

# Algorithm Design Manual Solutions

Zachary William Grimm

Solutions to Selected Problems

zwgrimm@gmail.com

February 4, 2019

## Contents

<b>1</b>	<b>Introduction To Algorithm Design</b>	<b>1</b>
1.1	Finding Counter Examples . . . . .	1

## 1 Introduction To Algorithm Design

### 1.1 Finding Counter Examples

**Ex:1-1.** *Show that  $a + b$  can be less than  $\min(a, b)$*

Let  $a = -1, b = -1$

Then  $a + b = -2, \min(a, b) = -1$

$\therefore \exists a, b \in \mathbb{Z} : a + b < \min(a, b)$

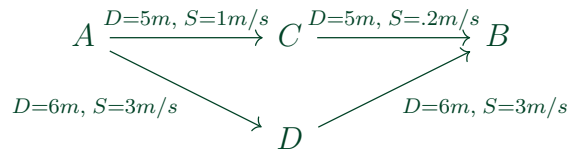
**Ex:1-2.** *Show that  $a * b$  can be less than  $\min(a, b)$*

Let  $a = -1, b = 5$ .

Then  $a * b = -5, \min(a, b) = -1$

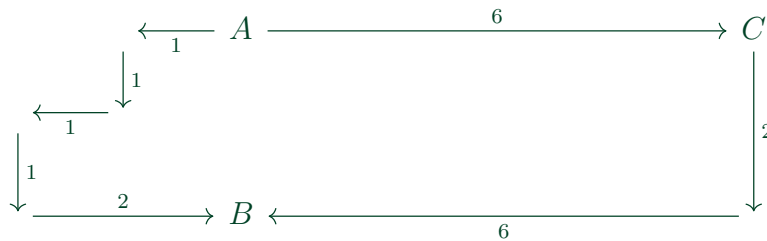
$\therefore \exists a, b \in \mathbb{Z} : a * b < \min(a, b)$

**Ex:1-3.** *Design/draw a road network with two points  $a$  and  $b$  such that the fastest route between  $a$  and  $b$  is not the shortest route*



Although the distance from A to B through C is shorter than going through D, road constraints limit the time it takes making the route through D faster despite it being longer.

**Ex:1-4.** Design/draw a road network with two points  $a$  and  $b$  such that the shortest route between  $a$  and  $b$  is not the route with the fewest turns



The route from A through C to B has only two turns but is a total length of 14 units while the direct route from A to B (the shortest) has 4 turns and is a length of 6 units.  $\therefore$  The shortest route between A and B is not the route with the fewest turns.

## Proofs of Correctness

**1-7.** Prove the correctness of the following recursive algorithm to multiply two natural numbers, for all integer constants  $c \geq 2$

function  $multiply(y, z)$

1. If  $z = 0$  then return(0) else
2. return  $multiply(cy, \lfloor (z/c) \rfloor) + y(z \bmod c)$

**case 1:  $c = z$**

$$\begin{aligned} \text{multiply}(y, z) &= \text{multiply}(cy, \lfloor (z/c) \rfloor) + y(z \bmod c) \\ &= \text{multiply}(zy, \lfloor (z/z) \rfloor) + y(z \bmod z) \\ &= \text{multiply}(zy, 1) + y(z \bmod z) \\ &= \text{multiply}(zy, 1) \\ &= \text{multiply}(czy, \lfloor (1/c) \rfloor) + zy(1 \bmod c) \\ &= \text{multiply}(zzy, \lfloor (1/z) \rfloor) + zy(1 \bmod z) \\ &= \text{multiply}(zzy, 0) + zy(1 \bmod z) \\ &= yz \end{aligned}$$

**case 2:  $c > z$**

$$\begin{aligned} \text{multiply}(y, z) &= \text{multiply}(cy, \lfloor (z/c) \rfloor) + y(z \bmod c) \\ &= \text{multiply}(cy, 0) + y(z \bmod c) \\ &= \text{multiply}(cy, 0) + y(z \bmod c) \\ &= yz \end{aligned}$$

**case 3:  $c < z$**

**Assumptions:**

$$c \geq 2$$

$$z \leq n \text{ (inductive hypothesis)}$$

$$y \geq 0$$

**Base Case:**  $z = 0$ ,  $multiply(y, 0) = 0$ , (which is true)

**Lemma:** we show that

$$\lfloor (z/c) \rfloor * c + (z \bmod c) = z$$

by the quotient remainder theorem

$$\begin{aligned} z &= cq + r \\ &= cq + z \bmod c \\ &= \lfloor (z/c) \rfloor * c + (z \bmod c) \quad (1^*) \end{aligned}$$

Assuming the algorithm holds for all numbers  $\leq n$ , we must show that

$$multiply(y, n + 1) = y(n + 1)$$

Now,

$$multiply(y, n + 1) = multiply(cy, \lfloor ((n + 1)/c) \rfloor) + y((n + 1) \bmod c)$$

$$\text{since } c \geq 2,$$

$$\lfloor ((n + 1)/c) \rfloor < n + 1$$

$\therefore$  the first term returns a valid result (based on our inductive hypothesis) so following the algorithm:

$$multiply(y, n + 1) = multiply(cy, \lfloor ((n + 1)/c) \rfloor) + y((n + 1) \bmod c)$$

(for simplicity let  $z' = n + 1$ )

$$= cy \lfloor ((z')/c) \rfloor + y((z') \bmod c)$$

$$= y(c \lfloor (z'/c) \rfloor + (z' \bmod c))$$

and from (1\*)

$$= yz'$$

$$= y(n + 1)$$

■

**1-8.** Prove the correctness of the following algorithm for evaluating a polynomial:

$$P(x) = a_n^n + a_{n-1}^{n-1} + \dots + a_1 + a_0$$

*function* horner( $A, x$ ) :

$$p = A_n$$

for  $i$  from  $n - 1$  to 0

$$p = px + A_i$$

return  $p$

For polynomials of degree 0,  $P(x) = A_0$ , which the algorithm satisfies.

