Theory Problems

1.13)

a) 17 is O(1)

$$f(n)_1 = 17 \quad f(n)_2 = 1$$

$f(n)_1$ *is* $\mathrm{O}(f(n)_2)$ or 17 is O(1) because there exists a constant c and n_0 such that $17 \leq c * 1$ for $n > n_0$,

Proof

Assuming c is some arbitrary constant, $f(n)_1 \leq cf(n)_2$

$$\rightarrow 17 \leq C * 1$$
$$\rightarrow \frac{17}{1} \leq C$$
$$\rightarrow C \geq 17$$

Assuming C = 18

If $if\ 17 \leq 18 * 1, thus\ when\ c \geq 17, 17\ is\ stricly\ bound\ above\ by\ a\ constant\ or\ \mathrm{O}(1)$

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

$$f(n)_1 = \frac{n(n-1)}{2} \rightarrow \frac{n^2 - n}{2} \quad f(n)_2 = n^2$$

$f(n)_1\ is\ \mathrm{O}(f(n)_2)\ or\ \mathrm{O}(n^2)\ since\ there\ exists\ a\ constant\ C\ and\ n_0\ such\ that\ f(n)_1 \leq cf(n)_2\ for\ n > n_0$

Proof

$$Assuming\ C\ is\ some\ arbitrary\ contanst, \frac{n(n-1)}{2} \leq cn^2$$
$$\rightarrow \frac{n^2 - n}{2} \leq cn^2$$
$$\rightarrow \frac{n^2 - n}{2n^2} \leq c \rightarrow c \approx \frac{1}{2}$$
$$\rightarrow \frac{1}{2} - \frac{1}{n} \leq \frac{1}{2}n^2$$
$$\rightarrow n(n-1) \leq n^2$$
$$\rightarrow n^2 - n \leq n^2$$
$$\rightarrow n^2 - n^2 \leq n$$
$$\rightarrow n \geq 0$$

$thus\ when\ n \geq 0\ and\ c\ is\ \approx \frac{1}{2}, f(n)_1\ is\ \mathrm{O}(f(n)_2)\ or f(n)_1\ is\ \mathrm{O}(f(n)_2)\ thus\ proving\ the\ above\ hypothesis$

c) $\max(n^3, 10n^2)\ is\ \mathrm{O}(\mathrm{n}^3)$

$$The \max of\ n^3 and\ 10n^2 is\ n^3 and\ is\ upper\ bounded\ by\ n^3$$
$$\max(n^3, 10n^2) = n^3\ is\ \mathrm{O}(\mathrm{n}^3)\ since\ there\ exists\ a\ constant\ C\ and\ n_0$$
$$such\ that\ n^3 \leq cn^3 for\ n > n_0$$

Proof

$$n^3 \leq cn^3$$
$$\to \frac{n^3}{n^3} \leq c$$
$$\to 1 \leq c$$
$$Assuming\ c \approx 100,$$
$$n^3 \leq 100n^3$$
$$\to \frac{n^3}{n^3} < 100$$
$$\to 1 \leq 100$$
$as\ shown\ above, 1\ is\ always\ less\ than\ 100\ thus\ proving\ the\ statement\ that\ for\ c \geq 1\ , n^3 will\ always\ be\ upper\ bouned\ by\ cn^3\ or\ \mathrm{O}(\mathrm{n}^3)$

d) $\sum_{i=1}^{n} i^k\ \ is\ \mathrm{O}\left(\mathrm{n}^{\mathrm{k}+1}\right)$ and $\Omega\left(\mathrm{n}^{\mathrm{k}+1}\right)$ for integer k

$$f(n)_1 = \ n^k \to \quad f(n)_2 = \mathrm{n}^{\mathrm{k}+1}$$
$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k\ always\ produces\ a\ dominant\ term\ n^k thus,$$
$$\sum_{i=1}^{n} i^k\ \ is\ \mathrm{O}\left(\mathrm{n}^{\mathrm{k}+1}\right)\ \text{ since there exists a constant c and } n_0$$
$$such\ that f(n)_1 \leq cf(n)_2\ or\ \ n^k \leq n^{k+1}\ \ for\ n >\ n_0$$

