COL 100 . Assignment 5

KSHITIJ AGRAWADHI . 2019EE30577.

Q1. Proof of correctness for merge sort.

(Refer to sm2 file for algo).

① fun. Splitter splits a list into two equal (if even elements) else 2 lists with one extra element in l_1.

Basis: if $ls = []$, split $([]) = ([], [])$

I.H.: Let the size traversed till now be n, $0 \leq n \leq N$

$split(ls) = (l_1, l_2)$.

I.S: Now let the function encounter two more elements in the list, let the counter 'i' be currently even. Therefore, by algo,

$xs = x_1 :: x_2 :: ls$

$split(xs) = (x_1 :: l_1, x_2 :: l_2)$

$split(xs) = (x_1', x_2')$ still holds.

Hence proved (split function).

(2) Next we encounter the fun. merge which merges two sorted lists.

Basis :

In case of of any of the 2 lists empty, non empty list output, of singleton

merge(x::[], y::[]) =
    if x ≤ y → [x, y]
    else [y, x].

I.H. : let
    0 < size(l_1), size(l_2) ≤ N.
merge(l_1, l_2) gives a sorted list of size(l_1) + size(l_2)

I.S : let WLOG size(l_1) = N+1.

merge(l_1, l_2) = merge(x::xs, y::ys)

Assume x ≤ y.

⇒ x::merge(xs, y::ys)

Size increased by 1.

Here true by I.H.

Similar proof by taking size(l_2) as induction parameter.

(3) Now for mergesort.

Basis: If $l_1 = [\,]$, $msort(l_1) = [\,]$.

If $l_1 = [x]$, $msort(l_1) = [x]$.

I.H: let $0 \leq size(l_1) \leq N$

$msort(l_1)$ returns a sorted list of $size(l_1)$.

I.S: let size of $l_1 = N+1$.

$msort(l_1) = msort(x :: xs)$

$= merge(msort(p_1), msort(p_2))$

$(p_1, p_2) = split(l_1)$

as $size(p_1), size(p_2) \leq N$.

$msort(p_1)$, $msort(p_2)$ gives 2 sorted lists by I.H.

and merge ( ) gives a sorted list as proved in (2).

also $size(p_1) + size(p_2)$

$= size(merge(msort(p_1), msort(p_2)))$

Hence proved by inducting on size of l_1.

Q3. a) Addition of 2 numbers:

The algorithm is basically emulating the same adding procedure we learn in schools where we carry over a 1 when the sum of digits exceed 10 and write the remainder.

let $a = [a_0, a_1, a_2 \ldots a_{N-1}]$

$b = [b_0, b_1, \ldots b_{M-1}]$.

and let $x = [x_0, x_1 \ldots x_{i-1}]$ contain the sum till i^{th} digits.

Dividing the sum into three parts:

$(\Sigma)_{0 \to i-1} + (\ )_{i} + (\Sigma)_{(i+1) \to \max(M,N)-1}$.

Basis: When $i=0$,
x is empty.

I.H.: let at $i=i-1$.
x contain the sum of first i digits and carry overs have been adjusted.

I.S.: at i^{th} posn.

Case I: if $a_i + b_i < 10$.

then digit at that place is updated by $(a_i + b_i)$.

Case II: if $a_i + b_i > 10$.

let $a_i + b_i = 10p + q$.

$p = (a_i + b_i)$ div 10

$q = (a_i + b_i)$ mod 10.

We update the digit at that place by q and carry over p at the next digit as $10p + q$ is still $a_i + b_i$.

Hence, I.H. still holds.

∴ add (a, b) → gives the sum of 2 no.s.

Q3. b) Subtraction of 2 numbers.

This algorithm as well is emulating the same subtraction procedure we learn in schools, where we subtract the 2 digits and if the subtraction is less than 0, we carry over a 10 from the next digit.

let $a = [a_0, a_1 \_ \_ \_ a_{N-1}]$

$b = [b_0, b_1 \_ \_ \_ b_{M-1}]$

and let x contain the subtraction till (i-1) digits.

