2

Problems due as noted.

Assignment

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, Due Jan 17

- 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, Due Jan 19

- 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 $k_{max}$.

Assignment 2c, Due Jan 22

- 2.5(f, h) Use the substitution method. Show the pattern and determination of $k_{max}$.
- 2.16 Find an algorithm, give pseudo-code, argue correctness, analyze the runtime, showing it is $O(\log(n))$. The values stored are integers, not necessarily positive Hint: You should know how to find items in a sorted array in $O(\log(n))$.
- Write the function `unsigned int binary_search(const std::vector<int> &data, int value)` Verify that the function will correctly find the index of `value` within `data`. You may assume that `value` is present, and `data` is already sorted in ascending order. Submit statement of correctness, and estimated Big-Oh complexity of the algorithms. For Python students, your function will receive a list of numbers, and a number from the list. It will return the index of the number. (Same as the C++ version).

Assignment 2d, Due Jan 24

- 2.5(i, j) Use the substitution method. Show the pattern and determination of $k_{max}$.
- 2.19 Analyze the complexity of the algorithm for part (a). Provide your divide and conquer solution and its complexity analysis for part (b).
- Write the function `unsigned int ternary_search(const std::vector<int> &data, int value)`. Verify that the function will correctly find the index of `value` within `data`. You may assume that `value` is present, and `data` is already sorted in ascending order. Submit statement of correctness, and estimated Big-Oh complexity of the algorithms. For Python students, your function will receive a list of numbers, and a number from the list. It will return the index of the number. (Same as the C++ version). `ternary_search` divides its input array into 3 equally sized groups, in the same way that `binary_search` divides into 2 equally sized groups.

Assignment 2e, Due Jan 26

- 2.5(k) Use the substitution method. Show the pattern and determination of $k_{max}$.
- 2.22 Find an algorithm, give pseudo-code, argue correctness, analyze the runtime, showing it is $O(\log(m) + \log(n))$.
- If one algorithm is $O(\log(m+n))$, another is $O(\log(m) + \log(n))$, which is more efficient? Give your proof.
- Time `binary_search` and `ternary_search` on vectors of sizes $2^0, 2^1, ..., 2^{30}$. Be sure to do correct statistical data collection. Submit a statement of data collected, and declaration of which appears to be faster.

Assignment 2f, Due Jan 29

- 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.
- Chart the normalized runtimes of `binary_search` and `ternary_search`, along with $N^{\sqrt{2}}, N^{\sqrt{3}}, \text{LOG}_2(N), \text{LOG}_3(N)$ and 1. Submit the chart, and a statement discussing which algorithm is faster.

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 © 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

• At the beginning of class on the due dates, submit paper copies of your solutions.