CS 101, Assignment 1

Christopher Huynh, Oliver Rene

April 2016

1 Question 1

1.1

Let n = A.length
For i = 1 to n:
    For j = 0 to n-1:
        if A[j] $\geq$ A[j+1]:
            Swap A[j] and A[j+1]

1.2

Bubble Sort runs $\frac{n(n+1)}{2}$ times
Prove $\frac{n(n+1)}{2} = \theta(\text{n}^2)$
Need to show that there exists c_1, c_2, and n_0 such that

$$0 \leq c_1 \cdot n^2 \leq \frac{n(n+1)}{2} \leq c_2 \cdot n^2 \text{ for all } n \geq n_0$$

Let $\text{c}_1 = \frac{1}{2}$ and $\text{c}_2 = 1$ and $\text{n}_0 = 1$
Therefore, $\frac{n(n+1)}{2} = \theta(\text{n}^2)$
Bubble Sort time complexity is $\theta(\text{n}^2)$

1.3

Loop invariant: At the end of each iteration of i of the for loop, the subarray $A[(n-i)..n]$ consists of the largest i elements originally in A[1..n] except in sorted order.
Base case/Initialization: i =1, algorithm does nothing in the first loop. A[1] is trivially in sorted order.
Induction/Maintenance: Set $i > 2$. Assume loop invariant is true for i - 1. Thus, at the end of the (i - 1) loop (which is the beginning of the ith loop), A[(n - (i - 1))..n] is the sorted version of the largest (i-1) elements originally in A[1..n].

Observe that the index j tracks the position of the ith largest number. The j for loop effectively pushes the ith largest number to the $A[n-i]$ position of the sorted array $A[n-(i-1)..n]$. So A[(n-i)..n] is sorted. ■

2 Question 2

Hypothesis: Let n $\in$ N. For a set of n elements, there are 2^{n-1} subsets that have an odd number of elements.
Base case: n=1. set is 1. There should be $2^{(1)-1}$=1 subsets. There is only one subset of 1 which with an odd number of elements which is 1
Inductive Hypothesis: Let k $\in$ N. Let P(k) = 2^{k-1} be the statement that a set of k elements has 2^{k-1} subsets that have an odd number of elements.
Inductive Step: We need to prove that P(k+1) = $2^{(k+1)-1}$ or simply
P(k+1) = 2^k. Let there be a set A and an element a that is in set A. Now let A' = A - {a}, therefore A' has k elements. Lets split the subsets of A into two groups. Group X has subsets that do not have element a and Group Y has subsets that do have element a.

The subsets in Group A are exactly the subsets of A'. Because A' has k elements, the statement P(k) is true for A'. This means that Group A contains 2^{k-1} subsets that have an odd number of elements. Now, let there be a set B = A' + {a} - {}. The subsets in Group B are exactly the subsets of B. Because the empty set {} does not have element a, the empty set is removed from A'. Because an element is removed but the element a is included in its place, the total elements of set B is k. This means that Group B contains 2^{k-1} subsets that have an odd number of elements.

If we combine Group A and Group B, there are $2^{k-1} + 2^{k-1} = 2^k$ subsets that have an odd number of elements. Therefore, the statement P(k+1) = 2^k is true, completing the inductive step. $\blacksquare$

3 Question 3

Let $f(n) = a_0 + a_1 n + a_2 n^2 + ... + a_k n^k$ be a degree-k polynomial, where every $a_i > 0$.

3.1

Show that $f(n) \in \theta(n^k)$.
We have the definition that $f(n) = \theta(n^k) \implies c_1 \leq \lim_{n\to\infty} \frac{f(n)}{n^k} \leq c_2$, where c_1 and c_2 are positive non-zero constants.
If we find $\lim_{n\to\infty} \frac{f(n)}{n^k}$, we get $\lim_{n\to\infty} \frac{f(n)}{n^k} = \lim_{n\to\infty} \frac{a^k n^k}{n^k} = a^k$. Therefore, we can find some c_1 and c_2 such that $c_1 \leq a^k \leq c_2$.
Therefore, $f(n) \in \theta(n^k)$.