Basis: when i=0,
x is empty.

I.H.: let till (i-1) digits,
x contains the subtraction with carry overs adjusted and accounted for.

I.S : at i^{th} posn.

Case I : If $a_i > b_i$.

then digit at that place is simply replaced by $(a_i - b_i)$.

Case II : If $a_i < b_i$.

we carry over a 1 from adjacent place and add 10 to it.

$\ldots a_{i+1}\ a_i \ldots$

$(\ldots b_{i+1}\ b_i \ldots)$.

$(10a_{i+1} + a_i) - (10b_{i+1} + b_i)$

~~$= 10(a_{i+1} - 2) + a_i$~~

$= 10(a_{i+1} - b_{i+1}) + (a_i - b_i)$

$= 10((a_{i+1} - 1) - b_{i+1}) + (a_i + 10 - b_i)$.

Hence, induction hypothesis still holds.

Looking only at these 4 digits as the previous part is accounted for. and rest is not in picture yet in iterative version.

Q3. c) Multiplication of 2 integers.

Defined a function update (l) which updates the list to make sure all the digits are +ve and less than 10 during operation.

Also, a representation:
the list being multiplied represented as I(l).

Claim: I(list) = I(update(list))

Basis: update([]) = []

update([x]) =
if $x \geq 10$ then (x mod 10) :: (x div 10)

let $x = 10a + b$.

$I(x) = 10a + b$

$I(update[x]) = b + 10a = I(x)$

I.H.: let for n size(list) $\leq N$

I(list) = I(update(list))

I.S.: size(ls) = N+1.

let ls = $[a_0, a_1, \ldots, a_N]$

update (x :: y :: xs) =

(I) if $x \geq 10$ then (x mod 10) :: update ((y + x div 10) :: xs)

I (update (xs))

$a_0 = 10a + b$.

$= b + 10(a_1 + a) + \sum_{i=2}^{n} a_i 10^i$

$= a_0 + 10 a_1 + \sum_{i=2}^{n} a_i 10^i$

$= I(x)$

(II) if $x < 10$ then $x :: update(y :: xs)$.

$= I(x)$ by I.H.

Hence proved.

Now for multiply (ls_1, ls_2)

= multiply_iter $(ls_1, ls_2, [\,], 0)$.

where the functions,

zero(n) = [0, 0 - - - 0] → n zeroes.

mul (x :: xs, y :: ys) →.

let $y = [y_0, y_1, - - - y_n]$

$\Rightarrow [xy_0, xy_1, - - - xy_n]$.

Now, proof for the iter part.

let, $ls_1 = \sum_{i=0}^{N} y_i 10^i$

$ls_2 = \sum_{i=0}^{M} z_i 10^i$

INV at $n = n_0$.

$$b = \left(\sum_{i=0}^{n_0} y_i 10^i \right) \left(\sum_{i=0}^{M} z_i 10^i \right)$$

thus at $n = N$, $b = I(ls_1) \times I(ls_2)$.

Basis: If size$(ls_1) = 0$ then b is trivial for empty list $ls_1 = [\,]$.

multiply$([\,], ls_2)$ = mul$([\,], ls_2)$

$= 0 \cdot \sum_{i=0}^{M} z_i 10^i = 0 \cdot I(ls_2)$

$= I(ls_1) \cdot I(ls_2)$

Hence basis holds.

I.H: let for size$(ls_1) \le N$

multiply$(ls_1, ls_2) = I(ls_1) \cdot I(ls_2)$

I.S : let size(ls_2) = N+1 .

multiply(ls_1, ls_2) $ls_1 = y :: ys$.

= iter(ls_1, ls_2, [], 0)

= iter(ys, ls_2, add([], mul(ls_1, ls_2), 1))

size of ys = N hence holds by I.H.

:= I(ls_1) . I(ls_2) .

Hence proved.

Here we inducted on the length of ls_1.

Basically what the algorithm is doing is multiplying 2 numbers the way we manually do (taught in preschool).

and the putzero(n) function is accounting for the zeroes in each step.