Question#2:

Part (a)

Counter example details:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Week 1 | Week2 | Week 3 | Week 4 |
| L | 10 | 1 | 10 | 10 |
| H | 100 | 50 | 5 | 1 |

With inputs above, the algorithm in the book returns the wrong answer which is still be 50+10+10=70, the actual answer is 100+1+10+10

Counter example argument:

The book’s algorithm will ignore the first week’s high stress value if the high stress value in week 2 is greater than the sum of week1+week2 low stress value. Because it always calculates the high value of next week to the low values of week1 +2. So the algorithm will output a 50+10+10 which is not an actual answer (100+1+10+10).

Part (b)

Algorithm idea:

My algorithm is something similar to the algorithm in the book, the difference is that my algorithm iterate through every week, and doesn’t jump. My algorithm start with index 0 which you can understand as same as 1 in the book (because 1 looks strange to me). first my algorithm set up a base case to get the first value in answer list in order to avoid indexOut error, then for each of the index, I check both hi>li+l(i-1) and hi>h(i-1) which give me the most possible value for this index which is current i, and store the possible value to my possible answer list, also store the state you goes to (h:1 or l:0) store in state list. In the end of the first for loop, my answer list will contain all of the best choice in each index of the list, and also the state of each index. Lastly, iterate through the state list from the back, if the index is high(1 in other word), set the previous index to 0. Finally, the answer list is my actual answer now.

Algorithm detail:

answer = [] #temporary store the value for each index of my schedule

state = [] # a array to store all a sign of what should the previous value be(if 1 set the #previous value in answer to 0)

for (int i =0;i<h.size();i++):

if i==0:

if hi>li:

answer[i]=hi

state[i]=1

else:

answer[i]=li

state[i]=0

else if hi>li+l(i-1) and hi>h(i-1):

answer[i]=hi

state[i]=1

state[i-1]=0

else:

state[i]=0

answer[i]=li

endfor

for (int j = h.size()-1;j>0;j--):

if state[j]==1:

answer[j-1]=0

finalanswer = 0

endfor

for number in answer:

finalanswer+=number

endfor

return finalanswer #answer in Integer

Proof idea:

Using the idea the dynamic programing. Proof my algorithm output a correct answer by showing that my algorithm covers all the index and give each index a proper value.

Proof detail:

In my algorithm above, the first step is to give my answer list a beginning value which is max(li,hi). Then for the all other iteration, the current index will compare with the previous index h value and the sum of the li + l(i-1). If both case satisfy, I set the current index of answer list to hi since the value of hi is high to cover the previous index and also set the state value to 1; else set the answer value to low and state value 0 which mean I am not going to set the previous value to 0. Keep following the previous statement until hit the end of the for loop. Therefore, I get a list with every index satisfies the max(li,hi,li+l(i-1)). And state list. Lastly iterate through the entire state list from the back in order to avoid two continuous h in answer, the back one has higher priority, so set the previous index value to 0 if state[i] is 1. From last step, this algorithm satisfies the requirement of high-stress work need one day zero before the work present in the question description.

Runtime Analysis:

1. Generate the possible answer list and state list: O(n)
2. Set all previous index of high value to 0: O(n)
3. Sum up values: O(n)
4. Overall: O(n)+ O(n)+ O(n)=O(n)