Proof

$$n^k \leq cn^{k+1}$$
$$\to n^k \leq cn^k n$$
$$\to \frac{n^k}{n^k} \leq cn$$
$$\to \frac{1}{n} \leq c \approx \frac{1}{4}$$
$$\to n^k \leq \frac{1}{4} n^{k+1}$$
$$\to\ n^k \leq \frac{1}{4} n^k n$$
$$\to \frac{n^k}{n^k} \leq \frac{n}{4}$$
$$\to 4 \leq n$$
$$, thus\ when\ c = \frac{1}{4}\ and\ n \geq 4, n^k is\ O(n^{k+1})\ thus\ proving\ the\ above\ thesis.$$

$$\sum_{i=1}^{n} i^k\ \ is\ \Omega\left(\mathrm{n}^{\mathrm{k}+1}\right)\ \text{ since there exists a constant c and } n_0$$
$$such\ that\ such\ that f(n)_1 \geq \ cf(n)_2\ or\ n^k \geq cn^{k+1}\ for\ n >\ n_0$$

Proof

$$n^k \geq cn^{k+1}$$
$$\to n^k \geq cn^k n$$

$$\rightarrow \frac{n^k}{n^k} \geq cn$$
$$\rightarrow \frac{1}{n} \geq c \approx \frac{1}{4}$$
$$\rightarrow n^k \geq \frac{1}{4} n^{k+1}$$
$$\rightarrow \ n^k \leq \frac{1}{4} n^k n$$
$$\rightarrow \frac{n^k}{n^k} \geq \ \frac{n}{4}$$
$$\rightarrow 4 \geq \ n$$
$$, thus\ when\ c = \ \frac{1}{4}\ and\ n \leq \ 4, n^k is\ lower\ bounded\ by\ \ (n^{k+1})$$
$$thus\ proving\ the\ above\ thesis.$$

e) $p(x) = \ x^k \approx an^k\ when\ a > 0$

$$f(n)_1 = \ \ an^k \rightarrow \ \ \ f(n)_2 = \mathrm{n^k}$$
$$`p(n) = \ an^k\ is\ O(n^k)\ since\ there\ exists\ a\ constant\ c\ and\ n_0$$
$$such\ that\ \ f(n)_1 \leq cf(n)_2\, an^k \leq cn^k\ when\ n > \ n_0$$

Proof

$$an^k \leq cn^k$$
$$\rightarrow a \leq c$$
$$\rightarrow c \geq a$$
$$c \approx 6, a \approx 4$$
$$assuming\ c = 6\ and\ a = 4, 4n^k \leq 6n^k \rightarrow 4 < 6\ thus\ proving$$
$$that\ an^k\ is\ upper\ bound\ by\ n^k when\ c > a.$$
$$The\ inverse\ is\ true\ when\ c < a\ proving\ that\ f(n)_1$$
$$\geq \ cf(n)_2\ \ or\ f(n)_1\ is\ lower\ bound\ by\ f(n)_2$$

Proof

$$`p(n) = \ an^k\ is\ \Omega(n^k)\ since\ there\ exists\ a\ constant\ c\ and\ n_0$$
$$such\ that\ \ f(n)_1 \geq \ cf(n)_2\, an^k \geq \ cn^k\ when\ n > \ n_0$$
$$an^k \geq cn^k$$
$$\rightarrow \ c \leq a$$
$$c \approx 4, a \approx 6$$
$$6n^k \geq 4n^k \rightarrow 6n^k \geq 4n^k \rightarrow 6 \geq$$
$$4\ thus\ satisfying\ the\ above\ conditions\ showing\ that\ an^k is\ \Omega\left(\mathrm{n^k}\right)\ when\ a \geq c$$

