STAT 598G Spring 2011
Sergey Kirshner
Homework 1 Solution (Written Exercises)

Sergey Kirshner

January 24, 2011

1. **(5 pts)** Arrange the following functions in increasing order of rate of growth. If some functions have the same rate of growth, identify them.

   (a) n

   (b) $\sqrt{n}$

   (c) $n \log n$

   (d) $n!$

   (e) $n \log \log n$

   (f) $n^{\log n}$

   (g) n^2

   (h) $n/\log n$

   (i) 2^n

   (j) n^n

   (k) 42

   (l) $1234567n^2$

   Solution: With an abuse of notation, we will say $f(n) = g(n)$ if $f(n) \in \Theta(n)$, and $f(n) < g(n)$ if $f(n) \in o(g(n))$. Since $f(n) \in o(g(n))$ and $g(n) \in o(h(n))$ imply $f(n) \in o(h(n))$ for asymptotically positive functions f, g, and h (transitivity), it is enough to show which function is "next" in the order. We will show the following:

   $$42 < \sqrt{n} < n/\log n < n < n \log \log n < n \log n < n^2 = 1234567n^2 < n^{\log n} < 2^n < n! < n^n.$$

   Using the definition of Θ and o, we can prove the following expressions.

   $42 \in o(\sqrt{n})$: For any $c > 0$, pick $n_0 = \frac{42^2}{c^2}$.

   $\sqrt{n} \in o(n/\log n)$: $\lim_{n\to\infty} \frac{\sqrt{n}}{n/\log n} = \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} = \lim_{n\to\infty} \frac{1/n}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0$. Thus for any $c > 0$, there exists n_0 such that for all $n > n_0$, $\frac{\sqrt{n}}{n/\log n} < c$ or equivalently, $\sqrt{n} < cn/\log n$.

   $n/\log n \in o(n)$: For any $c > 0$, set $n_0 = e^{1/c}$. Then for any $n > n_0$, $n/\log n < cn$.

   $n \in o(n \log \log n)$: For any $c > 0$, set $n_0 = e^{e^{1/c}}$. Then for any $n > n_0$, $n < cn \log \log n$.

$n \log \log n \in o(n \log n)$**:** $\lim\limits_{n\to\infty} \frac{n \log \log n}{n \log n} = \lim\limits_{n\to\infty} \frac{\log \log n}{\log n} = \lim\limits_{n\to\infty} \frac{\frac{1}{n \log n}}{\frac{1}{n}} = \lim\limits_{n\to\infty} 1/\log n = 0$. Thus for any $c > 0$, there exists n_0 such that for all $n > n_0$, $\frac{n \log \log n}{n \log n} < c$ or equivalently, $n \log \log n < cn \log n$.

$n \log n \in o\left(n^2\right)$**:** $\lim\limits_{n\to\infty} \frac{n \log n}{n^2} = \lim\limits_{n\to\infty} \frac{\log n}{n} = \lim\limits_{n\to\infty} 1/n = 0$. Thus for any $c > 0$, there exists n_0 such that for all $n > n_0$, $\frac{n \log n}{n^2} < c$ or equivalently, $n \log n < cn^2$.

$1234567n^2 \in \Theta\left(n^2\right)$**:** $c_1 = 1234567$, $c_2 = 1234567$.

$n^2 \in o\left(n^{\log n}\right)$**:** $\lim\limits_{n\to\infty} \frac{n^2}{n^{\log n}} = \lim\limits_{n\to\infty} n^{2-\log n} = 0$. Thus for any $c > 0$, there exists n_0 such that for all $n > n_0$, $\frac{n^2}{n^{\log n}} < c$ or equivalently, $n^2 < cn^{\log n}$.

$n^{\log n} \in o(2^n)$**:** For any $c > 0$, we need to find n_0 such that $n > n_0$, $\frac{n^{\log n}}{2^n} < c$. Note that $\frac{n^{\log n}}{2^n} = \left(\frac{n^{n/\log n}}{2}\right)^n$. Let $h(n) = n^{n/\log n}$. We will show that for $n > e$, $h(n) < e^{1/e} \approx 1.4447 < 1.5$. Then for $n > e$, $\frac{h(n)}{2} < 0.75$. By setting $n_0 = \max\left(e, c \log \frac{4}{3}\right)$, we get $\left(\frac{n^{n/\log n}}{2}\right)^n < \left(\frac{e^{1/e}}{2}\right)^n < 0.75^n < c$. To show that $h(n) < e^{1/e}$ for $n > e$, $h'(n) = n^{\log n/n-1} (\log n/n) \left(\frac{1}{n^2} - \frac{logn}{n^2}\right) = n^{\log n/n-4} \log n (1 - \log n)$. Note that for $n > e$, $h'(n) < 0$ (as $1 < \log n$), so for $n > e$, $h(n)$ is strictly decreasing and thus $h(n) < h(e) = e^{\log e/e} = e^{1/e}$.

