Algorithm Design Manual Solutions

Zachary William Grimm
Solutions to Selected Problems
zwgrimm@gmail.com

2/2/2019

1 Introduction To Algorithm Design

Finding Counter Examples

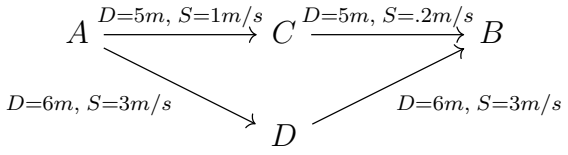
1-1. *Show that a + b can be less than min(a, b)*

$$\text{Let } a = -1, b = -1$$
$$\text{Then } a + b = -2,\ min(a, b) = -1$$
$$\therefore \exists\ a, b \in \mathbb{Z} : a + b < min(a, b)$$

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

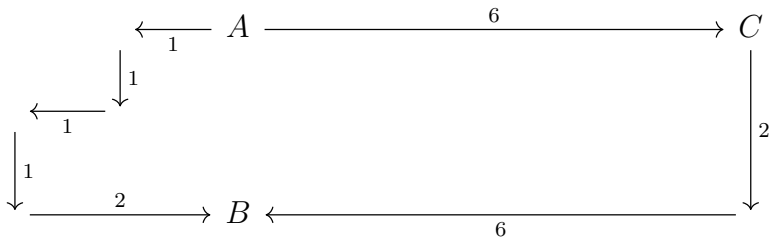
$$\text{Let } a = -1, b = 5.$$
$$\text{Then } a * b = -5,\ min(a, b) = -1$$
$$\therefore \exists\ a, b \in \mathbb{Z} : a * b < min(a, b)$$

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.

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. ∴ The shortest route between A and B is not the route with the fewest turns.

1-5. ***The knapsack problem is as follows:*** *Given a set of integers S = s1, s2,...,sn, and a target number T, find a subset of S which adds up exactly to T. For example, there exists a subset within S = 1, 2, 5, 9, 10 that adds up to T = 22 but not T = 23. Find counterexamples to each of the following algorithms for the knapsack problem. That is, giving an S and T such that the subset is selected using the algorithm does not leave the knapsack completely full, even though such a solution exists.*

(a) ***Put the elements of S in the knapsack in left to right order if they fit, i.e. the first-fit algorithm.***
Let $S = \{1, 7, 9\},\ T = 10$

(b) ***Put the elements of S in the knapsack from smallest to largest, i.e. the best-fit algorithm.***
Let $S = \{1, 7, 9\},\ T = 10$

(c) ***Put the elements of S in the knapsack from largest to smallest.***
Let $S = \{1, 4, 5, 7, 9\},\ T = 19$

1-6. ***The set cover problem is as follows:*** *Given a set of subsets S1, ..., Sm of the universal set $U = \{1, ..., n\}$, find the smallest subset of subsets $T \subset S$ such that $\cup_{t_i \in T} t_i = U$ For example, there are the following subsets, $S_1 = \{1, 3, 5\}, S_2 = \{2, 4\}, S_3 = \{1, 4\},\ and\ S_4 = \{2, 5\}$ The set cover would then be S_1 and S_2 .*

Find a counterexample for the following algorithm: *Select the largest subset for the cover, and then delete all its elements from the universal set. Repeat by adding the subset containing the largest number of uncovered elements until all are covered.*

$$U = \{1,2,3,4,5,6,7,8,9,10,11\},\ S_1 = \{1,2,3,4,5\},\ S_2 = \{6,7,8,9\},$$

$$S_3 = \{6,7,10\},\ S_4 = \{8,9,11\}$$

$$U^{'} = \{6,7,8,9,10,11\},\ T^{'} = S_1$$

The Set with largest number of uncovered elements is S_2. S_3 and S_4 must also be included in the set cover because they contain elements (10 & 11 respectivly) in U that are not members of any other subset. Therefore this algorithm gives us as the set cover all of the subsets, but $S_1 \cup S_3 \cup S_4$ covers U and is smaller than $S_1 \cup S_2 \cup S_3 \cup S_4$

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 $multiply(cy, \lfloor (z/c) \rfloor) + y(z \ mod \ c))$

case 1: *c = z*

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

case 2: *c > z*

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

case 3: *c* < *z*

Assumptions:
$c \geq 2$
$z \leq n$ (inductive hypothesis)
$y \geq 0$

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

Lemma: we show that

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

by the quotient remainder theorem

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

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

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

Now,

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

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:

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

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

$$= cy\lfloor ((z^{'})/c) \rfloor + y((z^{'})\ mod\ c)$$
$$= y(c\lfloor (z^{'}/c) \rfloor + (z^{'}\ mod\ c))$$

and from (1*)

$$= yz^{'}$$
$$= y(n+1)$$
$$\blacksquare$$

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

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

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 algoritm satisfies.
Assuming the algorithm holds for polyomials of degree $\leq n$ and that A is an ordered set of coefficients of siz
$horner(A, x) = P(x) = a_n^{x^n} + a_{n-1}x^{n-1} + ... + a_1x + a_0$
we must show it holds for polynomials of degree $n + 1$, ie: $A^{'}$ is an ordered set of coefficients of size
$n + 1$, $[A_{n+1}, A_n, ...A_0]$

horner($A^{'}, x)$:

$$p \Rightarrow$$
$$A^{'}_{n+1}$$
$$A^{'}_{n+1} * x + A^{'}_n$$
$$A^{'}_{n+1} * x^2 + A^{'}_n * x + A^{'}_{n-1}$$
$$.$$
$$.$$
$$.$$
$$A^{'}_{n+1} * x^{n+1} + A^{'}_n * x^n + ... + A^{'}_1 * x + A^{'}_0$$
$$=$$
$$A^{'}_{n+1} * x^{n+1} + horner(A, x)$$

■ *??? 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*

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

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

case 2: *a < b*

$$i = 2$$
$$j = 1$$
$$\text{if } (A[1] > A[2]) = \ false$$
$$\text{swap values, so } A = [b, a]$$
$$i = 1$$
$$j = 1$$
$$\text{if } (A[1] > A[2]) = \ false$$

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 rearange its items
Now we procede 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 $(A : list[1...n, n + 1])$

$$\text{var } \ inti, j$$
$$\text{for } \ ifromn + 1to1$$
$$\text{for } \ jfrom1toi - 1$$
$$\text{if } \ (A[j] > A[j + 1])$$
$$\text{swap the values of } \ A[j]andA[j + 1]$$

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. ■