CS 260 – Homework #2

Chris Kasper

January 21, 2018

Problem 1 (1.13 from text)

Show that the following statements are true.

$$a.\ 17\ is\ O(1)$$

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

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

$$d.\ \sum_{i=0}^{n} i^k\ is\ O(n^{k+1})\ and\ \Omega(n^{k+1})\ for\ integer\ k$$

e. If p(x) is any k^{th} degree polynomial with a positive leading coefficient, then p(n) is $O(n^k)$ and $\Omega(n^k)$

Solution

$a.\ 17\ is\ O(1)$ $c(1) \geq 17, \forall\, n > n_0$ $c = 34\, ;\, n_0 = 1$ $34 \geq 17$ $\therefore 17\ is\ O(1)\ \forall\, n \geq 1$	$b.\ \frac{n(n-1)}{2}\ is\ O(n^2)$ $cn^2 \geq \frac{n(n-1)}{2}, \forall\, n > n_0$ $c = 2\, ;\, n_0 = 1$ $4n^2 \geq n^2 - n$ $3n^2 + n \geq 0$ $n(3n+1) \geq 0$ $\therefore \frac{n(n-1)}{2}\ is\ O(n^2)\ \forall\, n \geq 1$

$c. \max(n^3, 10n^2)\ is\ O(n^3)$ $\underline{n \leq 10}$ $cn^3 \geq 10n^2$ $c = 1\ ;n_0 = 1$ $n^3 \geq 10n^2$ $n^3 - 10n^2 \geq\ 0$ $n^2(n-10) \geq\ 0$ $\underline{n > 10}$ $cn^3 \geq n^3$ $c = 2\ ;n_0 = 1$ $2n^3 \geq n^3$ $n^3 \geq\ 0$ $\therefore \max(n^3, 10n^2)\ is\ O(n^3)$	$d.\ \sum_{i=0}^{n} i^k\ is\ O(n^{k+1})\ and\ \Omega(n^{k+1})\ for\ integer\ k$ $\sum_{i=0}^{n} i^k\ is\ O(n^{k+1})$ $k = 1$ $Summation = \frac{n(n+1)}{2}$ $cn^2 \geq \frac{n(n+1)}{2}$ $c = 2, n_0 = 1$ $2n^2 \geq \frac{n^2+n}{2}$ $4n^2 \geq n^2 + n$ $3n^2 - n \geq 0$ $n(3n-1) \geq 0$ $\sum_{i=0}^{n} i^k\ is\ \Omega(n^{k+1})$ $k = 1$ $c * \frac{n(n+1)}{2} \geq n^2$ $c = 4\ ;n_0 = 1$ $4n^2 + 4n \geq 2n^2$ $2n^2 + 4n \geq 0$ $2n(2n+2) \geq 0$ $\therefore \sum_{i=0}^{n} i^k\ is\ O(n^{k+1})\ and\ \Omega(n^{k+1})\ for\ integer\ k$

e. If p(x) is any k^{th} degree polynomial with a positive leading coefficient, then p(n) is $O(n^k)$ and $\Omega(n^k)$

$$O(n^k)$$
$$cn^k \geq an^k + bn^{k-1} + dn^{k-2}$$
$$(c-a)n^k - bn^{k-1} - dn^{k-2} \geq 0$$
$$if\ k = 2\ ; c = 2; a = 1; b = 1; d = 1$$
$$(2-1)n^2 - 1n - 1n^0 \geq 0$$
$$n^2 - n \geq 1$$
$$n(n-1) \geq 1$$

$$\therefore p(x)\ is\ \ O(n^k)$$

$$\Omega(n^k)$$

$$if\ k = 2\ ; c = 1; a = 3; b = 1; d = 1$$
$$c(3n^k + n^{k-1} + 1n^{k-2}) \geq n^k$$
$$3n^2 + n + 1 \geq n^2$$
$$2n^2 + n + 1 \geq 0$$
$$\therefore p(x)\ is\ \ \Omega(n^k)$$

Problem 2 (1.16 from text)

Order the following functions by growth rate: (a) n, (b) $\sqrt{}$n, (c) logn, (d) loglogn, (e) log2n, (f) n/logn, (g) $\sqrt{}$nlog2n, (h) (1/3)n, (i) (3/2)n, (j) 17.

Solution

In decreasing order:

j. 17

i. $(3/2)^n$

a. n

f. n/log(n)

g. $\sqrt{}$nlog2n

b. $\sqrt{}$n

e. log2n

c. logn

d. log(log(n))

h. $(1/3)^n$

Problem 3 (1.18 from text)

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. Let T(n) be the worst-case time taken by max with second argument n. That is, n is the number of elements of which the largest is found. Write an equation expressing T(n) in terms of T(j) for one or more values of j less than n and a constant or constants that represent the times taken by individual statements of the max program.

b. Give a tight big oh upper bound on T(n). Your answer should be equal to the big omega lower bound, and be as simple as possible.

Solution

a. **n=2 ... 2 more max() calls in addition to the first**
   n=4 ... 4 more max() calls
   n=8 ... 8 more max() calls
   Mapped to a function, T(j)=2*T(j/2)+1

b. **T(n) is O(n)**

Problem 4 (2.9 from text)

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 }
```

Solution

An issue arises with this particular delete procedure when there are consecutive occurrences of the element x. After deleting an x, the position moves to the next one. However, the procedure fails to check if the new element in the position that was just deleted, if it contains the element x. This can be fixed by including an else statement to the if of the retrieve. This will make sure consecutive occurrences of x won't get missed.

Problem 5 (2.11 from text)

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);   O(1)
while p <> END(L) do begin O(n)
    q := p;
    while q <> END(L) do begin O(n)
        q := NEXT(q, L);
        r := FIRST(L);
        while r <> q do O(n)
            r := NEXT(r, L)
    end;
    p := NEXT(p, L)
end;
```

Solution

FIRST => 1+n*n = 1+n^2

END => n*n = n^2

NEXT => n*n*n = n^3