Discrete Structures. CSCI-150. Fall 2013.

Homework 6. Due Wed. Oct 16, 2013.

Problem 1

Solve linear recurrence

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

Problem 2

Solve linear recurrence

$$f(0) = 3, \quad f(1) = 1,$$
$$f(n) = 4f(n-1) + 21f(n-2).$$

Problem 3

Verify that

$$x^3 - 3x^2 + 4 = (x^2 - 4x + 4)(x + 1).$$

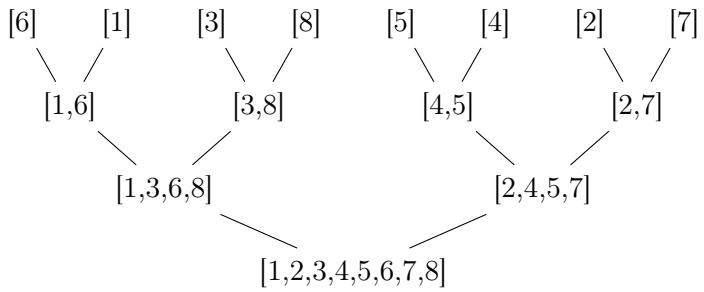
Solve linear recurrence

$$f(0) = 1, \quad f(1) = 0, \quad f(2) = 14,$$
$$f(n) = 3f(n-1) - 4f(n-3).$$

Problem 4

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 in terms of time complexity, we don't care about this stage.
2. In the second stage, we merge the lists:

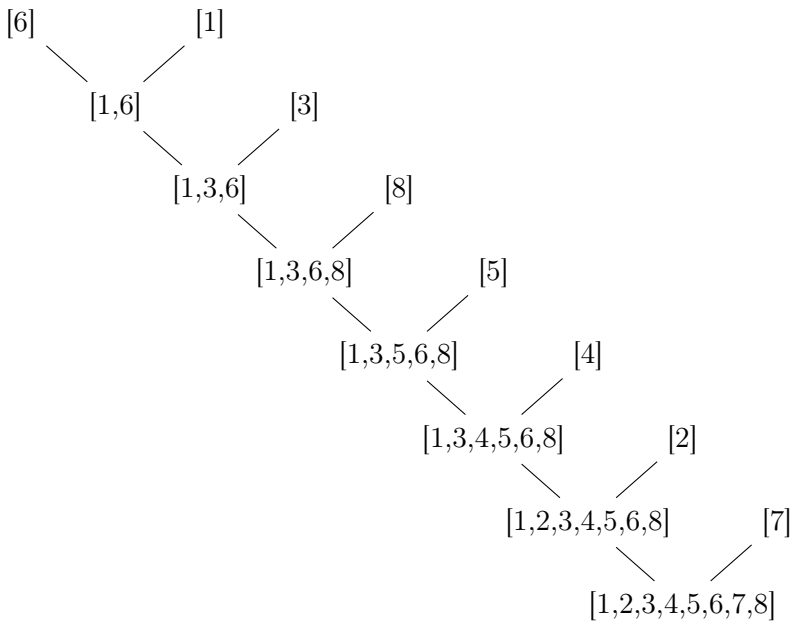


The time complexity of merging. When merging two lists l_1 and l_2 with the procedure we discussed in class, the number of comparisons is at most

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

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, and because of that the lists are merged in the following order:



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

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

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