Homework One

Robert George Phillips (rogphill@ucsc.edu)

January 21, 2018

1

Question: A common algorithm for sorting is Bubble-Sort. Consider an input array $A = [A[1], A[2], ..., A[n]]$, with n elements. You repeat the following process n times: for every i from 1 to $n-1$, if $A[i]$ and $A[i+1]$ are out of order, you swap them. Write this out as pseudocode. Do a time complexity analysis of Bubble-Sort. Prove that the time complexity is $\Theta(n^2)$. Prove that the algorithm works (i.e. that the final array is sorted) by using loop invariants.

1.1 Pseudocode

```
A = {A[1], A[2], ..., A[n]}
i = 1
j = 1
swapper = 0

for i to n-1 {
    for j to n-1 {
        if A[j] > A[j+1] {
            swapper = A[j+1]
            A[j+1] = A[j]
            A[j]= swapper
        }
    }
}
```

1.2 Time Complexity Analysis

Bubble sort will run n times in the inner loop, such that $1 \leq j \leq n-1$ for each time the outer loop iterates. Thus, bubble sort is $O(n * n = n^2)$.

$F(n) \in O(n^2)$

$G(n) \in O(n^2)$

$\lim_{n\to\infty} \frac{F(n)}{G(n)} = \frac{n^2}{n^2} = 1$

Since 1 is a constant, $F(n) \in O(n^2)$ and $F(n) \in \Omega(n^2)$.

Thus, $F(n) \in \Theta(n^2)$.

1.3 Proof by Induction

Invariant: After k iterations of the outer loop, the last k elements are the largest. Furthermore, the last k elements are a permutation of the original array A and are sorted in ascending order.

Base case ($k = 1$): Vacuously true. After one iteration of the outer loop, the last element in array A is the largest element in array A.

Induction: Suppose the invariant is true for k. Let us prove the invariant for $k+1$:

By induction, at the end of k outer loops, everything in $A[(n-k+1), \ldots, n]$ are the largest k elements in array A, and are sorted in ascending order.

When the $(k+1)^{\text{st}}$ iteration of the outer loop ends, the last $(k+1)$ elements are the largest elements in array A, in sorted order, such that $A[(n-k)] \leq A[(n-k+1), \ldots, n]$.

Therefore, $A[(n-k+1), \ldots, n]$ is sorted.

■

2

Question: Use induction to prove the following statement: The number of subsets of $\{1, 2, \dots, n\}$ having an odd number of elements is 2^{n-1}. Clearly state your hypothesis, base case and inductive step.

2.1 Hypothesis

There are 2^{n-1} subsets from the set $\{1, 2, \dots, n\}$ that have an odd number of elements.

2.2 Proof by Induction

Base case $(n = 1)$: Vacuously true. $2^{n-1} = 2^{1-1} = 2^0 = 1$. There is only one subset that is odd, $\{1\}$.

Induction: Suppose a set of n elements has 2^{n-1} subsets with an odd number of elements. Let us prove that a set of $n+1$ elements has 2^{n-1} subsets with an odd number of elements:

$2^{(n+1)-1} = 2^n$

$2^{n-1} \cup (n+1) = 2^n$

Since we have a set of size n, the total number of subsets is 2^n. Thus, there must be an equal number of even subsets and odd subsets such that $2^{n-1} + 2^{n-1} = 2^n$.

Therefore, the set $\{1, 2, \dots, n, n+1\}$ has 2^{n-1} subsets with an odd number of elements and thus all sets have 2^{n-1} subsets with an odd number of elements.

■

3

Question: Let $f(n) = a_0 + a_1n + a_2n^2 + \ldots + a_kn^k$ be a degree-k polynomial, where every $a_i > 0$. Show that $f(n) \in \Theta(n^k)$. Furthermore, show that $f(n) \notin O(n^{k\prime})$, for all $k\prime < k$.

3.1 Asymptotic Analysis

Let us take the highest degree polynomial in $F(n) = a_0+a_1n+a_2n^2+\ldots+a_kn^k$ where $a_i > 0$, and classify it as $O(n^k)$. Thus,

$F(n) = a_0 + a_1n + a_2n^2 + \ldots + a_kn^k = O(n^k)$

$G(n) = n^k = O(n^k)$

$\lim_{n\to\infty} \frac{F(n)}{G(n)} = \frac{a_0+a_1n+a_2n^2+\ldots+a_kn^k}{n^k} = 1$

Since 1 is a constant, $F(n) \in O(n^k)$ and $F(n) \in \Omega(n^k)$. Thus, $F(n) \in \Theta(n^k)$.

Furthermore, $\forall k\prime < k$, $\lim_{n\to\infty} \frac{a_0+a_1n+a_2n^2+\ldots+a_kn^k}{n^{k}\prime} = \infty$.

Therefore, $F(n) \geq G(n) \to a_0 + a_1n + a_2n^2 + \ldots + a_kn^k \geq n^{k}\prime$.

Hence, $a_0 + a_1n + a_2n^2 + \ldots + a_kn^k \in \Omega(n^{k\prime})$ but $\notin O(n^{k\prime})$ and thusly $\notin \Theta(n^{k\prime})$.

$\blacksquare$

4

Question: Prove that $\log_2 n = O(n^{1/3})$, but $\log_2 n$ is not in $\Omega(n^{1/3})$. Is $\log_2 n = \Theta(n^{1/3})$? Why or why not?

4.1 Asymptotic Analysis

By logarithmic properties and asymptotic analysis, $log_2(n) \in O(log(n))$. Thus,

$F(n) = log_2(n) = O(log(n))$

$G(n) = n^{1/3} = O(n^{1/3})$

$\lim_{n\to\infty} \frac{F(n)}{G(n)} = \frac{log(n)}{n^{1/3}} = 0$

Since the limit of $\frac{log_2(n)}{n^{1/3}}$ is 0, then $\frac{log_2(n)}{n^{1/3}} < \infty$ and thus,

$$log_2(n) < n^{1/3}$$

.

Hence, $F(n) \in O(G(n)) \to log_2(n) \in O(n^{1/3})$.

Thus, it is not possible for $log_2(n)$ to be $\Omega(n^{1/3})$ as $F(n) \ngeq G(n)$, and therefore $log_2(n) \notin \Theta(n^{1/3})$.

■

5

Question: 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.

5.1 Pseudocode

```
A = {A[0], A[1], ..., A[n]}
i = 1
swapper = 0
```

```
for i to n-1 {
    j = i
    while j > 0 && A[j-1] > A[j] {
        swapper = A[j-1]
        A[j-1] = A[j]
        A[j] = swapper
        j--
    }
}
```

5.2 Proof

For insertion sort, after k iterations the outer loop, the first $(k+1)$ elements in array A are in sorted ascending order. Furthermore, the inner loop only iterates when $A[k-1] > A[k]$ and continues to iterate until $A = \{A[0], A[1], \ldots, A[k]\}$ is sorted in ascending order.

Therefore, the inner loop runs at most k times, and the outer loop runs n times. Using asymptotic analysis, $A \in O(nk)$.

$\blacksquare$