**8.7 Assignment 7, due Thursday 2018-04-12, at 11:00pm**

For this assignment, you are to implement an abstract data type representing an array that supports an efficient arrayCombine operation that returns the result of aggregating the first k elements of the array using a combine function provided when the array is first created.

You may assume that combine is an associative operation: combine(x, combine(y, z))should always equal combine(combine(x, y), z). But it is not necessarily commutative: combine(x, y) may or may not equal combine(y, x).

8.7.1 Interface

The interface to the array is given by the file array.h, shown below:

#ifndef \_ARRAY\_H

#define \_ARRAY\_H

*// Abstract array data type that allows quick computation of an*

*// aggregate over a prefix of the array.*

*//*

*// The array is provided with a combine function for combining*

*// values. This should represent some associative binary operation on ints.*

*//*

*// The arrayCombine function returns the result of aggregating*

*// the first k values in the array in order using combine.*

#include <stdlib.h>

**typedef** **struct** array Array;

*// Create a new array holding n values, all initially 0.*

*// Behavior is undefined if n == 0.*

*// Cost: O(n).*

Array \*arrayCreate(int (\*combine)(int, int), size\_t n);

*// Free all space used by an array.*

*// Cost: O(n).*

void arrayDestroy(Array \*);

*// Return the number of elements of an array.*

*// Cost: O(1).*

size\_t arraySize(const Array \*);

*// Return the i-th element of an array*

*// or 0 if i is out of range.*

*// Cost: O(log n).*

int arrayGet(const Array \*, size\_t i);

*// Set the i-th element of an array to v.*

*// No effect if i is out of range.*

*// Cost: O(log n).*

void arraySet(Array \*, size\_t i, int v);

*// Return the result of aggregating the first k elements*

*// of an array in order using the combine function.*

*// If k is zero or greater than size, returns combination of all elements.*

*// Cost: O(log n).*

int arrayCombine(const Array \*, size\_t k);

#endif

[examples/2018/hw/7/array.h](http://www.cs.yale.edu/homes/aspnes/classes/223/examples/2018/hw/7/array.h)

This file can also be found in /c/cs223/Hwk7/array.h on the Zoo.

8.7.2 Your task

Your task is to write a corresponding array.c file that implements the data structure and functions declared in array.h. You should stay within the stated asymptotic bounds for each function.

8.7.3 Combining elements

Most of the functions in array.h are self-explanatory. The tricky one is arrayCombine. For values of k in the range 1 through the size of the array, the arrayCombine routine should produce the same result as running the following function, where combine is the function previously passed to arrayCreate.

int

arrayCombineIterative(const Array \*self, size\_t k, int (\*combine)(int, int))

{

int accumulator = arrayGet(self, 0);

**for**(int i = 1; i < k; i++) {

accumulator = combine(accumulator, arrayGet(self, i));

}

**return** accumulator;

}

Note that this implementation runs in *O*(*n*) time, which is too much. Your implementation should ensure that arrayCombine runs in *O*(log*n*) time. You may find it helpful to store aggregate data computing the result of combining specific intervals of the array, to allow you to compute the aggregate for a given prefix by combining the values of some of these intervals. But be careful not to store aggregates for too many intervals, or you will spend too much time updating this information in arraySet.