CPSC HW1 Written Questions

Benjamin McDonnough

1. $1/n$, 2^{100}, $\log\log n$, $\sqrt{\log n}$, $\log^2 n$, $2^{\log n}$, $5n$, $n\log_4 n$, $6n\log n$, $2n\log^2 n$, $n^{0.01}$, $\lceil\sqrt{n}\rceil$, $3n^{.5}$, $4n^{3/2}$, $n^2\log n$, $4^{\log n}$, n^3, 2^n, 4^n, 2^{2^n}

Key: $\_ = O(\log n)$, $= \; = O(n)$, $\equiv \; = (n\log n)$, $\sim \; = O(n^2)$, $\approx \; = O(n^3)$, $\approxeq \; = O(2^n)$

2.

	1 Second	1 Hour	1 Month	1 Century
$\log n$	$\approx 10^{300000}$ (2^{10^6})	$2^{36\cdot 10^8}$	$2^{2592\cdot 10^9}$	$2^{31536\times 10^{11}}$
$\sqrt{n}$	10^{12}	1296×10^{16}	6718464×10^{18}	$99582\cdot 10^{26}$
n	10^6	3600×10^6	2592×10^9	31536×10^{11}
$n\log n$	62746	133378058	71870856404	68654697441062
n^2	10^3	60×10^3	50×10^6	172×10^8
n^3	10^2	1532	13736	146677
2^n	19	31	41	44
$n!$	9	12	15	17

3. Set a temp var, currentStart = 0. In the first loop, if M[t-1] + numbers[t] > 0 then

M[t] ← M[t-1] + numbers[t]

else M[t] ← numbers[t]

currentStart ← t

In the second for loop, check to see if M[t] is greater than m. If yes, then j = currentStart and k = t.

4. Algorithm MaxsubFaster(A):

Input: An n-element array A of numbers
Output: The maximum subarray sum of array A

```
M ← 0.0
current_sum ← 0
for i ← 0 to n do
    current_sum ← current_sum + A[i]
    if current_sum > M then
        M ← current_sum
    if current_sum < 0 then
        current_sum ← 0
return M
```

5. $T(i) = \begin{cases} \Theta(i) & \text{multiple of } 3 \\ \Theta(1) & \text{otherwise} \end{cases}$

$$\sum_{i \text{ is a multiple of } 3} T(i) = \sum_{k=1}^{\lfloor n/3 \rfloor} \Theta(3k) = 3\sum_{k=1}^{\lfloor n/3 \rfloor} \Theta(k) = 3\cdot\Theta\left(\frac{n}{3}\cdot\frac{n}{6}\right) = \Theta\left(\frac{n^2}{6}\right) = \Theta(n^2)$$

$$\sum_{i \text{ not multiple of } 3} T(i) = \sum_{i=1}^{n} \Theta(1) = \Theta\left(\frac{2n}{3}\right) = \Theta(n)$$

Amortized Running Time $= \frac{T(n)}{n} = \frac{\Theta(n^2)}{n} = \Theta(n)$

6. This scenario is possible because $O(n^2)$ grows faster than $O(n\log n)$, meaning for larger numbers, $O(n\log n)$ is faster than $O(n^2)$.

An example of this would be 2 sorting algorithms: A: $100n\log n$ ($O(n\log n)$) and B: n^2 ($O(n^2)$). At $n=10$, $A \approx 1000$ operations while $B = 100$ operations.