$2^n \in o(n!)$**:** Since $n! > 2^{n+1}$ for all $n \geq 4$, one can choose $n_0 = \max\left(\frac{1}{c}, 5\right)$, so $cn! = cn(n-1)! \geq (n-1)! > 2^n$.

$n! \in o(n^n)$**:** Note that for any $n > 2$, $n \in \mathbb{N}$, $\frac{n!}{n^n} = \frac{1 \times 2 \times \cdots \times n}{\underbrace{n \times n \times \ldots n}_{n \text{ times}}} < \frac{1}{n}$, so by setting $n_0 = \lceil \frac{1}{c} \rceil$, for any $n > n_0$, $n! < n$.

2. **(1 pt)** Rigorously prove from basic definition that for asymptotically non-negative functions $f(n)$ and $g(n)$, $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

   Solution: To show that $\max(f(n), g(n)) \in \Theta(f(n) + g(n))$ we need to find $c_1, c_2 > 0$ and $n_0 > 0$ such that for all $n > n_0$, $c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n))$. $f(n) \leq \max(f(n), g(n))$ and $g(n) \leq \max(f(n), g(n))$. Set $c_1 = 0.5$, then for all $n \in \mathbb{R}$, $0.5(f(n) + g(n)) \leq 0.5 \times 2 \max(f(n), g(n)) = \max(f(n), g(n))$. On the other hand, for $f(n) \geq 0$ and $g(n) \geq 0$, $\max(f(n), g(n)) \leq f(n) + g(n)$, so set $c_2 = 1$. Asymptotically non-negative f and g means that there exists n_x and n_y such that for all $n > n_x$, $f(n) \geq 0$, and for all $n > n_y$, $g(n) \geq 0$. Pick $n_0 = \max(n_x, n_y)$; thus for all $n > n_0$, $f(n) \geq 0$ and $g(n) \geq 0$. Then for all $n > n_0$, $c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n))$.

3. **(2 pts)** Consider the pseudocode for another sorting algorithm (Algorithm 1, bubble sort).

   Why is the algorithm correct? What is its computational worst-case complexity (Θ)? What would constitute the worst case for this algorithm?

Algorithm 1 Bubble Sort

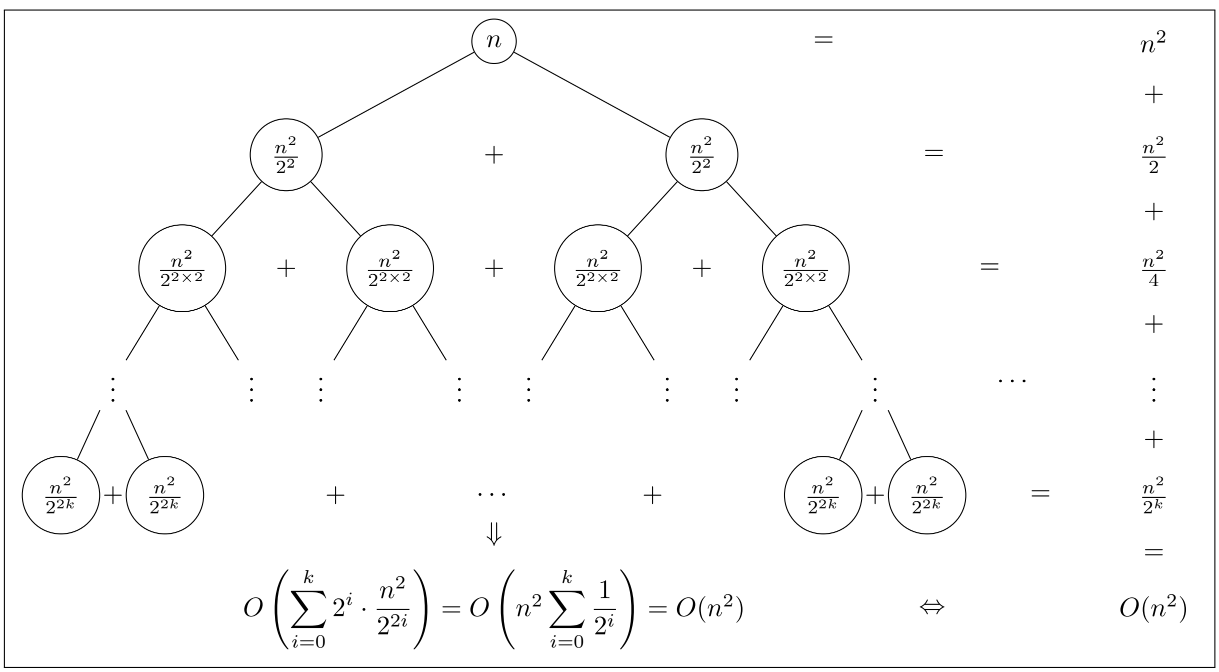
```
1:  function BubbleSort(A)
2:      INPUTS: n-element array A[1, ..., n]
3:      OUTPUT: sorted array A[1, ..., n]
4:      for i = 1, ..., n − 1 do
5:          for j = 1, ..., n − i do
6:              if A[j + 1] < A[j] then
7:                  swap A[j + 1] and A[j]
8:              end if
9:          end for
10:     end for
11:     return A
12: end function
```

