

# Problem set 2

Hantang Li, Kaiqu Liang

October 2018

## 1 Question 1

a)

a: 1, since  $*$  is the only element which has 0 left parenthesis

b: 1, since  $(**)$  is the only element which has 1 left parenthesis

c: 2, they are  $((**)*)$  and  $(*(**))$

d: 5, they are  $((**)(**))$ ,  $(((**)*)$ ,  $(*(**(**)))$ ,  $(((**)*)$  and  $((**)(**))$

e: 14, let the element represent by  $(e_1, e_2)$

there are 5 ways if  $e_1$  has 0 parenthesis

there are 2 ways if  $e_1$  has 1 parenthesis

there are 2 ways if  $e_1$  has 2 parenthesis

there are 5 ways if  $e_1$  has 3 parenthesis

Sum these ways together, we get 14

b)

Let  $c : \mathbb{N} \Rightarrow \mathbb{N}$  be the function

I define

$$c(n) = \begin{cases} 1 & n=0 \\ \sum_{i=0}^{n-1} c(i)c(n-i-1) & \text{if } n > 1, n \text{ even} \\ \sum_{i=0}^{n-1} c(i)c(n-i-1) - c(\frac{n-1}{2}) & \text{if } n > 1, n \text{ odd} \end{cases}$$

Explain:

The base case is  $n = 0$ ,  $c(n) = 1$ ,

since it cannot be decompose to any smaller problem.

For  $n \geq 1$ , we can decompose the final result to two parts.

Let  $e_1 e_2 \in T$ ,

any new elements can be construct by  $(e_1 e_2)$ .

Since we want  $(e_1 e_2)$  has  $n$  left parenthesis,

$e_1 e_2$  together must have  $n - 1$  parenthesis.

Thus there are  $n - 1$  possible number of left parenthesis  $e_1$  can have,

let  $i$  denote number of  $e_1$ 's parenthesis,

$e_2$  has  $(n - 1 - i)$  number of left parenthesis for each  $i$ .

Since by definition of function  $c$ ,

$c(i)$  is number of different element with  $i$  left parenthesis

We sum all possible ways of combining  $e_1 e_2$  together and get

$$\sum_{i=0}^{i=n-1} c(i)c(n-i-1).$$

However, for odd  $n$ ,  $i$  can be equal to  $n-1-i$  when  $i = \frac{n-1}{2}$ ,  
each different element in  $c(\frac{n-1}{2})$  is counted twice,  
therefore we derived the equation  
 $\sum_{i=0}^{i=n-1} c(i)c(n-i-1) - c(\frac{n-1}{2})$  when  $n$  is odd

## 2 Question 2

a)

We can think the problem as number of ways to add one stamp to a postage with less than  $n$  cents, By thinking this way, we can define the function recursively.

let  $p : \mathbb{N} \Rightarrow \mathbb{N}$  be the function,

let  $n$  be a natural number such that  $n > 12$ .

There are three ways to add one stamp to create postage worth  $n$   
so first define

$$p(n) \begin{cases} 0 & n=0 \\ 1 & 1 \leq n < 8 \\ 2 & 8 \leq n < 12 \\ 3 & n=12 \\ p(n) = p(n-3) + p(n-4) + p(n-5) & n \geq 13 \end{cases}$$

But, there are ways that counted twice.

For ways created by adding one stamp that worth 4 or 5 can be counted by  $p(n-4)$ ,  $p(n-5)$ .

since  $n > 12$ ,

both  $p(n-5)$ ,  $p(n-4)$  contains ways by adding a stamp that worth 5 or 4 to ways counted by  $p(n-9)$

so,  $p(n-9)$  is counted twice.

Same reasoning can be used in  $p(n-3)$ ,  $p(n-4)$ ,

which  $p(n-7)$  is counted twice

and  $p(n-3)$ ,  $p(n-5)$ , which  $p(n-8)$  is counted twice.

