Array Processing – Documentation

Csc 600-01

Keawa Rozet

Code available at: <https://github.com/krozet/Array-Processing.git>

The Problem

Array processing (elimination of three largest values) (one of many array reduction problems) The array a(1..n) contains arbitrary integers. Write a function reduce(a,n) that reduces the array a(1..n) by eliminating from it all values that are equal to three largest different integers. For example, if a=(9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9) then three largest different integers are 6,7,9 and after reduction the reduced array will be a=(1,1,1,2,3,3,5), n=7. The solution should have time complexity O(n).

Implementation

My method for reducing the array involves moving through the array and finding the three largest values. Then, I remove those values from the array and push all the values to the left. After this is done, I take the final size the array should be and copy its contents into a new array of this final size and print its contents to the screen. This algorithm is within O(n) complexity.

The portion of my method that takes the longest is where I have to remove each value from the array and shift the remaining elements over by one value to the left. This process can’t be helped while working with integer arrays, and I believe switching to using vectors instead would help improve performance here.

Since the problem asked for a simple and quick solution, I decided to go with the method down below. If I were asked to improve the algorithm’s performance, I would have gone with my original plan involving a pointer at each end of the array, both finding the three highest values, and once they pass each other begin removing those values from the array. I believe this would increase performance by a fair amount but not enough to sacrifice the simplicity the problem asked for.

Source Code

/\*

Keawa Rozet

CSc 600-01 - Programming Languages

Homework #2

Array Processing

\*/

#include <iostream>

#include <algorithm>

#include <stdio.h>

#include <cstring>

#include <vector>

int reduce(int\* array, int size);

int main(int argc, char const\*\* argv) {

int array[] = {9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9};

int size = sizeof(array)/sizeof(int);

//prints array before the reduce method

std::cout << "Before reduction: ";

for (int i = 0; i < size; i++)

std::cout << array[i] << " ";

//creates the new reduced array

int newSize = reduce(array, size);

int reducedArray[newSize];

std::copy(array, array + newSize, reducedArray);

//prints the new array after the reduce method

std::cout << "\nAfter reduction: ";

for (int i = 0; i < newSize; i++)

std::cout << reducedArray[i] << " ";

std::cout << "\n";

return 0;

}

int reduce(int\* array, int size) {

int min = 0;

int mid = 0;

int max = 0;

//finds min, mid, max

for (int i = 0; i < size; i++) {

if (array[i] > max) {

min = mid;

mid = max;

max = array[i];

}

else if (array[i] > mid && array[i] < max) {

min = mid;

mid = array[i];

}

else if (array[i] > min && array[i] < mid) {

min = array[i];

}

}

//removes min, mid, and max

for (int i = 0; i < size; i++) {

if (array[i] == min || array[i] == mid || array[i] == max) {

std::copy(array + i+1, array + size, array + i);

size -= 1;

i -= 1;

}

}

//returns the new size of the array

return size;

}

Example Build and Execution through Terminal

> g++ -o ap arr-process.cpp

> ./ap

Before reduction: 9 1 1 6 7 1 2 3 3 5 6 6 6 6 7 9

After reduction: 1 1 1 2 3 3 5