Homework Assignment #2
Due: May 26, 2023 at 10:00 p.m.

1. Suppose we want to implement an ADT for a set of integers that supports the operations

- INSERT(x), which adds x to the set (you may assume, as a precondition, that x is not already in the set), and
- SEARCH(x), which returns true if x is in the set and false otherwise.

Let n be the number of elements in the set. If we stored the elements in a sorted array, the worst-case time per operation would be $\Theta(\log n)$ for SEARCH and $\Theta(n)$ for INSERT. Instead, we will store the elements in a linked list of arrays. Each element of the set will appear in exactly one of the arrays. Each array will be sorted in increasing order, but there is no constraint on how the elements of one array compare to the elements of another array. Each array will have size 2^k for some non-negative integer k and no two arrays in the list will have the same size. The arrays will be ordered in the list by their size (in increasing order).

[1] **(a)** Draw an example of the data structure when the set being represented contains 11 elements.

[2] **(b)** Use Θ notation to give a good upper bound on the number of arrays in the list as a function of n. Briefly justify your answer.

[2] **(c)** Use Θ notation to give a good bound on the worst-case time of the following implementation of SEARCH. Briefly justify your answer.

```
1: function SEARCH(x)
2:     for each array A in the list of arrays do
3:         if x is found by a binary search of A then return true
4:         end if
5:     end for
6:     return false
7: end function
```

[1] **(d)** Consider the following implementation of INSERT. The *next* field of a list element points to the successor of that element.

```
8: function INSERT(x)
9:     precondition: x is not already in the set
10:    create an array of size 1 containing x and insert it at the beginning of the list
11:    current ← first element in the list
12:    done ← false
13:    loop
14:        exit when done or current.next is null        ▷ we reached end of list or did line 20
15:        if current and current.next are arrays of the same size then
16:            merge arrays current and current.next into a new array new
17:            replace current and current.next in the list by new
18:            current ← new                             ▷ advance to new array for next iteration
19:        else
20:            done ← true
21:        end if
22:    end loop
23: end function
```

The merge on line 16 takes two sorted arrays of size 2^k and merges them into one sorted array of size 2^{k+1} in $\Theta(2^{k+1})$ time (see Section 2.3.1 of the textbook for a description of such a merge algorithm).

To help you understand how this INSERT algorithm behaves, it might help if you spend some time thinking about how it is similar to the INCREMENT operation on a binary counter.

Draw a picture of the state that results from inserting an element into the data structure you drew in part (a).

[2] **(e)** Suppose we do a sequence of m INSERT operations, starting from an empty set. Show that *one* INSERT in that sequence takes $\Omega(m)$ time.

[5] **(f)** Consider a sequence of m INSERT operations, starting from an empty set. Our goal is to show that the total time for the sequence of operations is $O(m \log m)$. The intuition is that each inserted element starts out in an array of size 2^0, and may have to be merged into a larger set at most $\log_2 m$ times, so when we add the element to the set, we give the element $\log_2 m$ units of potential, and later use up that potential to offset the cost of merging that element into larger arrays. If the element is currently in an array of size 2^k, it has used up k units of its potential, so it should still have $\log_2 m - k$ units left.

Thus, when the list contains arrays of sizes $2^{k_1}, 2^{k_2}, \ldots, 2^{k_\ell}$, define the value of the potential to be $\Phi = \sum_{i=1}^{\ell} 2^{k_i}(\log_2 m - k_i)$.

(i) Explain why Φ is always non-negative.

(ii) How does line 11 change the value of Φ?

(iii) How do lines 16–17 change the value of Φ?

(iv) Use Φ to show that the total time for the sequence of m INSERTS is $O(m \log m)$.