And then, the function becomes

$$p(n) \begin{cases} 0 & n=0 \\ 1 & 1 \leq n < 8 \\ 2 & 8 \leq n < 12 \\ 3 & n=12 \\ p(n) = p(n-3) + p(n-4) + p(n-5) - p(n-7) - p(n-8) - p(n-9) & n \geq 13 \end{cases}$$

However,  $p(n-3)$ ,  $p(n-4)$ ,  $p(n-5)$  all contains ways that created by adding two stamps to ways counted by  $p(n-3-4-5) = p(n-12)$ ,

so  $p(n - 12)$  is counted three times.

But  $p(n - 12)$  is also subtract 3 times since  $p(n - 7)$ ,  $p(n - 8)$ ,  $p(n - 9)$  all contains ways that created by adding one stamp to ways counted by  $p(n - 3 - 4 - 5) = p(n - 12)$ .

Thus  $p(n - 12)$  is counted zero times, we need to add it back.

Then, we get the finalized formula

$$p(n) \begin{cases} 0 & n=0 \\ 1 & 1 \leq n < 8 \\ 2 & 8 \leq n < 12 \\ 3 & n=12 \\ p(n) = p(n-3) + p(n-4) + p(n-5) - p(n-7) - p(n-8) - p(n-9) \\ \quad + p(n-12) & n \geq 13 \end{cases}$$

b)

Proof:

To prove  $p(n)$  is monotonic non decreasing on  $\mathbb{N}^+$ ,

I first want to prove a lemma.

Which is  $\forall n \in \mathbb{N}, n \geq 8 \implies p(n) - p(n - 4) \geq p(n - 3) - p(n - 7)$ .

Define predicate  $Q(n)$ :  $p(n) - p(n - 4) \geq p(n - 3) - p(n - 7)$ ,

I will prove  $\forall n \in \mathbb{N}, n \geq 8 \Rightarrow Q(n)$  holds by complete induction.

Assume  $Q(k)$  holds for all integer  $k$  such that  $8 \leq k < n$ .

case 1:  $8 \leq n \leq 12$

When  $n = 8$ ,  $p(8) - p(8 - 4) = 2 - 1 = 1$

$p(8 - 3) - p(8 - 7) = 1 - 1 = 0$

since  $1 > 0$ ,  $Q(8)$  holds.

When  $n = 9$ ,  $p(9) - p(9 - 4) = 2 - 1 = 1$

$p(9 - 3) - p(9 - 7) = 1 - 1 = 0$

since  $1 > 0$ ,  $Q(9)$  holds.

When  $n = 10$ ,  $p(10) - p(10 - 4) = 2 - 1 = 1$

$p(10 - 3) - p(10 - 7) = 1 - 1 = 0$

since  $1 > 0$ ,  $Q(10)$  holds.

When  $n = 11$ ,  $p(11) - p(11 - 4) = 2 - 1 = 1$

$p(11 - 3) - p(11 - 7) = 2 - 1 = 1$

since  $1 \geq 1$ ,  $Q(11)$  holds.

When  $n = 12$ ,  $p(12) - p(12 - 4) = 3 - 2 = 1$

$p(12 - 3) - p(12 - 7) = 2 - 1 = 1$

since  $1 \geq 1$ ,  $Q(12)$  holds.

case 2:  $n > 12$

We can show  $p(n) - p(n - 4) - (p(n - 3) - p(n - 7)) \geq 0$  instead.

Since  $n > 12$ , by definition of  $p(n)$ ,

$p(n) = p(n - 3) + p(n - 4) + p(n - 5) - p(n - 7) - p(n - 8) - p(n - 9) + p(n - 12)$

By subtracting  $p(n-4)$  first and then subtracting  $p(n-3) - p(n-7)$  on both side of the equation, we get the following equation:

$$\begin{aligned} & p(n) - p(n-4) - (p(n-3) - p(n-7)) \\ &= p(n-3) + p(n-4) + p(n-5) - p(n-7) - p(n-8) - p(n-9) \\ &+ p(n-12) - p(n-4) - (p(n-3) - p(n-7)) \\ &= p(n-5) - p(n-8) - p(n-9) + p(n-12) \\ &= (p(n-5) - p(n-9)) - (p(n-9) + p(n-12)) \end{aligned}$$

Since  $n > 12$ ,  $8 \leq n-5 < n$ ,

by induction hypothesis,  $Q(n-5)$  holds

Therefore  $(p(n-5) - p(n-9)) - (p(n-9) + p(n-12)) \geq 0$

Then  $p(n) - p(n-4) - (p(n-3) - p(n-7)) \geq 0$ .

