n HQs are due tomorrow, and you must write them on A4 sheets; you can use at most 1 sheet for any solution, i.e., no solution can span multiple sheets, and you can write multiple solution on 1 sheet, as long as no solution spills over to a new sheet. The solutions need not be in sequence, i.e., it is not necessary for the solution of HQ6 to follow that of HQ5.

You have roughly solved them (with the help of your partner) and you have a good estimate of the amount of space that you would need to write each HQ; say, li is the number of lines needed to write your solution to HQi. Further, let L denote the number of lines that you can write on any A4 sheet.

This being the deadline season, there is a crunch in the supply of A4 sheets on campus, so, you must optimize on the number of A4 sheets. You simply do not have time to really really optimize, so you start by writing the solution to HQ1 on sheet S1 and then continue the following strategy for the solutions to HQ2, HQ3, ... HQn. For any HQ, you see which of the existing sheets can fit the solution; if such a sheet exists, you write the solution on that sheet, else, you write it on a fresh sheet; let **this algorithm be called APPROX**.

**(a) Show that this strategy need not use the optimal number of sheets.**

**Example** - lets there are five given homework { HQ1, HQ2, HQ3, HQ4, HQ5} with the requirement of lines { 1, 1, 1,3, 4} and page capacity of 5 lines, APPROX will lines in sheets as follows.

Sheet A1 = { HQ1, HQ2, HQ3 }

Sheet A2 = { HQ4 }

Sheet A3  = { HQ5 }

On the other hand, with OPT or the optimal way of solving, we get sheets as follows.

Sheet A1 = { HQ1, HQ2, HQ4 }

Sheet A2 = { HQ3, HQ5 }

Hence APRROX will take more than or equal sheets compared to OPT or the optimal way of solving.

**OPT <= APPROX SHEETS**

**(b) Prove that, at most, 1 sheet can be more than half-empty.**

Let's Assume that we { HQ1, HQ2 …HQx… HQn} with the requirement of lines { l1, l2 …lx… ln} is filled with sheets { A1, A2 …Ay… Ak};

let us assume up to **xth** homework is completed ( x <= n ) with **y** number of sheets( y<=k ); let us assume that **yth**sheet is more than half-empty, and for **(x+1)th** homework, another sheet **(y+1)th** is created, which is again **more than half-empty**, But this will **contradict the working principle of APPROX algorithm**. Since our algorithm starts its search at A1,... If any sheet can fulfill the criterion, then that sheet will be filled first, so according to the assumption, sheet Ay will be filled first; there will never create a sheet Ay+1 that is more than half-empty.

**(c) Prove that the optimal number of sheets >= V where V = ceil((l1 + l2 + ... + ln)/L).**

Assuming a new algorithm, **FractionalHomeWork**, which can split the homework into **two sheets**( Like a fractional knapsack ). The new algorithm will ensure the complete filling of a sheet before creating a new one, ensuring minimum use of sheets.

*Sheets require in FractionalHomeWork*

*=ceil(( Total lines of HomeWork ) / ( Single Sheet Capacity))*

*[ ceil because the remaining line requires a new sheet ]*

**V = ceil((l1 + l2 + ... + ln)/L)**.Where V is the minimum number of sheets.

Since OPT will **not divide** the homeworkinto multiple pages ( like 0/1 knapsack ), **unutilized** lines may exist on completing any homework, which may increase the number of pages.

So,

***V <= OPT***

**( d ) Let k denote the number of sheets used by you. Derive a lower bound on k using some expression of V.**

Let k be the number of sheets required by the APPROX algorithm. Since APPROX **cannot** **divide** homework into multiple sheets, it will need more than equal pages compared to FractionalHomeWork.

So,

***V <= k*** => ***ceil((l1 + l2 + ... + ln)/L) <= k***

**( e ) Derive an upper bound on k using some expression of V.**

Using statement (a), we know that up to one page is more than half-empty.

Let us say the last sheet is more than half-empty. So we can say (k-1) lines are filled with at least (L/2) lines, so there are at least (k-1)(L/2) lines that are less than or equal to the total number of lines.

i.e.

( k - 1 )( L / 2) <= (l1 + l2 + ... + ln)

( k - 1) /2 <= ( (l1 + l2 + ... + ln)/L) [ dividing both sides by L ]

( k - 1 ) / 2 <= V [ V = ceil((l1 + l2 + ... + ln)/L) ]

k/2 <= V+1

***k <= 2\*V +1***

**( f ) Let OPT denote the optimal number of sheets that could have been used. Derive a (maybe trivial) lower bound on OPT in terms of V.**

Refer to point (C)

***V <= OPT***

**( g ) Put all things above together and prove that OPT >= k/2. You will need to use the fact that OPT and k are both integers. Try to first derive a strict lower bound on OPT in terms of k.**

From point (a):

We can conclude that APPROX ( k ) will take more than equal pages than OPT (optimal way of solving ).

OPT <= k —(1)

———

From point (f):

We can conclude that

V <= OPT —(2)

———

From part (e) [ k<=2V+1 ]

(k-1)/2 <= V

(k-1)/2 <= V <=OPT

(k-1)/2 <= OPT

k/2 - 1/2 <= OPT

**k/2 <= OPT**  [OPT and k are integers]

From points (d) (e) —(3)

V <= k < = 2\*V

———

From above points (1), (2), (3), we can conclude that:

***OPT<=k<= 2\*OPT***

**(h ) What is the approximation ratio (relative or absolute) of your `algorithm'?**

The relative approximation ratio is **2**.