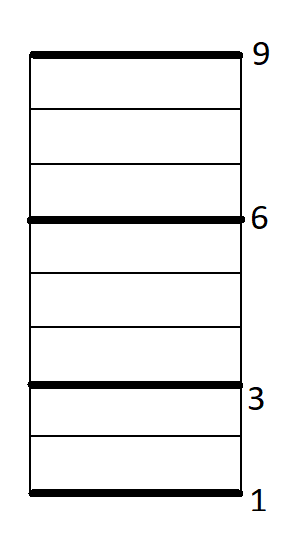
­­**Answer for ­­Question 3:**

1. As we have given k=2 jars. We have to find highest safe rung for some function f(n) which has time complexity less than linear.

The functions that are asymptotically better than O(n) can be O(n/2), O(

1. O(n/2) 🡪 O(c.n) where c=1/2 so O(n/2) is not efficient.
2. O(log n) 🡪 This is worst than binary search so we are dropping this possibility
3. O( We will consider this function as this is efficient that linear.



Suppose the ladder of 9 rugs so n=9

So

We drop jars from , 2

1. We will drop the jar at There are two case it may break at 3rd rung or it may not break at 3rd rung.

Case 1: If jar breaks at 3rd rung, now we can say that the highest safe rung height is below rung 3. We have just one jar left to decide highest safe rung. Now we will drop jar at 1st rug, if it does not break at 1st rug the highest safe rung is 2nd rung. If it breaks at 1st rung the there is no safe rung for the ladder.

Case 2: If jar do not break at 3rd rung, we will drop the same jar from next rung i.e. 2 rung

= 6th rung

1. Now we will drop the jar from 6th rung. Similarly, there are two cases for this rung. If the jar does not break at 6th rung we will proceed to drop the same jar from next rung i.e 3 rung

=9th rung.

If the jar breaks at 6th rung, we will drop the second jar from 4th rung if it does not break at this rung the highest safe rung is 5th rung and if it does not break then highest safe rung is below 4.

1. If the jar do not break at 9th rung then the highest safe rung is above the ladder. If it breaks at 9th rung we will drop the next jar from 7th rung and calculate the highest safe rung as stated above.

If n is not a perfect square, then we can use the same strategy dividing the latter in multiples of [

still obtaining an algorithm that runs in O(.

For n=8 , 2.83… so we will consider the steps of 2 for calculation.

So to generalize this case, Drop the first jar at the rung, 2rung, and so on until it breaks at the m rung. This takes at most steps . Then drop the second jar at every rung between the (m−1) rung and the m rung until you find the precise breakpoint. Therefore we can find highest safe rung of ladder using only two jars for f(n)= which has time complexity less than linear.

Time complexity: In binary search we get one block to search after breaking the first jar but in case of we get blocks. So the best balance with 2 jars is an even number between the number and size of each block of rungs. Therefore n/x= x which is n=x2. It leads to x=. So our algorithm do some x until we break first jar, plus x until it breaks second jar at highest rug. So it gives us 2

Time complexity 🡪 O(

1. With 2 jars i.e. k=2 we have equation as n/x=x ….. (1)

From this equation we get x=. If we substitute this value in equation (1) we get n/=.

For more jars i.e. k>2 we can change this equation to n/x= n1/k.

Therefore x=n/n1/k 🡪 x=n(k-1)/k

We will start dropping jars from height that is multiple of [ n(k-1)/k] until it reaches the highest rung. We will drop the first jar at most 2n1/k times.

If jar breaks at height m.n(k-1)/k, then we know that highest safe rung is between height

(m-1).n(k-1)/k and m. n(k-1)/k.

Now we are left with k-1 jars, We will repeat the procedure with k-1 jars in the interval (m-1).n(k-1)/k to m. n(k-1)/k.

Thus it will take less than 2(k-1)(n(k-1)/k)1/(k-1) = 2(k-1)n1/k drops.

Therefore, the entire algorithm will require less than 2kn1/k drops that we need to show.