Thus I have proven  $\forall n \in \mathbb{N} \ n \geq 8 \Rightarrow Q(n)$

prove monotonic non decreasing:

Let predicate  $T(n)$  be  $\forall m \in \mathbb{N}^+, m < n \implies p(m) \leq p(n)$

proof: Let  $n \in \mathbb{N}^+$  I will prove  $\forall n, T(n)$  holds by complete induction

Assume  $\forall k \in \mathbb{N}^+, 8 \leq k < n \implies T(k)$  holds, want to show  $T(n)$  holds

case 1:  $n \leq 13$  then from the definition of function  $p$

for  $1 \leq n \leq 7$ ,  $p(n) = 1$ ,

for  $8 \leq n \leq 11$ ,  $p(n) = 2$ ,

for  $n = 12$ ,  $p(n) = 3$ ,

for  $n = 13$ ,  $p(n) = 3$ ,

We can clearly see that  $p(n)$  is non decreasing when  $1 \leq n \leq 13$

case 2:  $n > 13$

Since  $n > 13$ , we know that  $n > n-1 \geq 1$  and  $n \geq 14$ ,

by inductive hypothesis,  $T(n-1)$  holds and by transitivity of  $\leq$ ,

we only need to show  $p(n-1) \leq p(n)$

Since  $n \geq 14$ , we know  $n \geq 13$ , By definition of  $p(n)$ ,

$$p(n) = p(n-3) + p(n-4) + p(n-5) - p(n-7) - p(n-8) - p(n-9) + p(n-12)$$

Since  $n-1 \geq 13$ , by definition of  $p(n)$ .

$$p(n-1) = p(n-4) + p(n-5) + p(n-6) - p(n-8) - p(n-9) - p(n-10) + p(n-13)$$

By subtracting  $p(n)$  and  $p(n-1)$ , we get the following equation:

$$\begin{aligned} & p(n) - p(n-1) = \\ & (p(n-3) + p(n-4) + p(n-5) - p(n-7) - p(n-8) - p(n-9) + p(n-12)) - \\ & (p(n-4) + p(n-5) + p(n-6) - p(n-8) - p(n-9) - p(n-10) + p(n-13)) \end{aligned}$$

$$= p(n-3) - p(n-6) - p(n-7) + p(n-10) + p(n-12) - p(n-13)$$

$$= (p(n-3) - p(n-7)) - (p(n-6) + p(n-10)) + p(n-12) - p(n-13)$$

$$\geq (p(n-3) - p(n-7)) - (p(n-6) + p(n-10)) + p(n-12) - p(n-13)$$

(Since  $n > 13$ ,  $n-3 > 8$ , by  $Q(n-3)$ ,  $p(n-3) - p(n-7) - (p(n-6) + p(n-10)) \geq 0$ )

$$\geq 0 + p(n-12) - p(n-13)$$

$$\geq 0$$

(Since  $n > 13$ , we have  $n > n - 12 \geq 1$ , by inductive hypothesis, we know  $T(n - 12)$  holds, therefore  $p(n - 12) \geq p(n - 13)$ )

Since  $p(n) - p(n - 1) \geq 0$ ,  $p(n) \geq p(n - 1)$ ,  
we can get when  $n > 13$ ,  $T(n)$  holds.

By case 1 and case 2, by using complete induction, we have  $\forall n, T(n)$  holds.

