Dillon Daudert

1. For the following pseudo-code, count the primitive operations considering the best and worst case. What conditions would lead to the best case? What conditions would lead to the worst case?

Algorithm BubbleSort(A):
Input: An array, A, of n comparable elements, indexed from 1 to n
Output: An ordering of A so that its elements are in nondecreasing order

```
for i ← 1 to n − 1 do                          — 3   asn, cmp, inc (subtraction)
  for j ← n down to i + 1 do                   — 3
    if A[j − 1] > A[j] then                    — 4   sub, index, index, cmp
      t ← A[j − 1]                             — 3
      A[j − 1] ← A[j]                          — 4
      A[j] ← t                                 — 2
return A                                       — 1   (i+=1; if i ≤ n-1)
```

a) Use operation counting to determing the running time in terms of input size n, $T(n)$, for the algorithm above:

$T(n)$: $\frac{17}{2}n^2 - \frac{3}{2}n - 3$ (worst case) — Occurs when the array is in reverse sorted order.

$T(n)$: $4n^2 + 3n - 3$ (best case) — occurs when array is already sorted.

$$T(n): \quad 3 + \sum_{i=1}^{n-1}\left[3 + (n-i)\left[4 + (3+4+2)+4\right] + 4\right] + 1$$

for — for — cmp — if inner — inc — return — inner loop — outer loop

b) What is the Big-Oh time complexity of this algorithm? Set up and solve the inequality to verify your answer.

BubbleSort is in $O(\underline{\quad n^2 \quad})$

$\frac{17}{2}n^2 - \frac{3}{2}n - 3 \leq cn^2$; let $c = \frac{17}{2}$,

$\frac{17}{2}n^2 - \frac{3}{2}n - 3 \leq \frac{17}{2}n^2$.

This inequality holds for all $n \geq 0$.

2. For the following pseudo-code, count the primitive operations considering the best and worst case. What conditions would lead to the best case? What conditions would lead to the worst case?

Algorithm LinearSearch(A, n, s):
Input: An array, A, of n integers, indexed from 0 to $n-1$ and an integer s which is the integer we are searching for
Output: The index of the first element that matches s, or -1 if no element matches s

```
for i ← 0 to n − 1 do
  if A[i] = s then
    return i
return -1
```

3
2
1
4 (total : compare i to n-1)

a) Use operation counting to determing the running time in terms of input size n, $T(n)$, for the algorithm above:

$T(n)$: $4+6n$ (worst case) - item is not in the list

$T(n)$: 6 (best case) - item is first in the list

$T(n) = 3 + n(2+4) + 1$

b) What is the Big-Oh time complexity of this algorithm? Set up and solve the inequality to verify your answer.

LinearSearch is in $O(n)$

$4+6n \leq cn$; let $c=7$

$4+6n \leq 7n$

$4 \leq n$

This inequality holds when $n \geq 4$.

3. The following pseudo-code swaps even indexed elements of an array with their consecutive odd indexed elements (e.g element at index 2 with element at index 3, element at index 4 with element at index 5 and so on). Count the primitive operations considering the best and worst case. What conditions would lead to the best case? What conditions would lead to the worst case?

Algorithm Swap(A,n)
Input: Array A containing integer numbers and integer n which shows the length of the array
Output: Array A which its even indexed elements are swapped with their consecutive odd indexed elements

```
1.  i ← 0                          ——— 1
2.  while (i < n)                  ——— 1
3.      temp ← A[i]                ——— 2
4.      A[i] ← A[i+1]              ——— 4
5.      A[i+1] ← temp              ——— 3
6.      i ← i+2                    ——— 2
7.  return A
```

a) Use operation counting to determing the running time in terms of input size n, $T(n)$, for the algorithm above:

$T(n)$: $3 + 11\frac{n}{2}$ (worst case)

$T(n)$: $3 + 11\frac{n}{2}$ (best case)

} these are the same because the algorithm always does the same operations, regardless of the input array's elements.

$T(n)$: $1 + 1 + \frac{n}{2}(2+4+3+2) + 1$

b) What is the Big-Oh time complexity of this algorithm? Set up and solve the inequality to verify your answer.

Swap is in $O(n)$

$\frac{11}{2}n + 3 \leq cn$; let $c = 6$

$\frac{11}{2}n + 3 \leq 6n$

$\frac{11}{2}n + 6 \leq 12n$

$6 \leq n$

this inequality holds for $n \geq 6$.

* this algorithm has an error when n is odd; the loop executes and attempts to access the $i+1$ element, but this will be out of range if $i = n-1$. *