Conner Deakin

CS 330 Homework 1

① a. Show $4n + 8 = O(n)$

$4n + 8 \le cn$ where $n \ge n_0$

$\longrightarrow c = 5$

$4n + 8 \le 5n$

$8 \le n$

$\longrightarrow n_0 = 8$

$4n + 8 \le cn$ where $c = 5$ $\forall (n \ge 8)$

b. Show $15n^2 - 50 = \Theta(n^2)$

$c_1 n^2 \le 15n^2 - 50 \le c_2 n^2$

left side

$c_1 n^2 \le 15n^2 - 50$

$\longrightarrow c_1 = 5$

$5n^2 \le 15n^2 - 50$

$-10n^2 \le -50$

$n^2 \ge 5$

$n \ge \sqrt{5}$

$n_0 = 3$

right side

$15n^2 - 50 \le c_2 n^2$

$\longrightarrow c_2 = 20$

$15n^2 - 50 \le 20n^2$

$-5n^2 - 50 \le 0$

$5n^2 + 50 \ge 0$

$n^2 \ge -10$

$n \ge 0$

$c_1 n^2 \le 15n^2 - 50 \le c_2 n^2$ where $c_1 = 5$, $c_2 = 20$ $\forall (n \ge 3)$

c. Show n is not in $O(\sqrt{n})$

$$n \le \sqrt{n}$$

$$n/\sqrt{n} \le 1$$

$$\sqrt{n} \le 1$$

this is not true

$\forall (n > 1)$

hence, n is not in $O(\sqrt{n})$

d. Show $n^2 = O(10^n)$

$$n^2 \le c\,10^n \quad c = 1$$

$$n^2 \le 10^n$$

$$\log(n^2) \le \log(10^n)$$

$$2\log(n) \le n\log(10)$$

$$\frac{\log(n)}{n} \le \frac{\log(10)}{2}$$

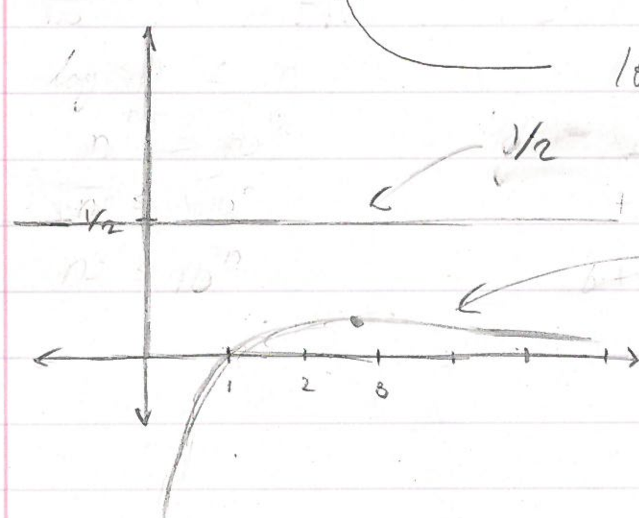
$$\log(n)/n \le 1/2$$

true for

$n_0 = 1$

$c = 1$

$\forall (n \ge n_0)$



How would one find the maximum of this?

$\log(n)/n$

②

```
for (i=0; i<n; ++i) {
    for (j=0; j≤n; ++j) {
        g();
        g();
    }
    for (k=0; k<n; ++k) {
        g();
    }
    g();
    g();
    g();
}
g();
g();
```

inner j loop: $4(n+1)$; k loop: $2n$; three g() calls: 6; final two g() calls: 4

a. find concrete runtime $T(n)$ $\quad$ g() = 2 units

The runtime of 1 iteration of scope 1 ({①) is

$$4(n+1) + 2n + 6$$ This runtime is done n times

So the runtime of the code with for loop is

$$\overbrace{n(4(n+1)+2n+6)} + 4$$ 2 g() calls

$$T(n) = n(4(n+1) + 2n + 6) + 4$$
$$= n(4n + 4 + 2n + 6) + 4$$
$$= n(6n + 10) + 4$$
$$T(n) = 6n^2 + 10n + 4$$

6. Prove that the efficiency class is $O(n^2)$

$f(n) = 6n^2 + 10n + 4$
$g(n) = n^2$

show that: $f(n) \le c\,g(n) \ \forall (n \ge n_0)$ $\quad C = 7$

$6n^2 + 10n + 4 \le 7n^2$
$-n^2 + 10n + 4 \le 0$
$n^2 - 10n - 4 \ge 0$

We should try to make this factorization easier

$(6n + 2)(n - 2)$
$6n^2 - 12n + 2n - 4$
$6n^2 - 10n - 4$

C = 12 makes this possible

$6n^2 + 10n + 4 \leq 12n^2$
$-6n^2 + 10n + 4 \leq 0$
$6n^2 - 10n + 4 \geq 0$
$(6n+2)(n-2) \geq 0$

inequality is true for $n \geq 2$.

