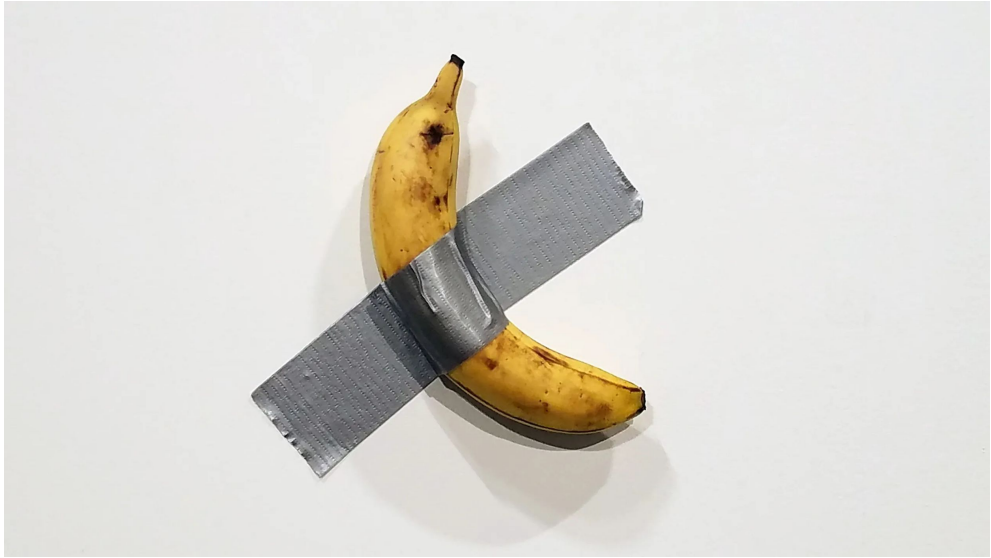
Algorithm Design - Term Project
CSX 3009 (541)

By : Ekkasith Singmaneechai 6213934
Saranya Sangsuk-iem 6237407

OUTLINE

- 1. Problem Overview
- 2. Problems Analysis
- 3. Solutions
- 4. Conclusion
- 5. References



Problems : *875. Koko Eating Bananas*

Difficulty : Medium

Problem Overview:

Koko loves to eat bananas. There are **n** piles of bananas, the **i^{th}** pile has **piles[i]** bananas. The guards have gone and will come back in **h** hours.

Koko can decide her bananas-per-hour eating speed of **k**. Each hour, she chooses some pile of bananas and eats **k** bananas from that pile. If the pile has less than **k** bananas, she eats all of them instead and will not eat any more bananas during this hour.

Koko likes to eat slowly but still wants to finish eating all the bananas before the guards return.

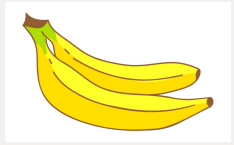
*Return the minimum integer **k** such that she can eat all the bananas within **h** hours.*

Koko wants to *Annihilate all bananas before guard come*

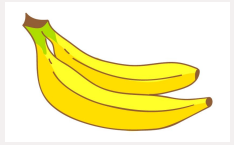
n Piles	piles[0]	…	…	piles[i]

Time Left : h

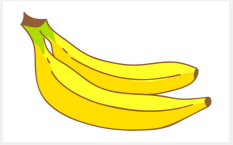
N Piles



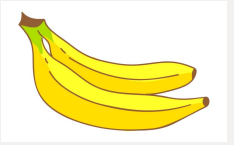
piles[i]



piles[i]



piles[i]



piles[i]

Objective : find the right k value
Banana per Hour: k

Koko the Monke

Condition : Must be the longest time possible.

