Spring 2019 CS 372 Homework Assignment 1 Solutions

1. In class we discussed Insertion sort and computed its running time.

```
INSERTION-SORT(A, n)
for j = 2 to n
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

Bubble sort is another sorting algorithm with the following pseudocode:

```
BUBBLE-SORT(A,n)
  for i=1 to n
        for j=n downto i+1
                if  A[j] < A[j-1] then exchange A[j] ↔ A[j-1]
```

(a) Compute running time of Bubble sort. Use cost and times columns to compute the running time. (Cost of a line is the amount of time that is required to execute the line, e.g. constant c_1. Times is the number of times the line is executed.) After you get the exact expression for the running time, represent it in big-Oh notation (O-notation).

Answer:

BUBBLE-SORT(A,n)	cost	times
for i=1 to n	c_1	n+1
for j=n downto i+1	c_2	n + (n-1)+…+2+1
if A[j] < A[j-1] then exchange A[j] ↔ A[j-1]	c_3	(n-1)+(n-2)+…+1+0

Total: $c_1(n+1) + c_2(n+(n-1)+\ldots+2+1) + c_3((n-1)+(n-2)+\ldots+1+0)$
$= c_1(n+1) + c_2\,(n+1)n/2 + c_3\,n(n-1)/2 = \tfrac{1}{2}\,(c_2+c_3)\,n^2 + (c_1+ \tfrac{1}{2}\,c_2 - \tfrac{1}{2}\,c_3)\,n + c_1 = O(n^2)$

(b) Which one do you think is faster in practice, INSERTION-SORT or BUBBLE_SORT, and why?

Answer: Regardless of the input array A, the inner loop in Bubble sort scans elements from A[n] all the way down to A[1], then from A[n] all the way down to A[2], then from A[n] down to A[3], ... , from A[n] down to A[n]. Therefore, the amount of work done by BUBBLE-SORT is $\Theta(n+ (n-1) + (n-2) + \ldots+ 1)$ which is equal to $\Theta(1+2+\ldots+(n-1)+n)$. This is the worst-case scenario of Insertion sort where in every iteration the maximum amount of shifting is done. Therefore, practically, Insertion sort performs better than Bubble sort.

2. Explain clearly why the statement "The running time of algorithm A is at most $\Omega(n^3)$" does not make sense.

Answer: Ω notation is a lower bound notation. Therefore, $\Omega(n^3)$ by itself means "at least cn^3". Therefore, the statement "The running time of algorithm A is at most at least cn^3" does not make sense.

3. Let $f(n) = 4n^5 – 2n^4 - 6n^2 + 7$ and $g(n) = 2n^5$. Prove that $f(n) = O(g(n))$ using the definition of Big-O notation. (You need to find constants c and n_0).

Answer: There are many ways to prove it. Two of them are listed below.
One way:

$$f(n) = 4n^5 - 2n^4 - 6n^2 + 7 \le 4n^5 + 7 \overset{(if\ n\ge 1)}{\le} 4n^5 + 7n^5 = 11n^5 = 5.5 * 2n^4$$
$$= 5.5 * g(n)$$

so c = 5.5, and $n_0 = 1$.

Another way:

$$f(n) = 4n^5 - 2n^4 - 6n^2 + 7 \le 4n^5 + 7 < 4n^5 + 2^5 \overset{(if\ n\ge 2)}{\le} 4n^5 + n^5 = 5n^5$$
$$= 2.5 * 2n^5 = 2.5 * g(n)$$

so c = 2.5, and $n_0 = 2$.

4. Order the following 16 functions by asymptotic growth rate from lowest to highest. If any are of the same order then circle them on your list.

$n+2\log n$, 7, $5n+n^2$, $3\log n$, $\log n^2$, $n^{1/3}$, $2n^2-5+n^4$,
$3n+n\log n$, $7n^2$, 2^n, 3^n, 3^{n+1}, $n!$, $\sqrt{n}$, $n^{2.01}$, $4^{\log_2 n}$.

Note: When comparing two functions f(n) and g(n) you may use $\lim_{n\to\infty} (f(n) / g(n))$ to compare their asymptotic growth rates.

Answer:

$7 = \Theta(1)$ has the lowest growth rate.
$3\log n$ and $\log n^2$ have the same growth rate: $3\log n = \Theta(\log n)$, $\log n^2 = 2\log n = \Theta(\log n)$.
$n^{1/3}$,
$\sqrt{n}$,
$n+2\log n = \Theta(n)$,
$3n+ n\log n = \Theta(n\log n)$,
$4^{\log_2 n}$ $(4^{\log_2 n} = n^{\log_2 4} = n^2)$, $5n+n^2$, and $7n^2$ have the same growth rate: $\Theta(n^2)$,
$n^{2.01}$,
$2n^2-5n+n^4$,
2^n,
3^n and 3^{n+1} have the same growth rate: $3^{n+1}=3*3^n = \Theta(3^n)$,
$n!$