Assuming the algorithm holds for polynomials of degree  $\leq n$  and that  $A$  is an ordered set of coefficients of size

$$\text{horner}(A, x) = P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

we must show it holds for polynomials of degree  $n + 1$ , ie:  $A'$  is an ordered set of coefficients of size  $n + 1$ ,  $[A_{n+1}, A_n, \dots, A_0]$

**horner**(  $A', x$ ) :

$$p \Rightarrow$$

$$A'_{n+1}$$

$$A'_{n+1} * x + A'_n$$

$$A'_{n+1} * x^2 + A'_n * x + A'_{n-1}$$

.

.

.

$$A'_{n+1} * x^{n+1} + A'_n * x^n + \dots + A'_1 * x + A'_0$$

=

$$A'_{n+1} * x^{n+1} + \text{horner}(A, x)$$

■ ??? seems circular... define what we need to show better

**1-9.** Prove the correctness of the following sorting algorithm:

**function** bubblesort ( $A$  : list[1 ...n])

var int  $i, j$

for  $i$  from  $n$  to 1

for  $j$  from 1 to  $i - 1$

if  $A[j] > A[j + 1]$

swap the values of  $A[j]$  and  $A[j + 1]$

**Base case** is a list of two elements  $[a, b]$  (we are indexing from 1 not 0 as usual)

**case 1:  $a > b$**

```

        i = 2
        j = 1
        if (A[1] > A[2]) = true
        swap values, so A = [b, a]
        i = 1
        j = 1
        if (A[1] > A[2]) = false

```

so  $A = [b, a]$  and from our assumption that  $a > b$  for case 1 the algorithm is true.

**case 2:  $a < b$**

```

        i = 2
        j = 1
        if (A[1] > A[2]) = false
        swap values, so A = [b, a]
        i = 1
        j = 1
        if (A[1] > A[2]) = false

```

so  $A = [a, b]$  and from our assumption that  $a < b$  for case 2 the algorithm is true.

**case 3:  $a = b$**

The list is considered sorted regardless of how the algorithm may rearrange its items  
 Now we proceed with induction, assuming that for all lists of size  $\leq n$  the algorithm returns a sorted list  
 We must show that for a list of size  $n+1$ , the algorithm returns an ordered list that is function bubblesort ( $A : list[1...n, n+1]$ )

```

        var int i, j
        for i from n + 1 to 1
        for j from 1 to i - 1
        if (A[j] > A[j + 1])
        swap the values of A[j] and A[j + 1]

```

The inner loop "bubbles up" the largest num to position i  
 on the last j iteration of the first i iteration the altered set has  $n+1$  elems with the largest element in position  $n+1$  now we have sort the rest of the list which is of size  $n$ .  
 Our inductive hypothesis states we can do this. ■

**Induction**

**1-10.** Prove that  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$  for  $n \geq 0$ , by induction

Let  $n = 1$ , then

$$\frac{n(n+1)}{2} = 1 = \sum_{i=1}^1 i$$

establishing our base case. We assume that  $\exists n \geq 0 \in \mathbb{N}$  where

$$\sum_{i=1}^n i = \frac{n(n+1)}{2} \text{ for } n \leq \text{some } k \in \mathbb{N}$$

We must show

$$\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

We can see that

$$\begin{aligned} \sum_{i=1}^{n+1} i &= (n+1) + \sum_{i=1}^n i \\ &= (n+1) + \frac{n(n+1)}{2} \\ &= \frac{[2(n+1) + n(n+1)]}{2} \\ &= \frac{(n+1)(n+2)}{2} \blacksquare \end{aligned}$$

**1-11.** Prove that  $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$  for  $n \geq 0$ , by induction

Let  $n = 1$ . Then

$$\frac{n(n+1)(2n+1)}{6} = 1 = \sum_{i=1}^1 i^2$$

establishing our base case. We assume that  $\exists n \geq 0 \in \mathbb{N}$  where

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} \text{ for } n \leq \text{some } k \in \mathbb{N}$$

We must show

$$\sum_{i=1}^{n+1} i^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

We can see that

$$\begin{aligned}\sum_{i=1}^{n+1} i^2 &= (n+1)^2 + \sum_{i=1}^n i^2 \\&= (n+1)^2 + \frac{n(n+1)(2n+1)}{6} \\&= \frac{(n+1)[6(n+1) + n(2n+1)]}{6} \\&= \frac{(n+1)[2n^2 + 7n + 6]}{6} \\&= \frac{(n+1)(n+2)(2n+3)}{6} \blacksquare\end{aligned}$$

**1-12.** Prove that  $\sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}$  for  $n \geq 0$ , by induction