Example & Constraints

Example 1:

Input : piles = [3,6,7,11], h = 8

Output: 4

Example 2:

Input : piles = [30,11,23,4,20], h = 5

Output: 30

Example 3:

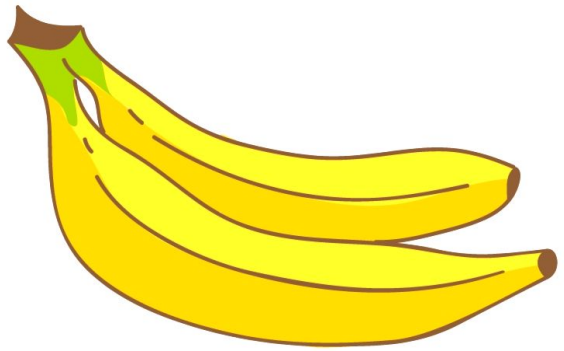
Input : piles = [30,11,23,4,20], h = 6

Output: 23

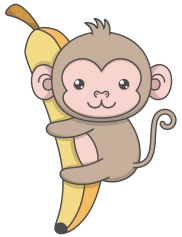
Example & Constraints

Constraints:

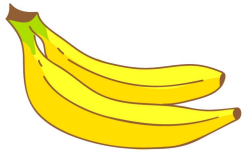
- $1 <= piles.length <= 10^4$
- $piles.length <= h <= 10^9$
- $1 <= piles[i] <= 10^9$



Problem Analysis



= n piles of bananas



= x bananas per hour

= y time taken

Ex. If she eats *3* bananas per hour, it takes her *2* hours to eat a pile of *4* bananas.

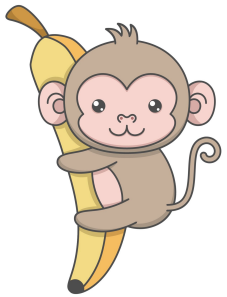
Problem Analysis

h $>$ 

The first constraint of the problem is that Koko has to eat all the piles within h hours, where h is no less than the number of piles.

Workable Speed	Unworkable Speed
Finish within h hours	Cannot finish within h hours

Solution 1
– Brute Force –



What is Brute Force?

- Solve the problem by instructing to do a 'loop' until you get an answer.
  - Bubble sort, Selection sort

How it works?

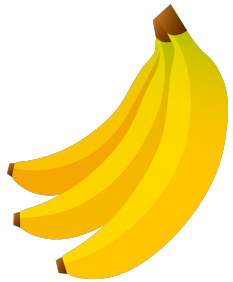
1. Pick up objects one by one
2. Check whether each object meets the conditions
   a. pick up the next object. If the object runs out, then stops.
   b. check if the object is the one you want to search for.
   c. if yes, return the object.
   d. if not, go back to step a.

How Brute Force solve the problem?

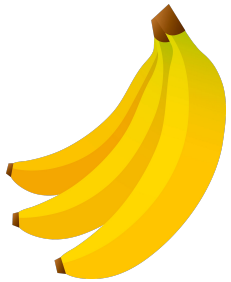
3 6 7 11

currTime = [NumberOfBananas / speed]
totalTime = prevTotalTime + currTime

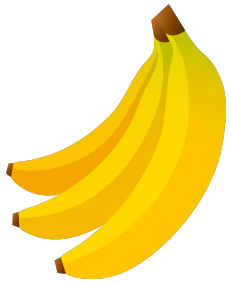
How Brute Force solve the problem? Workable Speed



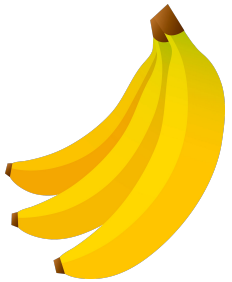
3



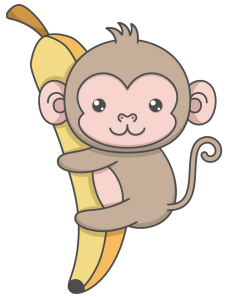
6



7



11



Koko:
I eat 5 bananas per hour.