1.16)

ordering from decreasing to increasing

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

$$\rightarrow \log(n)$$
$$\rightarrow \log(n)^2$$
$$\rightarrow \sqrt{n}$$
$$\rightarrow \sqrt{n}\log(n)$$
$$\rightarrow nlog(n)$$
$$\rightarrow n \rightarrow \left(\frac{3}{2}\right)^n$$

1.18) Here is a function *max*(*i*, *n*) that returns the largest element in positions *i* through *i*+*n*-1 of an integer array *A*. You may assume for convenience that *n* is a power of 2.

function *max* (*i*, *n*: integer):

```
    integer; var
    m1, m2: integer; begin
    If n = 1 then
        return (A[i])
    else begin
        m1 := max(i, n div 2);
         m2 :=max(i+n div 2,n div 2)
        if m1 < m2 then
            return (m2)
            else
            return (m1)
    end
end
```

a)

$$\max(1,1,) \rightarrow 1$$

$$\rightarrow \max(2,2)\ \ yields \max(2,1), \max(2,1) \rightarrow 3$$

$$\rightarrow \max(4,4)\ y\ ields \max(4,4) \max(4,2) \max(4,2) \max(4,1) \max(4,1) \max(3,1) \max(3,1) \rightarrow 7$$

$$\rightarrow \max(8,8)\ yields \max(8,8)\ \max(8,4) \max(8,4) \max(8,2) \max(8,2) \max(6,2) \max(6,2) \max(8,1)$$

$$\max(5,1)\max(8,1)\max(5,1)\max(6,1)\max(6,1),\max(4,1)\max(4,1) \rightarrow 15$$

b)

$$n = 1 \rightarrow 1$$

$$n = 2 \rightarrow 3$$

$$n = 4 \rightarrow 7$$

$$n = 8 \rightarrow 15$$

$$since\ n\ is\ a\ power\ of\ two, the\ following\ expression\ models\ the\ relationship\ between\ n\ and\ the$$

$$number\ of\ \max function\ call\ performed.$$

$$T(n) = 2^{\log(n)+1} - 1$$

$$T(n)\ is\ O(n)\ since\ there\ exists\ a\ constant\ c\ and\ n_0\ such\ that\ T(n) \leq cn\ for\ n > n_0$$

$$\boldsymbol{Proof}$$

$$2^{\log(n)+1} - 1 \leq cn^2$$

$$\rightarrow simpify\ 2^{\log(n)+1} - 1\ first$$

$$\rightarrow y = 2^{\log(n)+1} - 1$$

$$\rightarrow log_2 y = log_2 2^{\log(n)+1} - 1$$

$$\rightarrow log_2 y = log_2 2^{\log(n)} 2 - 1$$

$$\rightarrow log_2 y = log_2 (n) log_2 2 * 2 - 1$$

$$\rightarrow log_2 y = log_2 (n) log_2 4 - 1$$

$$\rightarrow log_2 y = log_2 (n) * 2 - 1$$

$$\rightarrow n * 2 - 1$$

$$\rightarrow 2n - 1$$

$$\rightarrow continuing\ with\ proof$$

$$\rightarrow 2n - 1 \leq cn$$

$$\rightarrow \frac{2n-1}{n} \leq c$$

$$\rightarrow c = 10$$

$$\rightarrow 2n - 1 \leq 10n$$

$$\rightarrow\ -1 \leq 8n \rightarrow\ n \geq\ -\frac{1}{8}\ t$$

$$thus\ the\ above\ condition\ is\ satisfied\ when\ n >\ -\tfrac{1}{8}\ and\ c\ is\ 10.$$

$$T(n) =\ o(n)\ since\ \lim_{n\to\infty}\left(\frac{(2^{\log(n)+1} - 1)}{cn}\right)^{1} = 0\ \ for\ c = 10\ and\ n <\ -1/8$$