### 3 Question 3

a)

proof: Let  $P(n)$  be the following predicate:

$$P(n) : \forall m \in \mathbb{N}^+, m < n \implies T(m) < T(n)$$

We use complete induction to prove that  $\forall n \in \mathbb{N}^+, P(n)$ .

Inductive step: Let  $n \in \mathbb{N}^+$ , Assume  $P(j)$  holds for all integer  $j$   
such that  $1 \leq j \leq n$  we must prove that  $P(n)$  holds as well.

Base case1:  $n \leq 2$

when  $n = 1$ ,  $P(n)$  holds trivially.

Since there is no positive integer  $m < 1$ .

When  $n = 2$ ,  $T(1) = c'$ ,  $T(2) = 1 + T(\lceil \frac{2}{2} \rceil) = 1 + T(1) = 1 + c'$

Therefore  $T(2) > T(1)$

Hence  $P(2)$  holds

case2:  $n > 2$

since  $n > 2$ , we know that  $1 \leq n - 1 < n$

So, by our induction hypothesis, we know that  $P(n - 1)$  holds and (by transitivity of  $\leq$ )

We need only show that  $T(n - 1) \leq T(n)$ .

$$\begin{aligned} T(n - 1) &= 1 + T(\lceil \frac{n - 1}{2} \rceil) \text{ since } n - 1 > 1 \\ &\leq 1 + T(\lceil \frac{n}{2} \rceil) \end{aligned}$$

(Since  $1 \leq \lceil \frac{n - 1}{2} \rceil < \lceil \frac{n}{2} \rceil < n$ . By IH,  $P(\lceil \frac{n}{2} \rceil)$  holds, hence  $T(\lceil \frac{n - 1}{2} \rceil) < T(\lceil \frac{n}{2} \rceil)$ )

$$= 1 + T(n)$$

b) proof:

I want to show  $\forall k, n \in \mathbb{N}, n = 2^k \implies T(n) = \log_2 n + c'$

That is equivalent to show  $\forall k \in \mathbb{N}, T(2^k) = \log_2(2^k) + c' = k + c'$

Define  $P(k) : T(2^k) = k + c'$

I want to show  $\forall k, n \in \mathbb{N} P(k)$  by simple induction on  $k$ .

Base case:

$$T(2^0) = T(1) = c'$$

$$k + c' = 0 + c' = c'$$

Hence  $T(2^0) = 0 * k + c'$ ,  $P(0)$  holds.

Inductive step:

Let  $k, n \in \mathbb{N}$  assume  $P(k)$ .

That is  $T(2^k) = k + c'$

I will show that  $P(k + 1)$  follows

$$\begin{aligned} T(2^{k+1}) &= 1 + T(\lceil \frac{2^{k+1}}{2} \rceil) \text{ (since } 2^{k+1} > 1) \\ &= 1 + T(2^k) \\ &= 1 + k + c' \end{aligned}$$

Therefore  $P(k + 1)$  holds

By base case and inductive step,

I can conclude that  $\forall k, n \in \mathbb{N}, P(k)$

Hence  $\forall k, n \in \mathbb{N}, n = 2^k \implies T(n) = \log_2 n + c'$

c)

Proof:

First I want to show  $T \in O(\lg n)$

That is to show  $\exists c_0, n_0 \in \mathbb{N}^+$

s.t  $\forall n \in \mathbb{N}, n \geq n_0 \implies T(n) \leq c_0 \lg n$

Take  $c_0 = c' + 2, n_0 = 2$

Let  $n \in \mathbb{N}^+$

Assume  $n \geq n_0$ , hence  $n \geq 2$

Let  $m = 2^{\lceil \log_2 n \rceil}$

Since  $\lceil \log_2 n \rceil - 1 \leq \log_2 n \leq \lceil \log_2 n \rceil$

we have  $\frac{m}{2} < n \leq m$

Then

$$\begin{aligned} T(n) &\leq T(m) \text{ (since T non decreasing)} \\ &= \lceil \log_2^n \rceil + c' \text{ (by part b conclusion)} \\ &\leq \log_2^n + 1 + c' \\ &\leq \log_2^n + (1 + c') \log_2^n \text{ (since } n \geq 2, \log_2 n \geq 1) \\ &= (c' + 2) \log_2^n \\ &= c_0 \log_2 n \end{aligned}$$