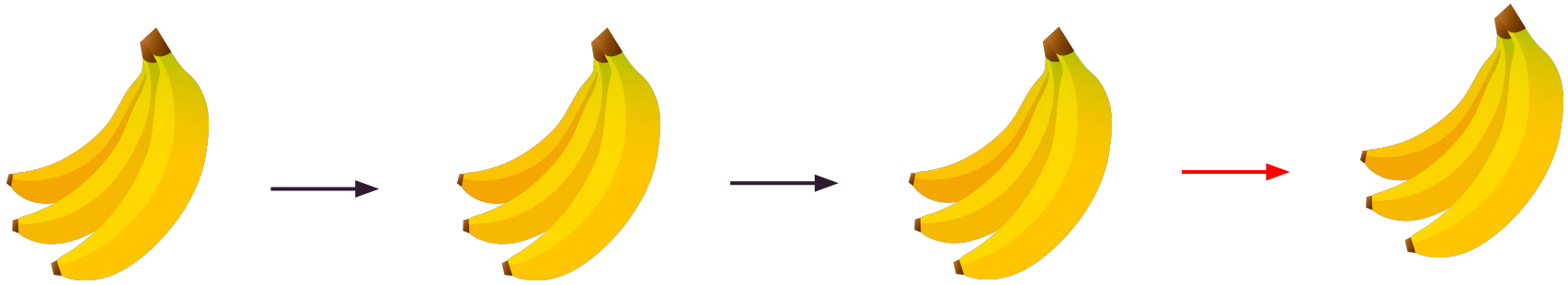
Guards:
I will return in h = 8 hours.

How Brute Force solve the problem? Workable Speed

Koko:
I eat 5 bananas per hour.

Guards:
I will return in h = 8 hours.

currTime = [NumberOfBananas / speed]
totalTime = prevTotalTime + currTime



3

currTime: 3 / 5 = 1
totalTime: 0 + 1 = 1

6

currTime: 6 / 5 = 2
totalTime: 1 + 2 = 3

7

currTime: 7 / 5 = 2
totalTime: 3 + 2 = 5

11

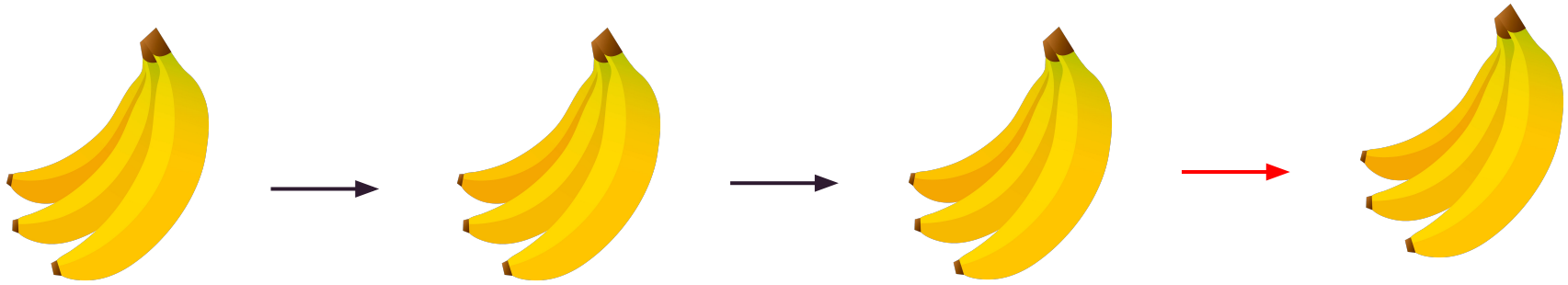
currTime: 11 / 5 = 3
totalTime: 5 + 3 = 8

How Brute Force solve the problem? Unworkable Speed

Koko:
I eat 3 bananas per hour.

Guards:
I will return in h = 8 hours.

currTime = [NumberOfBananas / speed]
totalTime = prevTotalTime + currTime



3

currTime: 3 / 3 = 1
totalTime: 0 + 1 = 1

6

currTime: 6 / 3 = 2
totalTime: 1 + 2 = 3

7

currTime: 7 / 3 = 3
totalTime: 3 + 3 = 6

11

currTime: 11 / 3 = 4
totalTime: 6 + 4 = 10

Brute Force Solution

Does the order by which Koko eats affect the overall time?

NO

Brute Force Solution - Algorithm

1. Start at $speed = 1$.

2. Given the current speed, calculate how many hours Koko needs to eat all of the piles.
   a. If Koko cannot finish all piles within h hours, $speed = speed + 1$
   b. If koko can finish all piles within h hours, go to step 3.

3. Return the $speed$ as the answer.

Implementation

```python
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) -> int:
        speed = 1

        while True:
            hour_spent = 0

            for pile in piles:
                hour_spent += math.ceil(pile / speed)

            if hour_spent <= h:
                return speed
            else:
                speed += 1

```

Submission

Time Submitted	Status	Runtime	Memory	Language
02/23/2022 18:00	Time Limit Exceeded	N/A	N/A	python3

Complexity Analysis - Time complexity

Let n be the length of input array piles
and m be the upper bound of elements in piles.

