Discrete Structures. CSCI-150. Fall 2015.

Homework 5. Due Wed. Oct 7, 2015.

Problem 1

Solve the linear recurrence (for $n \geq 0$)

$$f(0) = 1, \quad f(1) = -1,$$
$$f(n) = f(n-2).$$

Although this problem is not graded, it's easier than the other linear recurrences in this homework, so you are advised to do it first, before solving problems 2 and 3.

Problem 2 (Graded)

Solve the linear recurrence (for $n \geq 1$)

$$f(1) = 10, \quad f(2) = -2,$$
$$f(n) = f(n-1) + 12f(n-2).$$

Problem 3 (Graded)

First, verify that $x^3 - 3x - 2 = (x^2 + 2x + 1)(x - 2)$.

Then, solve the linear recurrence

$$f(0) = 0, \quad f(1) = 1, \quad f(2) = 7,$$
$$f(n) = 3f(n-2) + 2f(n-3).$$

Problem 4

Consider a rewriting operation that transforms a given bit string into a new bit string according the following two rules:

each **1** is replaced by **100**
each **0** is replaced by **1**

For example, if we apply this rewriting operation several times to the string "**1**", it will be transforming as follows:

$$\mathbf{1} \mapsto \mathbf{100} \mapsto \mathbf{10011} \mapsto \mathbf{10011100100} \mapsto \cdots$$

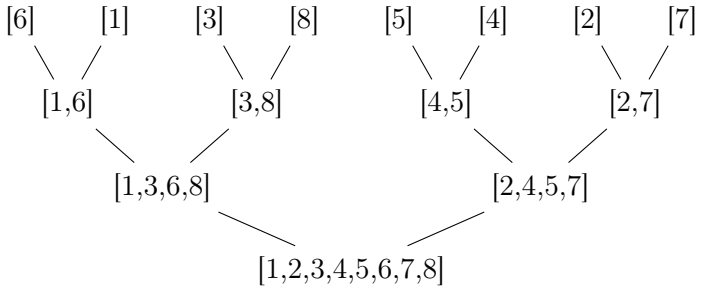
(a) Assume that the starting string is "**1**" as in the example above. Find a recurrent formula for **the number of 1s in the string after n rewrites**.

(b) Prove that the recurrence has the following closed-form solution: $\dfrac{2^{n+1} + (-1)^n}{3}$.

Problem 5. Bonus (Extra credit)

Conceptually, the merge sort algorithm consists of two stages:

1. In the first stage, the input list is broken down into small lists of unit length. It takes 0 comparisons, so this stage does not contribute anything to the total time complexity of the algorithm.
2. In the second stage, we merge the lists:

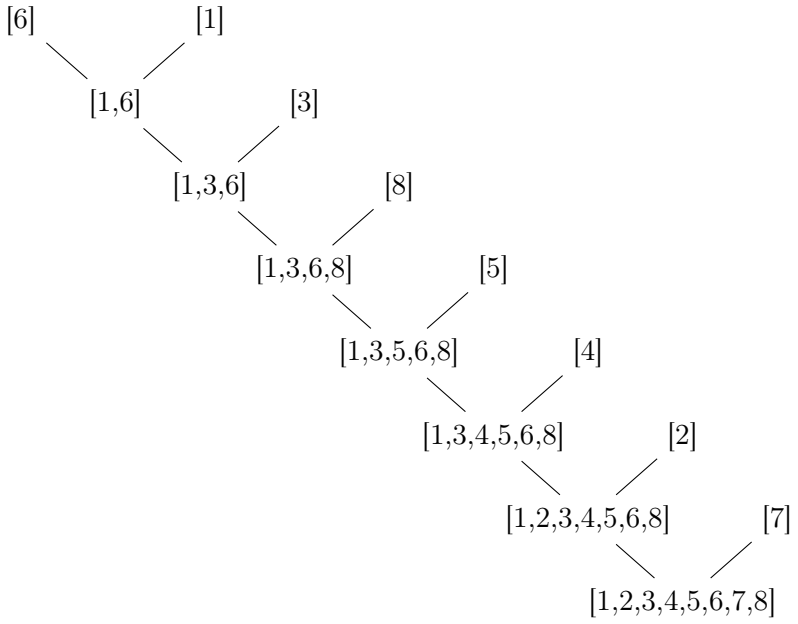


The time complexity of merging. When merging two lists l_1 and l_2 with the procedure we discussed in class, the number of comparisons is at most

$$\text{lenght}(l_1) + \text{length}(l_2) - 1.$$

Using this fact, we proved that, in the worst case, the merging stage takes $T(n) = n \log_2 n - n + 1$ comparisons.

Consider a similar sorting algorithm, call it INCREMENTAL SORT. It is using the same merge function, but the input is split differently. Because of that, on the input $[6, 1, 3, 8, 5, 4, 2, 7]$, the lists are merged in the following order:



(a) How would you implement INCREMENTAL SORT? We know that the merge function is the same as before. Try to define the splitting function. (You don't have to write actual code, explain how it works).

(b) What is the worst input for this algorithm? That is, suggest an input such that the algorithm makes the maximum number of comparisons.

(c) In the worst case, what is the number of comparisons for this algorithm?

(d) Discuss the results. Compare the new algorithm with the merge sort.