$T_1(n) \leq cn^2$
where $c = 12$, $n_0 = 2$
$\forall (n \geq n_0)$

③ ALG(A)

```
{
  R = 0
  I = A
  while (I > 0) {
      R = R + 2
      I = I - 1
  }
  return R;
}
```

Our loop invariants are

$I_k = A - k$
$R_k = 2k$

Base case: $k = 0$

$I_0 = A - 0 = A = I_0$
$R_0 = 2(0) = 0 = R_0$

How our invariants change over an iteration

$I_{k+1} = I_k - 1$
$R_{k+1} = R_k + 2$

Inductive Step

Assume: $d_k = A - k$, $R_k = 2k$

Show: $d_{k+1} = A - (k+1)$, $R_{k+1} = 2(k+1)$

$d_k - 1 = A - (k+1)$ $\quad$ $R_k + 2 = 2(k+1)$

$A - k - 1 = A - (k+1)$ $\quad$ $2k + 2 = 2(k+1)$

$A - (k+1) = A - (k+1)$ $\quad$ $2(k+1) = 2(k+1)$ ✓

Proof of loop termination

Since d decreases

$\exists d_t \, ((d_t > 0) \wedge (d_{t+1} \leq 0))$ ✓

Proof of algorithm correctness.

We know $d_{t+1} = 0$ $\longrightarrow$ what is k when d_k is 0

$d_k = A - k$

$0 = A - k$

$k = A$

From our loop invariant

$R_k = 2k$

$R_A = 2A$ $\longrightarrow$ our algorithm returns $2A$ ✓

(4) ALG(A) → Returns A/2

{

$R = 0$

$d = 2$

while ($d \le A$) {

  if (d is even)

    $R = R + 1$

  $d = d + 1$

}

return R

}

Our loop invariants

$d_k = k + 2$

$R_k = \left\lfloor \frac{k+1}{2} \right\rfloor$ — or

d_k is odd

$R_k = (d_k - 1)/2$

d_k is even

$R_k = (d_k - 2)/2$

~~Proof of Base Case~~

$k = 0$

$d_0 = 0 + 2$

$2 = 2$ ✓

$R_0 = \left\lfloor \frac{0+1}{2} \right\rfloor$

$0 = \lfloor 1/2 \rfloor$

$0 = 0$ ✓

How our invariants change over an iteration.

$d_{k+1} = d_k + 1$

R_{k+1}

d_k is odd

$R_{k+1} = R_k$

d_k is even

$R_{k+1} = R_k + 1$

Inductive Step

Assume:

$A_k = k + 2$

If A_k is odd $R_k = (A_k - 1)/2$

If A_k is even $R_k = (A_k - 2)/2$

Show:

$A_{k+1} = (k+1) + 2$

If A_{k+1} is odd $R_{k+1} = (A_{k+1} - 1)/2$

If A_{k+1} is even $R_{k+1} = (A_{k+1} - 2)/2$

$A_{k+1} = (k+1) + 2$ ← A_{k+1} proof

$A_k + 1 = (k+1) + 2$

$k + 2 + 1 = k + 3$

$k + 3 = k + 3$ ✓

$R_{k+1} = (A_{k+1} - 2)/2$ ← R_{k+1} where A_k is odd

$R_{k+1} = (A_k + 1 - 2)/2$ ↑ A_{k+1} is even

$R_k = (A_k + 1 - 2)/2$

$(A_k - 1)/2 = (A_k - 1)/2$ ✓

$R_{k+1} = (A_{k+1} - 1)/2$ ← R_{k+1} where A_k is even

$R_k + 1 = A_k/2$ ↑ A_{k+1} is odd

$(A_k - 2)/2 + 1 = A_k/2$

$A_k/2 - 1 + 1$

$A_k/2 = A_k/2$ ✓

Proof of loop termination

Since d represents a strictly increasing sequence of integers, d cannot be bounded so

$$\exists t\,((d_t \le A) \wedge (d_{t+1} > A)) \quad ✓$$

Proof of algorithm correctness

When we reach the loop terminator d_{t+1}, our algorithm should return $A/2$

So we look at the values when our loop terminates

$$d_{t+1} > A$$
$$d_{t+1} = A+1$$
$$(t+2)+1 = A+1$$
$$t+3 = A+1$$
$$t+2 = A$$

d_{t+1} is odd

$$R_{t+1} = (d_{t+1} - 1)/2$$
$$R_{t+1} = (d_t + 1 - 1)/2$$
$$R_{t+1} = (t+2)/2$$
$$R_{t+1} = A/2$$

d_{t+1} is even

$$R_{t+1} = (d_{t+1} - 2)/2$$
$$R_{t+1} = ((t+2)+1-2)/2$$
$$R_{t+1} = (A-1)/2$$

This makes sense because

$d_{t+1} = A+1$ → let's say $A=5$

$$d_{t+1} = 5+1$$
$$d_{t+1} = 6$$

assuming int values

$$\lfloor 5/2 \rfloor = (5-1)/2$$