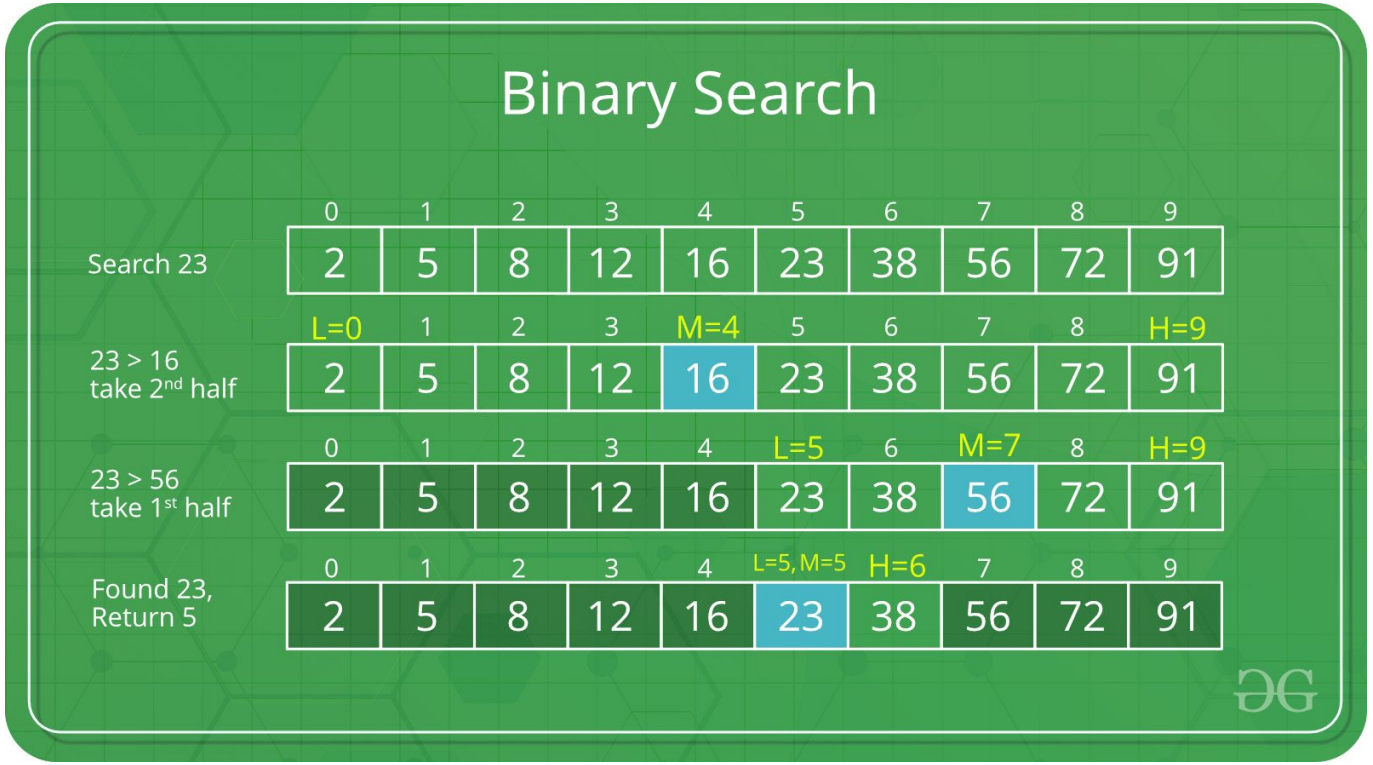
- Time complexity: O(nm)
  - It takes O(n) times.
  - Try every smaller eating speed from 1 to m

Complexity Analysis - Space complexity

- Space complexity: (O)1
  - Constant space is required to do calculations.

Solution 2
– Binary Search –

How Binary Search Solve the Problem ?



Link: **https://www.geeksforgeeks.org/binary-search/**

What we know?

Things we know from Brute Force:

1. The order doesn't matter.
2. We know that if Koko can eat the banana at speed *n* then she can finished it with the speed of *n + 1*, *n+2*, and so on.
3. From 2. it means that if Koko can't finished it in n so can't she in *n-1*.