2.9) The following procedure was intended to remove all occurrences of element *x* from list *L*. Explain why it doesn't always work and suggest a way to repair the procedure so it performs its intended task.

```
procedure delete ( x: elementtype; var L: LIST );

var p: position;

begin

p := FIRST(L);

while p <> END(L) do

begin

        if RETRIEVE(p, L) = x then

                DELETE(p, L);

        p := NEXT(p, L) end

end; { delete }
```

This procedure misbehaves because when deletion occurs, all prior element indexes get decremented by a factor of 1. Now, because we are incrementing our current position by one upon deletion, we are potentially skipping an element

Index 0	Index 1	Index 2	Index 3	Index 4	Index 5
data 1	data 2	data 2	data 4	data 5	data 1

Assuming we aim to delete all values of 2 in the array and we delete the first occurence of 2, upor deletion the new list looks like the one below as all indexes are decremented

Index 0	Index 1	Index 2	Index 3	Index 4
data 1	data 2	data 4	data 5	data 1

We increment our position by 1 thereby missing the second occurence of two or skipping one element

To fix this, we could condition on a delete event in the following manner,

```
Begin
  p = first(L)
    while <> End(L) do
      Begin
        If retrieve(p, L) = x then
          delete(p, l)
          p = p   # dont increment the pointer as new list is defined wih prior indexes
                − 1
       Else
         p = next(p, L)
      End
End
```

So we only increment our pointer if an item was not deleted

2.11) Suppose L is a LIST and p, q, and r are positions. As a function of n, the length of list L, determine how many times the functions FIRST, END, and NEXT are executed by the following program.

```
p := FIRST(L);        Line 1   First runs 1 time
    while p <> END(L) do begin        Line 2   End runs n times
        q := p;              Line 3
        while q <> END(L) do begin        Line 4   End runs n-q+1 times
            q := NEXT(q, L);          Line 5     Next runs n-q times
            r := FIRST(L);          Line 6       First runs n-q times
            while r <> q do          Line 7
                r := NEXT(r, L)         Line 8  Next runs q-1 times
        end;           Line 9
        p := NEXT(p, L)      Line 10    Next runs n-1 times
end;      Line 11
```

First()

$$\rightarrow runs\ once\ in\ line\ 1$$

$$\rightarrow runs\ n-q\ in\ line\ 8\ with\ q\ taking\ values\ 1\ to\ n-1$$
$$and\ n\ taking\ values\ n-1\ to\ 1 \rightarrow \boldsymbol{\frac{n(n-1)}{2}}$$

End()

$$\rightarrow runs\ n\ times\ in\ line\ 2$$

$$\rightarrow runs\ n-q+1\ times\ in\ line\ 4\ with\ q\ taking\ valus\ in\ the\ range\ 1\ to\ n$$
$$-1\ and\ n\ taking\ values\ in\ the\ range\ n\ to\ 2 \rightarrow$$
$$\frac{n(n-1)}{2} - 1 \rightarrow \boldsymbol{\frac{n(n-1)}{2} - 1}.$$

$$\boldsymbol{Next}()$$

$$in\ line\ 5, runs\ n-q\ times\ with\ q\ taing\ values\ in\ the\ range\ 1\ to\ n-1\ and\ n\ taking\ values\ in\ the\ range\ n-1\ to\ 1\ for\ \frac{n(n-1)}{2}$$

$$\rightarrow in\ line\ 8, runs\ q-1\ \ times\ with\ q\ taking\ vlaues\ in\ the\ range\ 2\ to\ n\ and\ n\ takig\ values\ in\ the\ range\ 1\ to\ n-2\ for\ \frac{(n-2)(n-1)}{2} \rightarrow in\ line\ 10\ runs\ n-1\ times\ for\ a\ total\ of$$

$$\rightarrow \boldsymbol{\frac{n(n-1)}{2} + \frac{(n-2)(n-1)}{2} + (n-1)}$$