Solution: The algorithm is correct because at the end of i-th iteration of the outer loop lines (4-10), element $A[n-i+1]$ will contain the i-th largest element of the array. The iteration i of the outer loop (lines 4-10) contains $n-i$ comparisons (and at worst the same number of swaps) in the inner loop (lines 5-9). So the worst case complexity is of the order of $(n-1)+(n-2)+\cdots+1=\frac{1}{2}n(n-1)\in\Theta\left(n^2\right)$. The worst case would correspond to starting with an array sorted in decreasing order. In this case each comparison on line (6) would produce a swap on line (7) as the i-th largest element would "bubble-up" from the very bottom to the its position at each iteration.

4. **(2 pt)** Write the recursion tree for the following recurrences. Use the tree to find the complexity of $T(n)$ (Θ).

   (a) $T(n)=2T(n/2)+n^2$,

   Solution:



   (b) $T(n)=3T(n/3)+n$.

   Solution: See Figure 1.

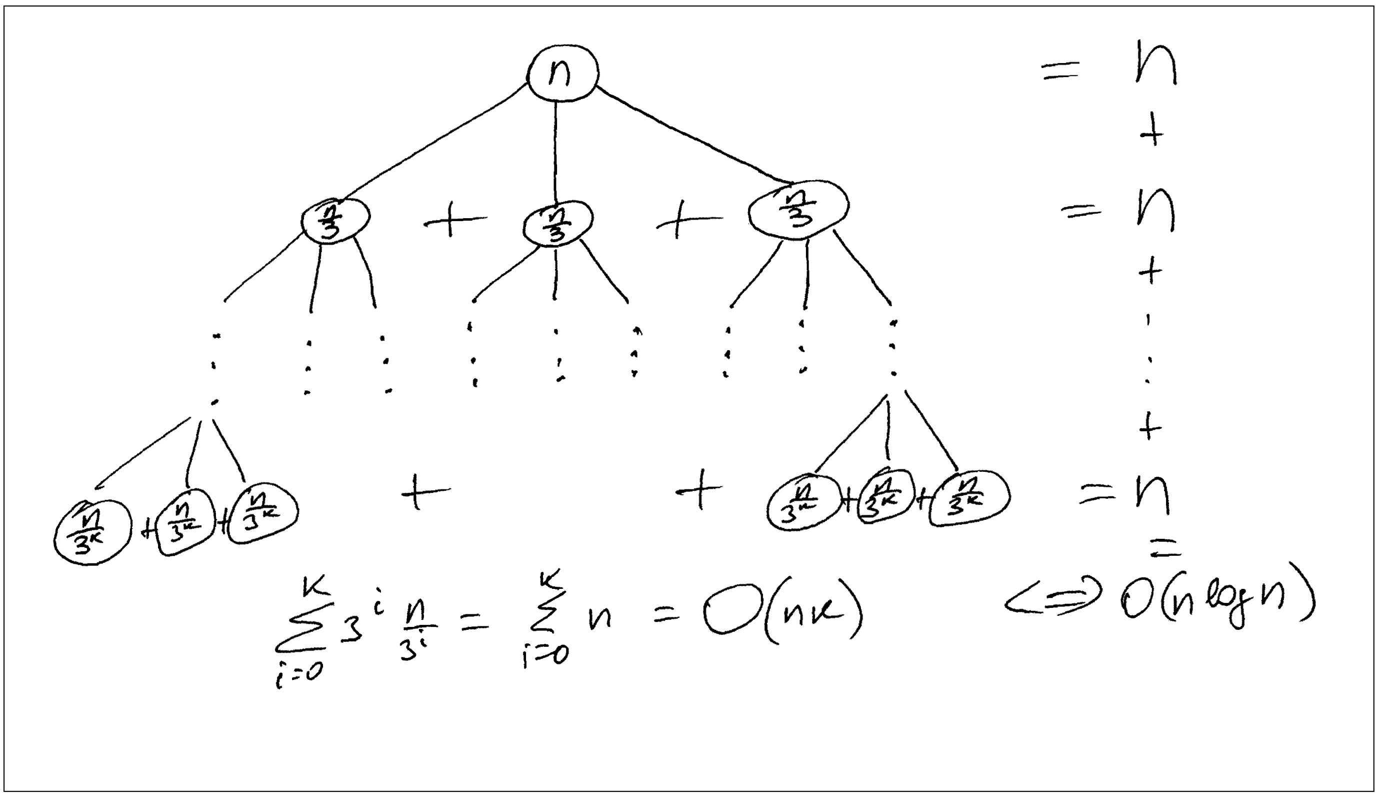


Figure 1: Solution to Problem 5b.