Therefore,  $T \in O(\lg n)$

Then I want to show  $T \in \Omega(\log_2 n)$

That is to show  $\exists G, n_1 \in \mathbb{N}^+, \forall n \in \mathbb{N}, n \geq n_1 \implies T(n) \geq G \log_2^n$

Take  $G = 1, n_1 = 2$

Let  $n \in \mathbb{N}^+$  Assume  $n \geq n_1$ , that is  $n \geq 2$

Let  $m = 2^{\lceil \log_2 n \rceil}$

$m = 2^{\lceil \log_2 n \rceil} \leq 2^{\log_2 n} = n$

Since T is non decreasing

$$\begin{aligned}
T(n) &\geq T(m) = T(2^{\lceil \log_2 n \rceil}) \\
&= T(m) = \lceil \log_2 n \rceil + c' \text{ (by part b conclusion)} \\
&\geq \log_2 n - 1 + c' \\
&\geq \log_2 n \text{ (since } c' \geq 1) \\
&= G \log_2 n
\end{aligned}$$

Therefore  $T \in \Omega(\log_2 n)$

Hence by  $T \in O(\lg n)$  and  $T \in \Omega(\lg n)$ ,  $T \in \Theta(\lg n)$

## 4 Question 4

Let  $s1$   $s2$  be arbitrary strings.

Define predicate  $P(j) : \text{if } \forall i, 0 \leq i \leq \text{len}(s1) \text{ and } 0 \leq j \leq \text{len}(s2) \text{ then when } \text{count\_subsequences}(s1, s2, i, j) \text{ is called, this call terminates and return number of times } s1[:i] \text{ occurs as a subsequence of } s2[:j]$

Let  $j$  be arbitrary integer  $\geq 0$ , assume  $0 \leq j \leq \text{len}(s2)$  otherwise  $P(j)$  vacuously true. Assume  $\forall k \in \mathbb{N}, 0 \leq k < j, P(k)$  holds. Want to show  $P(j)$  holds. I will prove this by complete induction.

Case 1:  $j = 0$  Let  $i \in \mathbb{N}$  assume  $0 \leq i \leq \text{len}(s1)$ ,

Sub case 1:  $i = 0$ , then the first if branch executes, and the call terminates by return 1

Since  $s1[:0]$  and  $s2[:0]$  are empty strings.

There is only one time  $s1[:0]$  occurs as subsequence of  $s2[:0]$ .

Sub case 2:  $i > j = 0$  then goes in the second if branch and terminates by return 0.

Since  $\text{len}(s1[:i]) > \text{len}(s1[:0])$ ,  $s1[:i]$  cannot be any subsequence of  $s2[:0]$ . Since in all cases the function can terminates and return correct value,  $P(0)$  holds.

Case 2:  $j > 0$ , let  $i \in \mathbb{N}$  assume  $0 \leq i \leq \text{len}(s1)$

Sub case 1:  $i = 0$ , then the first if branch executes, and the call terminates by return 1.

Since  $s1[:0]$  is an empty string,

there is only one time  $s1[:0]$  occurs as subsequence of any string.

Sub case 2:  $i > j$ , then goes into second if branch and terminates by return 0.

Since  $\text{len}(s1[:i]) > \text{len}(s2[:j])$ ,

$s1[:i]$  cannot be any subsequence of  $s2[:0]$ .

