Nelson Dai V00815253
CSC 225 Assignment 1
~~May~~ 17/2016

1. $6n\log n \to O(n\log n)$ , $2^{100} \to O(1)$ , $\log\log n \to O(\log\log n)$, $\log^2 n \to O(\log^2 n)$
$2^{\log n} \to O(n)$ , $2^{2^n} \to O(2^{2^n})$, $\sqrt{n} \to O(\sqrt{n})$ , $n^{0.01} \to O(n^{0.01})$ , $1/n \to O(1/n)$
$4n^{3/2} \to O(n^{\frac{3}{2}})$, $3n^{0.5} \to O(\sqrt{n})$ , $5n \to O(n)$ , $2n\log^2 n \to O(n\log^2 n)$, $2^n \to O(2^n)$
$n\log_4 n \to O(n\log_4 n)$ , $4^n \to O(4^n)$ , $n^3 \to O(n^3)$ , $n^2\log n \to O(n^2\log n)$, $4^{\log n} \to O(n^2)$
$\sqrt{\log n} \to O(\sqrt{\log n})$

$1/n \le 2^{100} \le \log\log n \le \sqrt{\log n} \le \log^2 n \le n^{0.01} \le \boxed{\sqrt{n} = 3n^{0.5}} \le \boxed{2^{\log n} = 5n} \le \boxed{n\log_4 n = 6n\log n}$
$\le 2n\log^2 n \le 4n^{3/2} \le 4^{\log n} \le n^2\log n \le n^3 \le 2^n \le 4^n \le 2^{2^n}$

2. since $f(n)$ is the highest rate of increase of $d(n)$ and $g(n)$ is the highest rate of increase of $e(n)$. $d(n) \le k f(n)$ for $n \ge N$, $k \in N$ and $e(n) \le L g(n)$ for $n \ge M$, $L \in N$
$d(n)\cdot e(n) \le k f(n) \cdot L g(n) \to d(n)\cdot e(n) \le kL(f(n)g(n))$ for $n \ge \max(N, M)$ let $c = kL$
then $d(n)e(n) \le c(f(n)g(n))$ $\therefore$ $d(n)e(n)$ is $O(f(n)g(n))$

3. From the properties of logarithms we know that $\log_b f(n) = \frac{\log f(n)}{\log b}$ and $\log_2 f(n) = \frac{\log f(n)}{\log 2}$. if $b > 1$ is a constant then $\log b$ is a constant so the big-oh of $\log_b f(n)$ is $O(\log f(n))$. on the other hand. since $\log 2$ is a constant so the big-oh of $\log_2 f(n)$ is also $O(\log f(n))$ which indicates that $\log_b f(n)$ is $\Theta(\log_2 f(n))$ when $b > 1$ is a constant.

4. base : $T(1) = 1$, $2^{0+1} - 1 = 1$, $T(2) = 1+2 = 3$, $2^2 - 1 = 3$.
IH : $T(n) = 2^{n+1} - 1$ when $n = k$ for some $k \ge 1$, $k \in N$
IS : we want $T(k+1) = 2^{k+2} - 1$

$$\begin{aligned} T(k+1) &= T(k) + 2^{k+1} \\ &= 2^{k+1} - 1 + 2^{k+1} \\ &= 2(2^{k+1}) - 1 \\ &= 2^{k+2} - 1 \to \text{what we want} \end{aligned}$$

conclusion: so by induction, $T(n) = 2^{n+1} - 1$

5. best case : 1A + 1C + 1A 1C + 1 return = 4
worse case : 1A + n + n + n + 1 return + 1C = 3n + 3

b) arrayFind return index if element is found -1 otherwish

let $S_k$ be current item at index k after the $k^{th}$ iteration

ie: $S_k$ = Array[k].

Base: show $S_0$ is true.

2 cases: 1. $X = S_0 = A[0]$ then it returns index 0. → true.

2. $X \neq S_0, A[0]$ → 2 cases: ① Array has next item then index increase by 1 and compare again → true

② Array only has 1 element then return -1 → true

IH: Assume $S_{i-1}$ is true.

ie: arrayFind (X; A) A has i items

Is: show $S_i$ is true, ie after $i^{th}$ iteration arrayFind return index i or go to next ($S_{i+1}$) or return -1.

3 cases: 1. $A[i] = X$ → return index i true.

2. $A[i] \neq X$ and array A has next item → compare(A[i+1], X) true

3. $A[i] \neq X$ and array A doesn't have next item → return -1 true

∴ $S_i$ is true.