Com Sci 311 HW #1

Jacob Boicken

$f \in O(g)$ iff f is dominated by g

1.)

a.) $6n^2 - 41n + 2 \quad \in O(n^2)$

$6n^2 - 41n + 2 \leq 6n^2 + 41n^2 + 2n^2 = 49n^2$

$\therefore \exists c = 49, \exists n_0 \geq 1$
such that $6n^2 - 41n + 2 \leq c \cdot n^2$
for $\forall n \geq n_0$

so $6n^2 - 41n + 2 \in O(n^2)$

b.) $\forall a \geq 1 : 2^n \in O(2^{n-a})$

Prove: $\exists c \geq 1, \exists n_0 \geq 1$ such that $\forall n \geq n_0$
for $\forall a \geq 1$

$2^n < c \cdot 2^{n-a} = c \cdot 2^n \cdot 2^{-a}$

$\frac{2^n}{2^n \cdot 2^{-a}} \leq c \qquad 2^a \leq c$

→ meaning there is a constant c greater than 2^a for all possible a-values (which can be used in each case)

$\therefore \exists c = 2^a, \exists n_0 = 1$, such that $2^n \leq c \cdot 2^{n-a}$
for $\forall n \geq n_0$ and $\forall a \geq 1$

So $2^n \in O(2^{n-a})$

Com Sci 311 HW #1

Jacob Boicken

4.)

c) $O(\log_2 n) \in O(\log_a n)$ for $\forall a > 1$

→ this means any function upper bounded by $\log_2 n$ is upper bounded by $\log_a n$ for $\forall a > 1$

So we only need to find if $\log_a n$ is upper bounded by $\log_2 n$

→ $\log_2 n \in O(\log_a n)$ for $\forall a > 1$

Fact: $\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$

Prove:

$\log_2 n \leq C \cdot \log_a n$

$\frac{\log_2(n)}{\log_a(n)} \leq C$

$\log_2(a) \leq C$

→ ∴ for every value of a there exists a constant C, $\exists C = \log_2(a)$ $\exists n_0 = 1$, such that $\log_2 n \leq C \cdot \log_a n$ for $\forall n \geq n_0$

∴ $O(\log_2 n) \in O(\log_a n)$

10)

d) $\forall a > 1, \quad a^{a^{n+1}} \in O(a^{a^n})$

Disprove say $a = 2$

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

→ $\exists c \geq 1, \quad \exists n_0 \geq 1$ such that $\forall n \geq n_0$

$$2^{2^{n+1}} \leq c \cdot 2^{2^n}$$

$$2^{2^{n+1} - 2^n} \leq c \qquad \lim_{n \to \infty} 2^{2^{n+1} - 2^n} = \infty$$

→ which means c must be greater than

$$2^{2^{n+1} - 2^n}$$

for all positive n

→ which is not possible

so $2^{2^{n+1}} \notin O(2^{2^n})$

thus $a^{a^{n+1}} \notin O(a^{a^n})$

1.)

e.) If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$,

then $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

$f_1 \in O(g_1) \rightarrow \exists c_1 \geq 1, n_1 \geq 1$ such $f_1(n) \leq c_1 \cdot g_1(n)$

$f_2 \in O(g_2) \rightarrow \exists c_2 \geq 1, n_2 \geq 1$ such $f_2(n) \leq c_2 \cdot g_2(n)$

Therefore $\exists c_1, c_2, n_1, n_2$ such that when

$n \geq \max(n_1, n_2)$

$\rightarrow f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n)$

$\leq \max(c_1, c_2) g_1(n) + \max(c_1, c_2) g_2(n)$

$= \max(c_1, c_2)(g_1(n) + g_2(n))$

$\therefore \exists c = \max(c_1, c_2), n_0 = \max(n_1, n_2)$
such that $\forall n \geq n_0$

$f_1(n) + f_2(n) \leq c \cdot (g_1(n) + g_2(n))$

so $f_1 + f_2 \in O(g_1 + g_2)$

2.)

a.)

```
for i=1, i<n-8, i++        - Σ_{i=1}^{n-9}
    for j=i, j<i+8, j++    - Σ_{j=i}^{i+7}
        exe                - c
```

$$f(n) = \sum_{i=1}^{n-9} \sum_{j=i}^{i+7} c = c \sum_{i=1}^{n-9} \sum_{j=i}^{i+7} 1$$

- 8 times: $i, i+1, i+2, i+3, i+4, i+5, i+6, i+7$

$$= c \sum_{i=1}^{n-9} (8)$$

$$\boxed{= c \cdot 8 \cdot (n-9)}$$

$$= 8cn - c \cdot 72 \leq 8cn + c72n = 80 \cdot cn$$

$$\boxed{\in O(n)}$$

2.)

b.)

for i in range 1..=n $\quad \sum_{i=1}^{n}$

for j in range i..=n $\quad \sum_{j=i}^{n}$

for k in range 1..=j-1 $\quad \sum_{k=1}^{j-1}$

exe $\quad -c$

$$f(n) = c \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{j-1} 1$$

$$= c \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (j-1) = c \sum_{i=1}^{n} \left(\frac{n(n+1)}{2} - n \right)$$

$$= c \cdot n \cdot \left(\frac{n(n+1)}{2} - n \right)$$

$$= \frac{cn^3 + cn^2}{2} - cn^2$$

$$= \frac{cn^3 - cn^2}{2} = \boxed{\frac{c(n^3 - n^2)}{2}} \leq \frac{1}{2}(cn^3 + \frac{1}{2}c^3$$

$$= cn^3$$

$$\boxed{\in O(n^3)}$$

2.)

c.)

```
x = pow(2,n)
i = 1

while i <= x
    for j in range 1..=i
        exe

    i = i * 2
```

→ for $i=1,\ i<=n^2,\ i*=2$

for j in range $1..=i$ — exe — c — $\sum_{j=1}^{i}$

# of starts	i value	# of interal loops
1	1	1
2	2	2
3	4	4
4	8	8
5	16	16
⋮	⋮	⋮
k	2^{k-1}	2^{k-1}

$i = 2^{k-1}$ at end

so $2^{k-1} = 2^n$

$k-1 = n$

$k = n+1$

Summation:

$c\cdot(2^0 + 2^1 + 2^2 \cdots 2^{n-1}) = c\cdot\frac{2^k-1}{2-1}$

$= c(2^k-1)$

→ $c(2^k-1) = \boxed{c(2^{n+1}-1)} \leq 2c2^n + c2^n$

$= 3c2^n$

$\boxed{\in O(2^n)}$

3.)
```
myf(int a, int n)
int a1;

while n>=0
    if n==0 return 1
    a1 = myf(a, n/2)
    if n is even
        return a1*a1
    else
        return a*a1*a1

    n=n/2
```

- While loop is meaningless
  program will return before looping because of if else

- Only recursion seems to matter which goes
  - $n \to n/2 \to n/4 \to n/8 \dots n/2^{k+1}$
  - Internal execution of myf is C_1
  - Final operation when $n==0$ is C_2

So only count until $n/2^{k-2}$ ends at 1 then add C_2

$n/2^{k-2} = 1 \quad n = 2^{k-2} \quad \log n = k-2$

$k = \log n + 2$

→ $\boxed{C_1(\log n + 2) + C_2}$ $\boxed{\in O(\log n)}$