Sub case 3:  $0 < i \leq j$ ,

If  $s1[i-1] \neq s2[j-1]$ , then the third if branch executed

and call  $\text{count\_subsequences}(s1, s2, I, j - 1)$   
 Since  $0 < j$ ,  $0 \leq j - 1 < j$ , by IH  $p(j-1)$  follows,  
 so the call can terminate.  
 Show the return value is correct:  
 since by definition of subsequence,  
 if the element in last index of  $s1[: i]$  does not match with last index in  $s2[: j]$ ,  
 we should check if that element is in  $s2[: j - 1]$ .  
 Since we have to find a match for every elements in  $s1[: i]$   
 and they must be in the same order as they were in  $s1[: i]$ .  
 Which the return value is exactly what  
 $\text{count\_subsequences}(s1, s2, i, j - 1)$  returns.  
 If  $s1[i - 1] = s2[j - 1]$ , then, the last it branch executed,  
 and call  $\text{count\_subsequences}(s1, s2, i, j - 1)$  and  
 $\text{count\_subsequences}(s1, s2, i - 1, j - 1)$   
 Since by the case's condition we know  $0 < j$  so  $0 \leq j - 1 < j$ ,  
 Therefore, by IH,  $p(j-1)$  follows,  
 So the call can terminate.  
 Show return value is correct:  
 By definition of subsequence,  
 if we find the last index of  $s1[: i]$  and  $s2[: j]$  match,  
 then we should find if  $s1[: i - 1]$  is a subsequence of  $s2[: j - 1]$ ,  
 and count the number of  $s1[: i - 1]$  as a subsequence of  $s2[: j - 1]$ .  
 By  $p(j - 1)$ ,  
 $\text{count\_subsequences}(s1, s2, i - 1, j - 1)$  would return the correct value,  
 denote  $v1$ .  
 We should also check if  $s1[: i]$  is a subsequence of  $s2[: j - 1]$ ,  
 and count the occurrence.  
 By  $p(j-1)$ , count subsequences  $(s1, s2, i, j-1)$   
 Would return the correct value.  
 denot  $v2$ .  
 Thus, by sum those two correct value  $v1, v2$ , the call returns correct value.  
 Hence, by summing all sub cases of  $i$ ,  $P(j)$  is true for  $j > 0$   
 Therefore  $P(j)$  is true for all nature number  $j$

## 5 Question 5

Proof: denote integer variables  $blue$ ,  $green$ ,  $red$  at the end of each iteration  
be  $blue_i, green_i, red_i$

Also denote *colour\_list* as  $L$

Define  $P(i)$  : at the end of loop iteration  $i$ , if occurs,

then  $0 \leq blue_i \leq green_i \leq red_i \leq len(L)$

and  $L[0 : green_i] + L[0 : red_i]$  same colours as before

and  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

and  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$

and  $all([c == "r" \text{ for } c \text{ in } L[red_i :]])$

I will prove  $\forall i \in \mathbb{N}, P(i)$  by simple induction.

Base case:  $i = 0$

The loop won't iterate once,

$blue_i = green_i = 0, red_i = len(L)$

Clearly  $0 \leq blue_i \leq green_i \leq red_i \leq len(L)$ .

Since  $L[0 : green_i]$  and  $L[0 : red_i]$  are empty lists

so,  $L[0 : green_i] + L[0 : red_i]$  vacuously same colours as before.

And since  $L[0 : blue_i]$  and  $L[blue_i : green_i]$  and  $L[red_i : ]$

are empty string,

any universal quantifier on them is vacuously true,

thus  $P(0)$  holds.

Inductive Step:

Let  $i \in \mathbb{N}$ , assume  $P(i)$

will show  $P(i + 1)$  follows, if there is  $i + 1$  th loop

then, from the while loop's condition since  $green_i < red_i$

and by  $P(i)$ ,  $0 \leq green_i < red_i \leq len(L)$ , so  $green_i$  is valid index

Sub case1:  $L[green_i] = "b"$

By the code

$green_{i+1} = green_i + 1$ ,

$blue_{i+1} = blue_i + 1$ ,

$red_{i+1} = red_i$

These variables are nature numbers

since nature numbers are closed under addition

thus adding one to an integer is still an integer.