How Binary Search Solve the Problem ?

Speed(k)	Time Taken to eats all piles of banana
1	…
…	N-3
2	N-2
3	N-1
4	N
5	N+1
…	N+2
Max(piles)	…

for N is the first workable speed, which is the target

Binary Search

Once we set the boundary we can apply the Binary Search with *1* is the left boundary and *max banana piles* as the upper or right boundary

1 left	…	Start Value middle (left+right)/2	…	Max(pile) right

Example:

Let's say that the given array are *[1,3,5,7,9]*, and *h = 8* Based on the shown diagram earlier we will get the value as follows

(always initiate as 1) 1 left	…	Start Value (left+right)/2 5 middle	…	Max(pile) 9 right

Example:

We then check if value *5* is a doable speed or not
by used the value to check if Koko can finished eating in time
within *h = 8* hours before guard arrives.

Pile(0) = 1/5 1 hour = 1+0 = 1
Pile(1) = 3/5 1 hour = 1+1 = 2
Pile(2) = 5/5 1 hour = 1+2 = 3
Pile(3) = 7/5 2 hour = 2+3 = 5
Pile(4) = 9/5 2 hour = 5+2 = 7 hours total

Example: If the middle is doable

You can see that with speed *5* Koko took *7* hour to eat all the banana before guards arrives which within the time frame. It means that from *middle* to *right* (*5* to *9*) is the *doable speed* but *not desirable* because Koko want to eats as slow as possible so we can set the *middle* as *right* instead.

(always initiate as 1) 1 left	…	(left+right)/2 3 middle	…	Max(pile) 5 right

Example: If the middle is NOT doable

However if it's not then we set the middle value to left instead of right.

5 left	…	(left+right)/2 7 middle	…	Max(pile) 9 right

Example: When do we know if the result is reach?

The result is reach when the *left* = *right* value. we can see that in this case the first *middle* value, *5* is the target value.

5 left	…	middle	…	Max(pile) 5 right

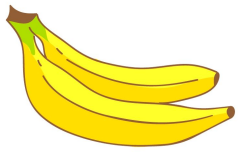
Binary Search Complexity Analysis

Let ***n*** = input array piles and ***m*** = maximum number of bananas in a single pile from piles.

Time complexity: ***O(n ·logm)***

In Binary Search, it takes ***log m*** to check from the start value to finished

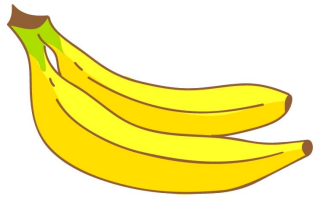
However, it require 1 for loop to calculate the piles of banana if it's satisfy the condition or not, which makes it total ***n ·logm***



Binary Search Complexity Analysis

Space complexity: ***O(1)***

For each eating speed *middle*, we iterate over the array and calculate the total hours Koko spends, which costs constant space.
Therefore, the overall space complexity is ***O(1)***.



Algorithm

1. Initialize the two boundaries of the binary search as *left = 1*, *right = max(piles)*.

2. Get the *middle* value from *left* and *right*, that is, *middle = (left + right) / 2*, this is Koko's eating speed during this iteration.

3. Iterate over the piles and check if Koko can eat all the piles within *h* hours given this eating speed of *middle*.

4. If Koko can finish all the piles within *h* hours, set *right* equal to *middle* signifying that all speeds greater than middle are workable but less desirable by Koko. Otherwise, set *left* equal to *middle + 1* signifying that all speeds less than or equal to middle are not workable.

5. Repeat the *steps 2, 3, and 4* until the two boundaries overlap, i.e., left = right, which means that we have found the minimum speed by which Koko could finish eating all the piles within h hours. We can return either left or right as the answer.

