Adam Jablonski
CS260 - HW2

$T(n)$ is $O(f(n))$
if: c and n_0 such that $n > n_0$
$T(n) \le c f(n)$ for $n > n_0$

1.13

a. 17 is $O(1)$

$T(n) = 17$; $f(n) = 1$

Prove: $T(n) \le c \cdot f(n)$

$17 \le c \cdot 1$

Chose $c = 17$ $n_0 = 0$ (n_0 can really be anything since the function is not dependent on n)

$\boxed{17 \le 17 \cdot 1 \checkmark \quad \text{for } n \ge 0}$

b. $\frac{n(n-1)}{2}$ is $O(n^2)$

$T(n) = \frac{n^2-n}{2}$; $f(n) = n^2$

Prove: $T(n) \le c \cdot f(n)$

$\frac{n^2-n}{2} \le c \cdot n^2$

chose $c = 5$ $n_0 = 0$

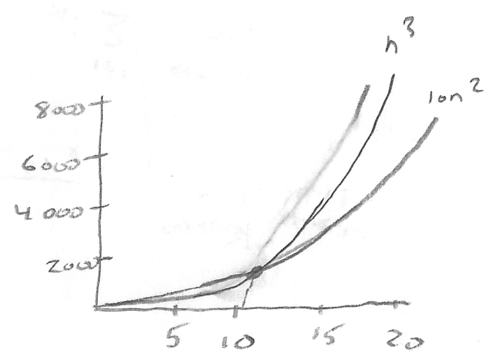
$\boxed{\frac{n^2-n}{2} \le 5 \cdot n^2 \quad n \ge 0 \quad \checkmark}$

c. $\max(n^3, 10n^2)$ is $O(n^3)$

$T(n) = \max(n^3, 10n^2)$; $f(n) = n^3$

Prove: $\max(n^3, 10n^2) \le c \cdot n^3$

chose $c = 10$ $n_0 = 10$



$\boxed{\max(n^3, 10n^2) \le 10 \cdot n^3 \quad n \ge 10}$

Because after $n_0 = 10$ the $\max(n^3, 10n^2)$ will always be n^3 therefore $n^3 \le 10 n^3$

d. $\sum_{i=1}^{n} i^k$ is $O(n^{k+1})$

$T(n) = 1^k + 2^k + 3^k + \ldots + (n-1)^k + n^k$; $f(n) = n^{k+1} = n^k + n$

Prove: $T(n) \leq c \cdot f(n)$ and $T(n) \geq c \cdot g(n)$

$$\lim_{n \to \infty} \left[\lim_{k \to \infty} T(n) \right] = \lim_{n \to \infty} \left[\lim_{k \to \infty} n^{k} + n \right]$$

* By applying L'Hopitals Rule all terms below n^k go away.

The leading coefficient (A) before the T(n) side must be equal to c fore the limit to equal 1. C must be chosen so

$C = A$ is true.

Because the limit is 1

$O(n^{k+1})$ and $\Omega(n^{k+1})$ are True

e) $p(x)$ is any k^{th} degree polynomial

$P(n)$ is $O(n^k)$

$T(n) = n^k + n^{k-1} + n^{k-2} + \dots n^3 + n^2 + n + c$; $f(n) = n^k$

Prove: $T(n) \leq C \cdot f(n)$ and $T(n) \geq c\, g(n)$

$n^k + n^{k-1} + n^{k-2} + \dots + n^3 + n^2 + n + c \leq c \cdot n^k$

let c=

$\lim_{n \to \infty} p(n) = \lim_{n \to \infty} n^k$

$\lim_{n \to \infty} \frac{n^k}{p(n)} \rightarrow$ L'Hopital Rule $\rightarrow$ all terms below n^k go away when you apply L'Hopitals Rule

therefore: The leading coefficient of the highest degree polynomial (A) c must be chosen so that A = c so the limit is 1.

Because the limit is 1 it is $O(n^k)$ and $\Omega(n^k)$

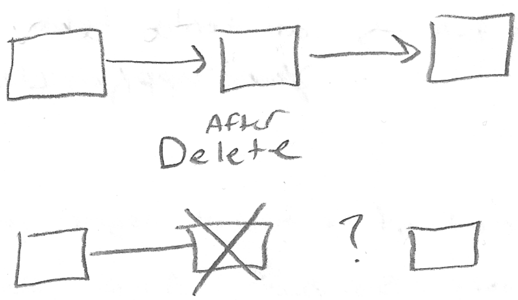
1.16

Order: $\left(\frac{3}{2}\right)^n$, $\sqrt{n}\log^2(n)$, n, $\log^2(n)$,

$\frac{n}{\log(n)}$, , $\log(n)$, $\log(\log(n))$, 17, $\left(\frac{1}{3}\right)^n$

2.9

I assume Delete(p, L) deletes both the pointer and the value therefore the next pointer location in the list will be gone



Rewritten If statement:

```
. . .                    . . .
. . .                    . . .
While p<>END(L) do begin
  if RETRIEVE(p,L)=x then
    tmp:=p
    DELETE(p,L);
    p:=NEXT(tmp,L)
  else
    p:=NEXT(p;L)
end
. . .          . . .          . . .
```

2.11

~~4~~

First loop = n times

First first = 1

First End = n

First Next = n-1

Second loop : n times

Second First = n-1

Second End : n

Second Next = n-1

third loop = q

third next = q

First: $1 + [(n-1)(n-1)] = 1 + (n^2 +$

End: $n + [n(n-1)]$

Next: $(n-1) + [(n-1)(n-1)] + [q(n-1)(n-1)]$

n = 3

||||| = 4

q = ||||||||

0 1 2

0 1 2 012 012

0 1 2

012 012 012

0 01 012 0 01 012 0 01 012

~~32~~ 18

Implementation Section Analysis

The worst case run times of two lists size 10,000:

list_concat = 0.038321018219 s

list_concat_copy = 0.0891029834747 s

List_concat takes the end pointer of list A and directs it to the beginning value of list B. the pro is that is a very fast implementation. The cons come in when any value in list A or B are changed. Since the pointer was copied the place in list C changes as well.

List_concat_copy takes the values in list A and copies it to a new list, then it takes the values of list B and copies it to the new list as well. The con is this implementation will always be slower then list_concat, and becomes more noticeable as the list sizes increase. The pro is that if a value is changed in list A or list B the new list remains unimpacted.

List_concat proves to preform faster than list_concat_copy as the size of the lists increase.