1. Alternative Analysis of Sequential Binary Addition

In class we analyzed the following situation: given n numbers written in binary, each equal to 1, repeatedly add two of the numbers, keeping the result and discarding the originals, until, after m-1 additions, computing the total sum (m). The cost of adding two numbers, of b₁ and b₂ bits respectively, is min {b₁,b₂} plus the number of carry propagations past the left end of the smaller number. We showed that the total cost of all the additions, no matter what the order, is O(m). The purpose of this problem is to give an alternative, perhaps simpler, analysis of a generalization of the same situation.

a. Consider a collection of particles of four kinds: a, c, 0, 1, and with the following transition rules for the particles:

```
0 + 0 + a → 0+a
0 + 1 + a → 1+a
1 + 1 + a → 0+c
0 + c → 1
1 + c → 0+c
```

(E.g. “0 + 0 + a → 0+a” means that two 0’s and one a can be converted into one 0 and one a.)

Given an arbitrary collection of nₐ a’s, nₖ c’s, n₀ 0’s, and n₁ 1’s, show that after O(nₐ + nₖ + n₀ + n₁) transitions, no more transitions are possible.

Do this by considering a potential function of the form Aₐnₐ + Cₖnₖ + Zₙ₀ + Eₙ₁ for non-negative constants A, C, Z, E. Find A, C, Z, E such that the amortized cost of any transition is ≤ 0 (if the actual cost of a transition is defined to be 1).

Conclude that the maximum number of transitions is at most the initial potential minus the final potential. Since the latter is non-negative, the initial potential is an upper bound on the number of transitions.
b. Use the result of (a) to show that, if we begin with positive integers \(a_1, a_2, \ldots, a_m\) written in binary and repeatedly destructively add them together pairwise, until after \(m-1\) additions forming the sum \(a_1 + a_2 + \ldots + a_m\), then the total cost of all the additions is

\[
O(m + \sum_{i=1}^{m} \lg a_i).
\]

2. Redundant Binary Numbers

Suppose we represent positive integers in binary but allow the digit “2” as well as “1” and “0”. That is, a sequence of digits \(d_n d_{n-1} \ldots d_0\) with \(d_i \in \{0,1,2\}\) represents the number \(\sum_{i=0}^{n} d_i2^i\). A digit sequence is well-formed if the leftmost digit is non-zero; and, if we ignore 1 digits, digits 0 and 2 alternate in the sequence. Show how to represent such redundant binary numbers in a data structure so that adding 1 to such a number takes \(O(1)\) time in the worst case; and, more generally, adding two redundant binary numbers \(a_1\) and \(a_2\) takes \(O(1 + \lg \min \{a_1,a_2\})\) time. Hint: figure out how to make the sum well-formed.

3. Heap Initialization

Review the way implicit heaps work (CLRS sections 6.1 – 6.4) and in particular the heap initialization procedure BUILD-MAX-HEAP (CLRS p.133). CLRS proves that this procedure runs in \(O(n)\) time on an array of size \(n\) by doing a summation (CLRS pp. 133-134). Provide an alternative proof of this result by using an amortization argument like that used in class to analyze binary addition. Hint: assign a potential of \(O(\lg n)\) to a maximal subtree of \(n\) nodes that has already been arranged in heap order.

4. Multiflips in Binary Trees

The multiflip operation on a binary tree works as follows: mark every node on the rightmost path from the root to a leaf, swap the left and right subtrees of these nodes, then erase all the marks. The cost of the operation is the number of nodes along the rightmost path. Prove that the total cost of \(m\) multiflip operations on a tree on \(n\) nodes is \(\Theta(n + m\lg n)\). (You must prove both an upper and a lower bound here.)

5. Comparison of the linked-list representation and the forest representation of disjoint sets (CLRS sections 21.2 and 21.3).

Consider an arbitrary intermixed sequence of MAKE-SET, UNION, and FIND-SET operations, implemented using the linked-list representation(LL) or alternatively the forest representation (F). Suppose the same union heuristic is used with both methods. Prove that the running time of the F method is no greater than a constant factor times that of the LL method. What can you say about the usefulness of the LL method versus that of the F method?