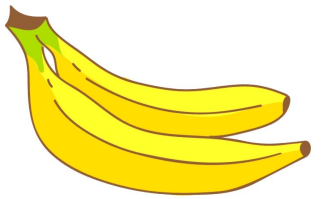
Implementation

```python
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) -> int:
        left  = 1
        right = max(piles)

        while left < right:
            middle = (left+right)//2
            hour_spent =  0

            for pile in piles:
                hour_spent += math.ceil(pile/middle)

            if hour_spent <= h:
                right = middle
            else:
                left = middle + 1
        return right
```



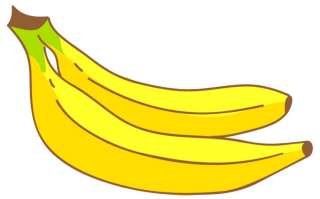
Submission

Success Details ›

Runtime: 886 ms, faster than 17.15% of Python3 online submissions for Koko Eating Bananas.

Memory Usage: 15.6 MB, less than 58.66% of Python3 online submissions for Koko Eating Bananas.

Time Submitted	Status	Runtime	Memory	Language
02/23/2022 11:03	Accepted	886 ms	15.6 MB	python3



Program Example

Test Case:

```
1,3,5,7,9
8
```

Result:

```
>python Koko.py < 1.in
5
Time: 0.0
```

Program Example

Test Case: 10000 piles

904456204,709336345,785787897,295543213,620888827,25221792,373224639,331963149,453419469,539351466,397782166,495271721,454809495,64867774,355692149,373472292,254501847,326006144,70211422,384948762,950119490,197958886,150946179,495888099
3,116369334,86274426,798061777,453149818,925753803,399193451,580348475,389887138,843320079,786347620,230904355,745412424,928018862,460931258,838222383,682438208,179530400,159611923,278086441,980885375,342014173,773521033,228450130,47145
66293735,763847350,797779158,257479738,612956074,401900347,718678796,536151888,400682624,393916061,193153020,698444941,584309259,83760676,577266259,667062980,936177130,538071967,88361682,48959076,174208982,512639485,687625608,866621577,
71348929,418990381,946573175,791315277,752522211,538948746,371755142,671716245,208980394,567388799,89516963,299041657,825614855,516372421,832271822,657998205,985634125,586740988,214746690,199782596,379956398,744977958,669259070,74778650
63,933998989,508259432,382879577,868346451,391294553,981187532,871319111,950570725,742229867,226729091,311341852,490704420,547020181,697564884,70319187,267183876,940369778,754654792,869226834,94743096,36465886,114558584,668987762,958655
,942054230,939189639,121665343,373298740,12898091,293298623,932956309,784669512,78611693,436210799,591229915,600497873,59006315,478253257,5490692,605016379,912848287,264117661,770025139,158063097,734672824,909613594,243272304,768022633,
858926734,673011692,698081942,939454196,819723961,106557168,798289048,524846900,906439891,59087496,467401254,664620822,627897003,710576568,920302687,331308976,605934400,325254352,991309550,432209544,925343115,138623664,794542592,9017614
26498043,522503938,836259679,686991561,170615250,419895370,120725014,529184755,360487800,112935483,316831799,634367791,584528566,206998041,818855965,182248053,992695267,505393951,400016392,766503232,143752167,410921739,973404274,5541221
8830722,775772480,110422349,331657695,708508822,328822529,75064555,10293832,110946364,353853,99273642,706144128,708999084,302429968,33361713,394494744,34228016,52746672,240389168,45392702,496006507,540713674,669275863,317813369,75893849
473299,267426037,314052749,298421759,13122901,123820374,234255048,695701577,643439267,870508979,734489894,255795456,374769107,103220628,505650907,507312251,22249687,364819552,127121191,626512080,258105223,414801879,321613464,338352181,4
382,920131740,159191500,977392606,225359669,749786753,245783476,280698715,307881486,486718997,375906647,469859177,547125061,998855921,408120481,51238031,435705091,460552988,235940305,509026603,644231117,11309020,65043236,536846968,6323
702,731252508,740474079,23542550,200750128,167612416,282699851,583519200,491676074,75683561,436338999,673979439,55610892,106922995,889430805,964776006,697135381,825700948,284976465,65126088,911913657,356332592,457137852,483087304,4315360
,655795487,848642369,800440176,606596770,204337377,319681431,969469316,918545914,898927498,253376114,576178107,27306575,737689724,949335996,556440525,346840765,811023783,909164741,64099554,418450145,535447385,359306818,614127464,5549114