By while loop's condition  $green_i < red_i$ , so  $green_i + 1 \leq red_i$

thus  $green_{i+1} \leq red_{i+1}$ , since  $red_{i+1} = red_i$

by IH  $0 \leq blue_i \leq green_i$  so  $0 \leq blue_i + 1 \leq green_i + 1$

thus  $0 \leq blue_{i+1} \leq green_{i+1}$

Therefore  $0 \leq blue_{i+1} \leq green_{i+1} \leq red_{i+1} \leq len(L)$

By IH,  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

since by sub case's condition  $L[green_i] = "b"$

the code swap  $L[green_i], L[blue_i]$ 's element

so  $L[blue_i] = "b"$ .

Thus  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i + 1]])$

Hence  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_{i+1}])]$

By IH  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$ ,  $L[blue_i] = "g"$

we swap  $L[blue_i]$  with  $L[green_i]$ ,  $L[green_i] = "g"$

Thus  $all([c == "g" \text{ for } c \text{ in } L[blue_i + 1 : green_i + 1]])$

by  $green_{i+1} = green_i + 1$  and  $blue_{i+1} = blue_i + 1$ ,

Hence  $all([c == "g" \text{ for } c \text{ in } L[blue_{i+1} : green_{i+1}])]$

By IH  $all([c == "r" \text{ for } c \text{ in } L[red_i :]])$ ,

since  $red_{i+1} = red_i$

Hence  $all([c == "r" \text{ for } c \text{ in } L[red_{i+1} :]])$ ,

By IH  $L[0 : green_i] + L[0 : red_i]$  same colour as before

We know  $L[0 : green_i] = L[0 : blue_i] + L[blue_i : green_i]$

$L[0 : green_{i+1}] = L[0 : green_i + 1] = L[0 : blue_i + 1] + L[blue_i + 1 : green_i + 1]$

colours of  $L[0 : blue_i + 1] = L[0 : blue_i] + L[blue_i]$

since  $L[blue_i] = "b"$  and  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

colours of  $L[0 : blue_i + 1]$  same as before

by previous deduction we know

$all([c == "g" \text{ for } c \text{ in } L[blue_i + 1 : green_i + 1]])$

and by IH  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$

so  $L[blue_i + 1 : green_i + 1]$  same colour as before.

Hence  $L[0 : green_{i+1}]$  same colour as before.

Also by previous deduction  $red_{i+1} = red_i$

So,  $L[0 : green_{i+1}] + L[0 : red_{i+1}]$  same colour as before

Thus  $P(i + 1)$  hold in this case

Sub case2:  $L[green_i] = "r"$

By the code

$green_{i+1} = green_i$ ,

$blue_{i+1} = blue_i$ ,

$red_{i+1} = red_i - 1$

These variables are nature numbers

since nature numbers are closed under addition

thus adding one to an integer is still an integer.

By while loop's condition  $green_i < red_i$ , so  $green_i \leq red_i - 1$

thus  $green_{i+1} \leq red_{i+1}$

By IH  $red_i \leq len(L)$

so  $red_i - 1 \leq len(L)$

since  $blue_{i+1}$  and  $green_{i+1}$  unchanged and we know  $red_{i+1} = red_i - 1$

Therefore  $0 \leq blue_{i+1} \leq green_{i+1} \leq red_{i+1} \leq len(L)$

By IH,  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

since  $blue_{i+1}$  unchanged

So  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_{i+1}])]$

By IH  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$ ,  $L[blue_i] = "g"$

since  $green_{i+1}$  unchanged

So  $all([c == "g" \text{ for } c \text{ in } L[blue_{i+1} : green_{i+1}]])$

By IH  $all([c == "r" \text{ for } c \text{ in } L[red_i :]])$ ,  
since  $red_{i+1} = red_i - 1$ , by case condition  $L[red_i - 1] = "r"$   
So  $all([c == "r" \text{ for } c \text{ in } L[red_i - 1 :]])$ ,  
Hence  $all([c == "r" \text{ for } c \text{ in } L[red_{i+1} :]])$ ,

