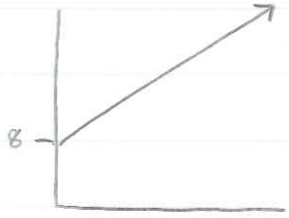
HW 2

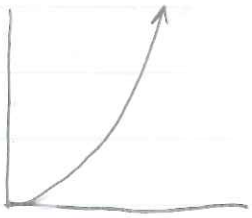
Nick Palumbo

3.1 $8n$

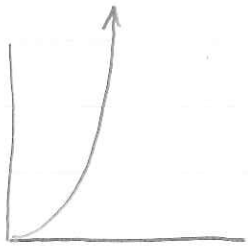


$4n \log n$

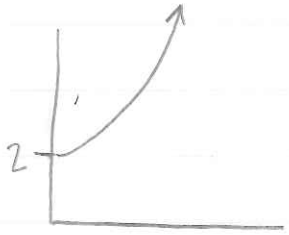
$2n^2$



n^3



2^n



3.2 $A = 8n \log n$ $\quad 8n \log n = 2n^2$

$B = 2n^2$ $\quad 4n \log n = n^2$

$4 \log n = n$ $\quad 4 = n / \log n$

$16 / \log 216 = 16/4 = 4$

$n_0 = 17$

3.8 2^{10}, $2^{\log n}$, $3n + 100 \log n$, $4n$, $n \log n$, $4n \log n + 2n$, $n^2 + 10n$, $n^2 + 10n$, n^3, 2^n

3.10 If $d(n)$ is $O(f(n))$, then

$d(n) \leq c_1 f(n) \quad n \geq n_1$

If $e(n)$ is $O(g(n))$, then

$e(n) \leq (2g(n)) \quad n \geq n_2$

Thus $d(n)e(n) \leq c_1 f(n) e(n)$

$c_1 f(n) e(n) \leq c_1 f(n) c_2 g(n)$

Thus $d(n)\, e(n) \leq c_1 c_2 f(n) g(n)$

$d(n)e(n)$ is $O(f(n) \cdot g(n)) \quad n \geq \max[n_1, n_2]$

3.14 $2 \max(f(n), g(n)) \geq (g(n) + f(n))$

$\max(f(n), g(n)) \geq \frac{1}{2}(g(n) + f(n))$

$n \geq n_0$

Thus $\max\{f(n), g(n)\} = \Omega(g(n) + f(n))$

where $c = \frac{1}{2}$

3.17 $(n+1)^5$

$n^5 + 5n^4 + n^3 + 10n^2 + 5n + 1$

$n^5 + 5n^4 + 0n^3 + 10n^2 + 5n + 1n^0 \leq n^5 + 5n^5 + 0n^5 + 10n^5 + 5n^5 + 1n^5$

Thus $1n^5 + 5n^4 + 10n^2 + 5n + 11 <= 22n^5$

$c = 22$ for any $n >= 1$

3.18 $2^{n+1} \in O(2^n)$

$2^{n+1} = 2^n \cdot 2^1 = 2 \cdot 2^n$

Thus $c = 2 \quad n_0 = 1$

3.20 Show that n^2 is $\Omega(n \log n)$

$n^2 \in \Omega(n \log n)$

n	n^2	$n \log n$
1	1	0
2	4	2
3	9	4.74
4	16	8
5	32	11.5

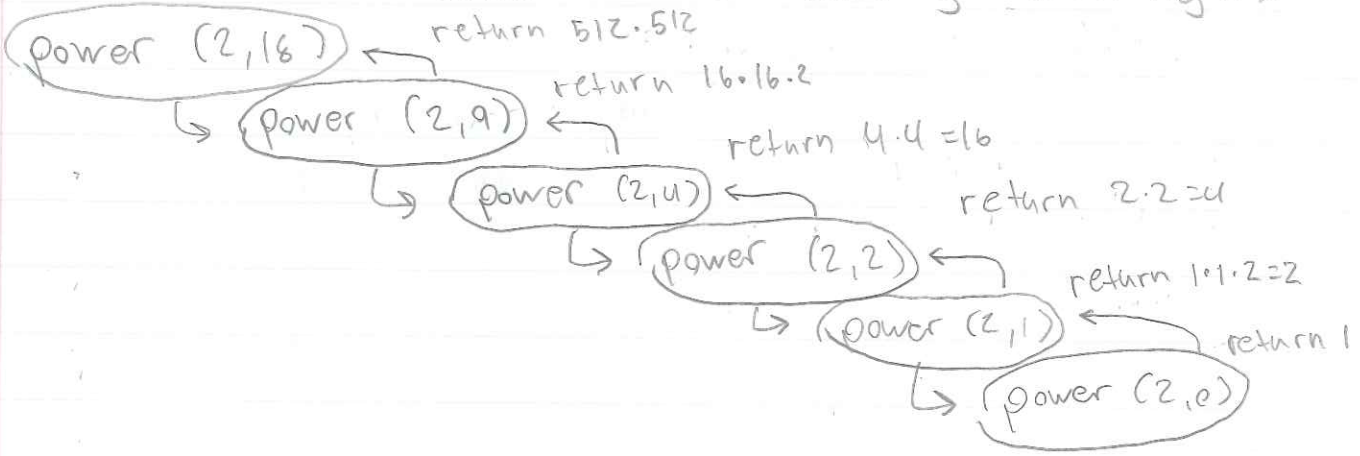
$n_0 = 1$

$c = 1$ Therefore n^2 is $(n \log n)$

3.27 There are two nested loops in an outer loop so $O(n^3)$

3.35 Combine all 3 sets and sort. That takes $O(n \log n)$. then traverse the list and find 3 matching elements consecutively. That's $O(n)$ n traversal would be canceled because it is a lower order term leaving $O(n \log n)$

4.3



4.7

```
def strToNum(strNum):
    if len(strNum) == 1:
        return int(strNum)
    else:
        new_digit = int(num[len(num)-1])
        return new_digit + 10*(strToNum(:len(num)-1])
```

The base case is a string of length 1. This should be the first number of the string (1 3531).

The recursive case caste the last number of the string. It returns the number plus 10 * a recursive call on the string except the last number.

1 + 10 (strNum (1353)
1 + 10 (3 + 10 (strNum (135)
1 + 10 (3 + 10 (5 + 10 (strNum (131)
1 + 10 (3 + 10 (5 + 10 (3 + strNum (str Num (1))))
1 + 10 (3 + 10 (5 + 10 (3 + 10 · 1)))
13531