Devin
Campbell
CS-216-02

CS216

HW5 (Based on Ch. 22: Developing Efficient Algorithms) Due: Dec. 7, 11:59 PM

Total points: 35

Submission instructions:

- Please submit one pdf file with solutions

1. (10 points) Design an *O(*n) algorithm for computing the sum of numbers from *n1* to *n2* for *(n1 < n2).* Can you design an *O(1)* for performing the same task?

```
Int sum =0;
for(int I = n1 ,
      i<=n2; i++){
      sum += i
}
```

Yes, summation of all whole numbers up to N is stated as $n(n^2+1)/2$.

If you subtract the summation of n1-1 from summation of n2, you will get the correct answer. Requiring no loops.

2. (10 points) Describe an algorithm for removing duplicates from an array. Analyze the complexity of the algorithm.

O(N^2)

Two nested for loops are required, first to designate the key, the second to iterate the array checking for duplicates and possibly deleting. Essentially a N^2 – n set.

3. (10 points) Describe an algorithm for finding the occurrence of the max element in an array. Analyze the complexity of the algorithm.

O(N)
Iterate the array, if max < array[i], max = array[i], continue.

4. (5 points) Analyze the following sorting

```
algorithm: for(int i=0; i<list.length;

i++){
        if (list[i] > list[i+1]){
                swap list[i] with
                list[i+1]; i = -1;
        }
}
```

O(N^2) – ascending order.
Worse case scenario, it's a n^2 – n algorithm

Go element by element, if the element ahead is greater, switch two values, go backwards, and check again.