3.2

Show that $f(n) \notin O(n^{k'})$, for all $k' < k$.
Lets say that $f(n) \in O(n^{k'})$, for all $k' < k$.
We have the defintion that $f(n) = O(n^{k'}) \implies \lim_{n\to\infty} \frac{f(n)}{n^{k'}} \leq c_1$, where c_1 is a positive non-zero constant.

If we find $\lim_{n\to\infty} \frac{f(n)}{n^{k'}}$, we get $\lim_{n\to\infty} \frac{f(n)}{n^{k'}} = \lim_{n\to\infty} \frac{n^k}{n^{k'}} = \lim_{n\to\infty} \mathrm{n}^k = \infty$.
Because $\lim_{n\to\infty} \frac{f(n)}{n^{k'}} = \infty$, then there exists no possible c_1 such that $\lim_{n\to\infty} \frac{f(n)}{n^{k'}} \leq c_1$.
Therefore, by proof by contradiction, $f(n) \notin O(n^{k'})$, for all $k' < k$.

4 Question 4

4.1

Prove $\log_2 n = O(\sqrt{n})$
Need to show that there exists c_1 and n_0 such that

$$0 \leq \log_2 n \leq c_1 \cdot \sqrt{n}, \text{ for all } n \geq n_0$$

Let c_1=1 and $n_0 = 16$.
Therefore, $\log_2 n = O(\sqrt{n})$.

4.2

Show that $\log_2 n \notin \Omega(\sqrt{n})$.
Lets say $\log_2 n \in \Omega(\sqrt{n})$.
We have the definition that $\log_2 n = \Omega(\sqrt{n}) \implies \lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} \geq c_1$, where c_1 is a positive non-zero constant.
If we find $\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}}$, then we get $\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = 0$.
Because $\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = 0$, we cannot find any positive non-zero c_1, such that $\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} \geq c_1$.
Therefore, by proof of contradiction, $\log_2 n \notin \Omega(\sqrt{n})$.

5 Question 5

Suppose the input array A is in sorted order, *except* for k elements. In other words, there are n - k elements of A that are already in sorted order, and the remaining k elements are out of order.

Prove that Insertion-Sort on A runs in $O(nk)$ time.
The pseudocode of Insertion-Sort is as follows:

```
n = A.length
for i = 1 to n
    j = i
    while j > 1 and A[j − 1] > A[j]
        Swap A[j − 1] and A[j]
        j = j - 1
```

When Insertion-Sort is performed on A, Insertoin-Sort from $A[1..(n-k)]$ runs $n-k$ times because the while loop is never entered. Insertion-Sort from $A[(n-k)..n]$ then runs at most $\frac{k(n-1)}{2}$ times. If we add these too run times together, we get $n + \frac{k(n-1)}{2} = \frac{2n+nk-k}{2}$. Therefore, Insertion-Sort on A runs at most $\frac{2n+nk-k}{2}$ times. Now to prove that Insertion-Sort on A runs in *O(nk)* time, we need to show that $\lim_{n\to\infty} \frac{\frac{2n+nk-k}{2}}{nk} \leq c$, where c is a positive non-zero constant. If we find $\lim_{n\to\infty} \frac{\frac{2n+nk-k}{2}}{nk}$ we get $\lim_{n\to\infty} \frac{\frac{2n+nk-k}{2}}{nk} = \lim_{n\to\infty} \frac{2n+nk-k}{2nk} = \lim_{n\to\infty} \frac{n(2+k)-k}{2nk} = \lim_{n\to\infty} \frac{n(2+k)}{2nk} = \frac{2+k}{2k}$. Because $\lim_{n\to\infty} \frac{\frac{2n+nk-k}{2}}{nk} = \frac{2+k}{2k}$, we can find a c so that $\lim_{n\to\infty} \frac{\frac{2n+nk-k}{2}}{nk} \leq c$. Therefore, $\frac{2n+nk-k}{2} = O(nk)$. Therefore, Insertion-Sort on A runs in $O(nk)$ time.