By IH  $L[0 : green_i] + L[0 : red_i]$  same colour as before  
since  $green_{i+1}$  unchanged  
We know  $L[0 : green_i] = L[0 : green_{i+1}]$   
by IH  $all([c == "r" \text{ for } c \text{ in } L[red_i :]])$ ,  
we know  $all([c == "r" \text{ for } c \text{ in } L[red_{i+1} :]])$  from previous  
So,  $L[0 : green_{i+1}] + L[0 : red_{i+1}]$  same colour as before

Thus  $P(i + 1)$  hold in this case

Sub case 3:  $L[green_i] = "g"$

By the code

$green_{i+1} = green_i + 1$ ,

$blue_{i+1} = blue_i$ ,

$red_{i+1} = red_i$

These variables are nature numbers

since nature numbers are closed under addition

thus adding one to an integer is still an integer.

By while loop's condition  $green_i < red_i$ , so  $green_i + 1 \leq red_i$

thus  $green_{i+1} \leq red_{i+1}$

since  $blue_{i+1}$  unchanged

by IH  $0 \leq blue_i \leq green_i$  so  $0 \leq blue_i \leq green_i + 1$

thus  $0 \leq blue_{i+1} \leq green_{i+1}$

Therefore  $0 \leq blue_{i+1} \leq green_{i+1} \leq red_{i+1} \leq len(L)$

By IH,  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

since  $blue_{i+1} = blue_i$  unchanged

So  $all([c == "b" \text{ for } c \text{ in } L[0 : blue_{i+1}]])$

By IH  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$ ,  $L[green_i] = "g"$

since  $L[green_i] = "g"$ , by the condition

So  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i + 1]])$

since  $blue_{i+1} = blue_i$  unchanged

So  $all([c == "g" \text{ for } c \text{ in } L[blue_{i+1} : green_{i+1}]])$

By IH  $all([c == "r" \text{ for } c \text{ in } L[red_i :]])$ ,

since  $red_{i+1} = red_i$  unchanged

Hence  $all([c == "r" \text{ for } c \text{ in } L[red_{i+1} :]])$ ,

By IH  $L[0 : green_i] + L[0 : red_i]$  same colour as before

by IH  $all([c == "g" \text{ for } c \text{ in } L[blue_i : green_i]])$ ,

$all([c == "b" \text{ for } c \text{ in } L[0 : blue_i]])$

we derived  $all([c == "g" \text{ for } c \text{ in } L[blue_{i+1} : green_{i+1}]])$

$all([c == "b" \text{ for } c \text{ in } L[0 : blue_{i+1}]])$

So,  $L[0 : green_{i+1}] = L[0 : blue_i + 1] + L[blue_i + 1 : green_i + 1]$  same colour as before

Since  $red_{i+1} = red_i$

Therefore  $L[0 : green_{i+1}] + L[0 : red_{i+1}]$  same colour as before

Thus  $P(i + 1)$  hold in this case

Since in all cases  $P(i + 1)$  hold.

Thus we can derive that the loop invariant is always true after each loop iteration

Show the code can terminate:

For  $i \in \mathbb{N}$  if the loop is able to execute at  $i$  the iteration,

then,  $red_i - green_i > red_{i+1} - green_{i+1}$

Proof: let  $i$  be a natural number, assume the code is executable at  $i$  the iteration.

Then, by while loop's condition  $green_i < red_i$  and  $green_i$  is valid index

Case1:  $L[green_i] = "b"$  or  $L[green_i] = "g"$

By the code

$green_{i+1} = green_i + 1,$

$red_{i+1} = red_i.$

So,

$red_{i+1} - green_{i+1} = red_i - (green_i + 1)$

$red_{i+1} - green_{i+1} < red_i - green_i$

Case2:  $L[green_i] = "r"$

By the code

$green_{i+1} = green_i,$

$red_{i+1} = red_i - 1$

So,

$red_{i+1} - green_{i+1} = red_i - 1 - green_i$

$red_{i+1} - green_{i+1} < red_i - green_i$

Thus, in all cases, we have exhibited a decreasing sequence of natural numbers linked to loop iterations. The last element of this sequence has the index of the last loop iteration, so the loop terminates.