Chris Nielsen

ESE 344

A. Doboli

ESE 344 HW 1

QUESTION 1:

1a.
Homers algorithm in an O notation is a runtime of O(n), because it iterates from i=n down to , and is a n+1 iteration structure.

1b.
One implementation in pseudo code for a naive polynomial evaluation algorithm could be:

```
naive-eval-funct(A, n, x)
    p = 0
    for i = 0 to n
        term = 1
        for j = 1 to i
            term = term * x
        p = p + A[i] * term
    return p
```

For each i in the loop, the total computer time sums to

$$\sum_{i=0}^{n} O(i) = O(n^2)$$

For our comparison we can see that homers is the MORE efficient solution due to its runtime.

1c.
Let's look at the loop-invariant proof and prove the termination

I. At initialization we have the state of

$$p = \sum_{k=0}^{n-(i+1)} A[k+i+1]x^k$$

Where i=n , there are NO terms (n-(n+1)=-1
p=0
Invariant will hold in this case.

II. During maintenance we have
P = A[i] + x*(previous p)

Matches required form for following iteration
Invariant still holds.

III. During the end or termination of this algorithm
Loop ends when i = -1

Then we substitute for -1 and get
P(x) = Summation(n-(-1+1) k =0 A[k+(-1+1)] xk

= summation nk = 0 A [k] xk, and this is the polynomial we actually wanted

Horners does correctly evaluate what we want.

QUESTION 2:

Given the array [2; 3; 8; 6; 1]

2a.
The inversions for the above array would be for any pair of indecises where i<J and A[i] > A[j]

(1,5) because 2>1

(2,5) because 3>1

(3,4) because 8>6

$(3,5)$ because $8>1$

$(4,5)$ because $6>1$

2b.
The set of numbers amongst all sets Z, that has the highest amount of inversions is in fact the "reverse sorted array" in which as the indices increase you grow closer to 1, starting with N and ending in 1. Like such:

$\langle n, n-1, \ldots, 2, 1 \rangle$

Total inversions for this case will be :

$$(n-1)+n-2+(\ldots\ldots)+1=\frac{n(n-1)}{2}$$

2c.
There are 3 cases for this algorithm given 3 different extremes:

If the array is already sorted : it will run in O (n) time

If the array is the reverse order array : it will run in O(n^2) time.

Under normal conditions the array will run in a random amount of time and the performance will be affected determined by n+I,

2d.

The following code pseudo snippet could be an option for determining that performance start:

```
Function: int merge_then_count (A, p, q, r);

Count_num_inversions(A, p, r):
    if p >= r:
        return 0
    q = floor((p + r) / 2)
    leftInv  = Count_num_inversions(A, p, q)
    rightInv = Count_num_inversions(A, q + 1, r)
    splitInv = Merge_then_count(A, p, q, r)
    return leftInv + rightInv + splitInv

Merge_then_count(A, p, q, r):
    Let L = A[p ... q]
    Let R = A[q+1 ... r]
    i = 0, j = 0, inv = 0
    for k from p to r:
        if i >= length(L):
            A[k] = R[j]
            j = j + 1
        else if j >= length(R):
            A[k] = L[i]
            i = i + 1
        else if L[i] <= R[j]:
            A[k] = L[i]
            i = i + 1
        else:
            A[k] = R[j]
            inv = inv + (length(L) - i)
            j = j + 1
    return inv
```

The runtime for this deterministic algorithm will in fact be the same amount of runtime as a normal merge sort, since it is a modified Θ(nlogn) algorithm.

QUESTION 3:

3a: Coding Exercise

SEE CPP file attached:

Visual Studio COPY PASTE

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;

//relative sort function
vector<int> relativeSortArray(const vector<int>& arr1, const vector<int>& arr2) {
   unordered_map<int, int> freq;
   //frequency count loop to use for later
   for (int num : arr1) {
       freq[num]++;
   }

   vector<int> result;
   //using counts from before insert into a new array having been counted.
   for (int num : arr2) {
       if (freq.count(num)) {
           //insertion
           result.insert(result.end(), freq[num], num);
           //move onto next one and remove from addition queue
           freq.erase(num);
       }
   }

   //array 2 needs a secondry vector for holding
   vector<int> extras;
   for (auto pair : freq) {
       // Insert the value "pair.second" times.
       extras.insert(extras.end(), pair.second, pair.first);
   }
   // Sort the extras in ascending order with built in sort function
   sort(extras.begin(), extras.end());
```

```
   // Append extras
   result.insert(result.end(), extras.begin(), extras.end());
   return result;
}

int main() {
  cout<< " EXPECTED BEHAVIOR : Example 1: \n \n Input: arr1 =
[2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6] \n Output: [2,2,2,1,4,3,3,9,6,7,19] \n
Example 2: \n Input: arr1 = [28,6,22,8,44,17], arr2 = [22,28,8,6] \n Output:
[22,28,8,6,17,44]";


   // Example 1 run
   vector<int> arr1_1 = {2, 3, 1, 3, 2, 4, 6, 7, 9, 2, 19};
   vector<int> arr2_1 = {2, 1, 4, 3, 9, 6};
   vector<int> sorted1 = relativeSortArray(arr1_1, arr2_1);

   cout << "\n\nNOW FOR SORTING PROOFS: \n";


   cout << "Sorted arr1 for Example 1: ";
   for (int num : sorted1)
       cout << num << " ";
   cout << "\n";

   // Example 2 run
   vector<int> arr1_2 = {28, 6, 22, 8, 44, 17};
   vector<int> arr2_2 = {22, 28, 8, 6};
   vector<int> sorted2 = relativeSortArray(arr1_2, arr2_2);

   cout << "Sorted arr1 for Example 2: ";
   for (int num : sorted2)
       cout << num << " ";
   cout << "\n";

   return 0;
}
```


ESE344
Hw1_demo.cpp
sort.h
Hw1_demo.cpp > ...
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;
//relative sort function
vector<int> relativeSortArray(const vector<int>& arr1, const vector<int>& arr2) {
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh
(base) chrisnielsen@Chriss-MacBook-Pro ESE344 % g++ -std=c++17 /Users/chrisnielsen/Documents/ESE344/Hw1_demo.cpp -o Hw1_demo && ./Hw1_demo
EXPECTED BEHAVIOR : Example 1:
Input: arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]
Output: [2,2,2,1,4,3,3,9,6,7,19]
Example 2:
Input: arr1 = [28,6,22,8,44,17], arr2 = [22,28,8,6]
Output: [22,28,8,6,17,44]
NOW FOR SORTING PROOF:
Sorted arr1 for Example 1: 2 2 2 1 4 3 3 9 6 7 19
Sorted arr1 for Example 2: 22 28 8 6 17 44
(base) chrisnielsen@Chriss-MacBook-Pro ESE344 % g++ -std=c++17 /Users/chrisnielsen/Documents/ESE344/Hw1_demo.cpp -o Hw1_demo && ./Hw1_demo