CMPS101, Spring 2016
HW #1

Names: Tarum Fraz, Mark Ruhal
Sources: TA's, sections, book, lectures

Q(1): A common sort algorithm for sorting is Bubblesort. Consider an array (input) 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

```
for i = 1 to n-1
  for j = 0 to n-1-i
    if A[j] > A[j+1]
      K = A[j]            // initialize k
      A[j] = A[j+1]       // swap values
      A[j+1] = K          // swap values
```

- Do a time complexity analysis of Bubble-sort. Prove that the time complexity is $\theta(n^2)$.

By analyzing the pseudocode above, Bubble sort will only complete one iteration of n elements, along with no swaps when the array A is already sorted. This is also called the best case possible. The time complexity of the best case is $\theta(n)$. The worst case possible occurs when the array is sorted, but in opposite order. Bubble sort will iterate through n elements, then n-1 elements, and so on till one comparison occurs. So, the time complexity (worst case) of Bubble sort is:

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

- Prove the algorithm works (i.e) the final array is sorted, by using loop invariants.

Loop invarient: After every iteration of i, A[A.length], or the last element in A, will contain the current maximum.

For example, let's say that A[i+1] to A[n] is sorted. Then, for the current i, it is noted that A[i] can change, but it will keep incrementing untill it reaches the end. once it reaches the end, i to n will be sorted.

Q(3) This problem will help you get some practice with induction proofs. Use induction to prove the following statement:

The number of subsets of $\{1, 2, \ldots, n\}$ having odd number of elements is 2^{n-1}.

Clearly state your hypothesis, base case, and inductive step.

Inductive hypothesis: 2^{k-1}

Base case: $n = 1$ $\quad 2^{1-1} = 2^0 = 1$ => one subset w/ one element, which is the null set. since this set has an odd # of elements, we have proven P(1).

Inductive step: our goal is to prove P(n) → P(n+1) by assuming P(n) is true. we can divide P(n+1) into 2 groups, a subset that does not contain (n+1), and a group that does. There is a 1 to 1 correspondence between these subsets. For every subset in group 1 called (s), we can obtain a corresponding subset called (t) in group 2, by adding (n+1). This means $t = s \cup \{n+1\}$, and that many of the subsets that are in group 2, are in group 1.
By IH: 2^{k-1} even subsets and 2^{k-1} odd subsets in group 1. And for every odd subset in group 1, there is a matching even subset in group 2, which you can obtain by adding (n+1) to the odd set. we have 2^{k-1} odd subsets in group 2. By summing it up, we have 2^k subsets in the set $\{1, \ldots, n+1\}$. ☑

a13) Let $f(n) = a_0 + a_1 n + a_2 n^2 + \dots + a_k n^k$ be a degree-k polynomial, where every $a_i > 0$.

Show that $f(n) \in \Theta(n^k)$:

- we need to show that there exists a c_1, c_2, and n_0 such that

$$0 \le c_1 f(n) \le T(n) \le c_2 \cdot f(n) \quad \text{for all } n \ge n_0$$

rewrite: $0 \le c_1 n^k \le a_0 + a_1 n + a_2 n^2 + \dots + a_k n^k \le c_2 n^k$ for all $n \ge n_0$

% by n^k: $0 \le \frac{c_1 n^k}{n^k} \le \frac{a_0}{n^k} + \frac{a_1 n}{n^k} + \frac{a_2 n^2}{n^k} + \frac{a_k n^k}{n^k} \le \frac{c_2 n^k}{n^k}$

so, as $n \to \infty$, $\frac{a_0}{n^k} + \frac{a_1 n}{n^k} + \frac{a_2 n^2}{n^k} + a_k \to a_k$

Let $c_1 = a_k - 1$

$c_2 = a_k + 1$

$n_0 = n_k$

Show that $f(n) \in O(n^{k'})$, for all $k' < k$:

- we can show this by a proof by contradiction
- By definition of $O(n^{k'})$,

$$0 \le a_0 + a_1 n + a_2 n^2 + \dots + a_{k'} n^{k'} \le c \cdot n^{k'} \quad \text{for all } n \ge n_0$$

% by $n^{k'}$: $0 \le \frac{a_0}{n^{k'}} + \frac{a_1 n}{n^{k'}} + \frac{a_2 n^2}{n^{k'}} + \dots + \frac{a_{k'} n^{k'}}{n^{k'}} \le \frac{c \cdot n^{k'}}{n^{k'}}$

$$0 \le \frac{a_0}{n^{k'}} + \frac{a_1 n}{n^{k'}} + \frac{a_2 n^2}{n^{k'}} + a_{k'} \le c$$

As $n_{k'} \to \infty$, $\frac{a_0}{n^{k'}} + \frac{a_n}{n^{k'}} + \frac{a_2 n^2}{n^{k'}} + a^k \to \infty$

which is not $\le c$

Q(4) Prove that $\log_2 n = O(\sqrt{n})$, but $\log_2 n$ is not in $\Omega(\sqrt{n})$

To show $\log_2 n = O(\sqrt{n})$:

Need to show that there exists a c and n_0 such that

$$0 \le \log_2 n \le c \cdot \sqrt{n} \quad \text{for all } n \ge n_0$$

Let $c = 2$, $n_0 = 1$ $\Rightarrow$ rewrite $0 \le \log_2 n \le c \cdot \sqrt{n}$

$0 \le 0 \le 2$

we have shown $\log_2 n = O(\sqrt{n})$

To show $\log_2 n$ is not in $\Omega(\sqrt{n})$:

we can prove this using a proof by contradiction

By definition of $\Omega(\sqrt{n})$: $0 \le cf(n) \le T(n)$ for all $n \ge n_0$.

$0 \le c \cdot \sqrt{n} \le \log_2 n$

Divide by $\sqrt{n}$: $0 \le \frac{c \cdot \sqrt{n}}{\sqrt{n}} \le \frac{\log_2 n}{\sqrt{n}}$

rewrite: $0 \le c \le \frac{\log(n)}{\sqrt{n}\log(2)}$

As $n \to \infty$, $\frac{\log(n)}{\sqrt{n}\log(2)} \to 0$. By definition of omega notation, our c has to be positive. So, our proof by contradiction shows that $\log_2 n$ is not $\Omega(\sqrt{n})$.

Q(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

- An array of length n that is not sorted, which is the worst case takes $O(n^2)$ time. If our array has n-k elements that are sorted, k elements need to be sorted

$$\frac{k}{n} \quad \begin{matrix} // \text{ need to be sorted} \\ // \text{ total \# of elements} \end{matrix}$$

$$\left(\frac{k}{n}\right) O(n^2) = O(kn)$$

- We have shown insertion sort is an adaptive sort, and runs in $O(kn)$ time on an array that is partially sorted with k elements not in order out of n elements.