Divide and Conquer

1. Use the divide-and-conquer integer multiplication algorithm to multiply the two binary integers 10011011 and 10111010.

2. Solve the following recurrence relations and give a Θ bound for each of them.

(a) $T(n) = 2T(n/3) + 1$

(b) $T(n) = 5T(n/4) + n$

(c) $T(n) = 7T(n/7) + n$

(d) $T(n) = 9T(n/3) + n^2$

(e) $T(n) = 8T(n/2) + n^3$

(f) $T(n) = 49T(n/25) + n^{3/2} \log n$

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

(h) $T(n) = T(n - 1) + n^c$, where $c \geq 1$ is a constant

(i) $T(n) = T(n - 1) + c^n$, where $c > 1$ is some constant

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

(k) $T(n) = T(\sqrt{n}) + 1$

3. Find the unique polynomial of degree 4 that takes on values $p(1) = 2$, $p(2) = 1$, $p(3) = 0$, $p(4) = 4$, and $p(5) = 0$. Write your answer in the coefficient representation.

4. How many lines, as a function of n (in Θ(·) form), does the following program print? Write a recurrence and solve it. You may assume n is a power of 2.

```
function f(n)
if n > 1:
    print_line(“still going”)
f(n/2)
f(n/2)
```

5. A binary tree is full if all of its vertices have either zero or two children. Let B_n denote the number of full binary trees with n vertices.

(a) By drawing out all full binary trees with 3, 5, or 7 vertices, determine the exact values of B_3, B_5, and B_7. Why have we left out even numbers of vertices, like B_4?

(b) For general n, derive a recurrence relation for B_n.

(c) Show by induction that B_n is Ω(2^n).

6. You are given an array of n elements, and you notice that some of the elements are duplicates; that is, they appear more than once in the array. Show how to remove all duplicates from the array in time O(nlogn).

7. Given a sorted array of distinct integers A[1, . . . , n], you want to find out whether there is an index i for which A[i] = i. Give a divide-and-conquer algorithm that runs in time O(log n).

8. A k-way merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

(a) Here's one strategy: Using the merge procedure , merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?

(b) Give a more efficient solution to this problem, using divide-and-conquer.

9. You are given two sorted lists of size m and n. Give an O(logm + logn) time algorithm for computing the kth smallest element in the union of the two lists.

10. An array A[1 . . . n] is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is A[i] > A[j]?". (Think of the array elements as GIF files, say.) However you can answer questions of the form: "is A[i] = A[j]?" in constant time.

(a) Show how to solve this problem in O(n log n) time. (Hint: Split the array A into two arrays A1 and A2 of half the size. Does knowing the majority elements of A1 and A2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

(b) Can you give a linear-time algorithm? (Hint: Here's another divide-and-conquer approach:

• Pair up the elements of A arbitrarily, to get n/2 pairs

• Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them

Show that after this procedure there are at most n/2 elements left, and that they have a majority element if and only if A does).

11. (a) Write down the pseudocode for quicksort.

(b) Show that its worst-case running time on an array of size n is Θ(n^2).

12. The Euclid’s algorithm is for computing the greatest common divisor (gcd) of two positive integers: the largest integer which divides them both.

Give an efficient divide-and-conquer algorithm for greatest common divisor. How does the efficiency of your algorithm compare to Euclid’s algorithm if a and b are n-bit integers? (In particular, since n might be large you cannot assume that basic arithmetic operations like addition take constant time.)

13. The square of a matrix A is its product with itself, AA.

(a) Show that five multiplications are sufficient to compute the square of a 2 × 2 matrix.

(b) What is wrong with the following algorithm for computing the square of an n × n matrix?

“Use a divide-and-conquer approach as in Strassen’s algorithm, except that instead of getting 7 subproblems of size n/2, we now get 5 subproblems of size n/2 thanks to part (a). Using the same analysis as in Strassen’s algorithm, we can conclude that the algorithm runs in time $O(n^{\log_2 5})$.”

(c) In fact, squaring matrices is no easier than matrix multiplication. In this part, you will show that if n × n matrices can be squared in time S(n) = O(n^c), then any two n × n matrices can be multiplied in time O(n^c).

i. Given two n × n matrices A and B, show that the matrix AB + BA can be computed in time 3S(n) + O(n^2).