667,801976834,539164748,875198536,655628919,672118947,191602192,968012859,145503115,5105306,150709586,145304903,157214057,169196188,973749988,309243223,124866894,375080874,656514430,235702670,660535893,181529427,773108035,814677768,1167
186,742027991,432396952,375032828,117388375,759669899,975109492,453400973,355237445,989724829,477544977,655908793,777708661,35938925,480508190,881808780,389730770,581498514,493306267,406242933,67522022,338529621,117969637,142341699,9727
869637708,338715743,555395117,525016214,689363374,958490465,797495659,629607002,332048487,465405004,957804666,960615337,845643303,171068857,194666074,754346581,772585491,933146054,274330902,779461402,104489111,779983891,571313203,808244
92398157,192344944,675735963,270386876,612216427,660323816,580686577,2019386,762128628,967500072,566859703,705280183,935588590,828362153,214228693,206161657,184533271,906536874,239318283,682919418,140940080,189153500,876948757,993549410
07525064,199349763,492582542,584251050,165857676,674353880,404181269,136490430,915992774,246507268,615718691,71982776,564504646,951914406,106239995,620533917,685566568,260787713,595162372,413368303,451003419,332197663,864972590,380615764
812,32288529,299074900,73790590,218043144,479819557,877833178,295305641,560587098,8285861,817415048,47000482,952984280,388422791,313770142,566931134,291415965,681064209,966939217,869751234,208843417,881341462,888957072,13709059,53181080
02944035,999002090,21419222,7584671,111072810,358273052,341918417,330996387,522613059,967640122,526025198,445436383,610458160,863197714,754692794,875863540,674572396,792660439,915881086,666557808,886991831,477230069,54744571,185559785,3
58460412,896887420,288666826,70504663,729358535,306485591,697707998,393712047,602351527,816580700,188547800,719474675,862268832,732823142,976450445,350994571,149006719,610609729,563490111,251556252,55101914,529386462,651724808,667646774
8,487078106,228636227,886931948,871215118,251361678,946246794,193018407,106106093,399776252,801723370,119855680,557310942,680037723,873317295,938014420,146743008,287582123,164849512,14220905,392256537,343195320,559693370,370943054,99799
4,427691960,103967436,962931630,683290624,719500637,5444408,813319179,276665697,718180000,95817260,277662988,990665919,323719317,287006633,846316453,133077773,664538976,95544039,158047740,951978853,540201835,345128907,680663591,26948478
9861169,542607248,106478705,581163922,173652717,546442430,533930907,518320685,32226446,842049989,302573884,947669741,97277406,778492530,99292300,296897144,226240301,77640145,294860174,509671510,719561272,220171927,773262096,966105031,63
,830294571,873818185,705055278,651804575,240556407,53144036,658933200,712111288,662102825,930377438,156645509,660974437,574030129,922906226,94752481,673285972,272036291,776556566,970654136,551419545,927350609,849710578,549025606,1487291
,878116002,781808557,21509303,738074532,912505843,521181170,911967451,237438619,729006636,869455523,95717573,406746402,214820493,703272069,434913729,515195338,194752012,271404900,345114068,41079805,672499561,835297308,113238925,45056558
458486,48365562,451061,708954217,615295148,489614330,298627341,486420774,621470215,135450054,904674383,501904877,525353808,406765305,255303573,888320798,499085387,778722,813829766,459962249,784118836,530109514,608274702,933631495,210490
1,996083574,692815329,928865754,104195907,225824994,190731584,835059309,705163001,804142024,258565286,596717256,853110146,318249722,895641749,373446651,67601097,803168574,417540009,389147278,259768282,110855311,724042739,11861278,314723
4,96367235,596918068,223864543,944308513,252912878,34150728,659082914,209130123,281548235,614940324,581377735,651072758,540689135,819470028,516284254,100103752,362297202,629502856,338645680,904230339,734481791,188138620,162085183,358068
34560830,832135336,427471439,401412773,240627026,957668814,959504821,54849858,518431183,529335785,169462739,534067939,693339797,214977402,564457542,647836168,42527866,184269807,187608700,118901141,466057874,859082938,235283094,698042119
48013796,275046156,285374325,66079440,617418296,742715710,742468992,849731845,954915752,377224543,840453260,893610481,311358627,940931669,647765228,29411114,497024272,872865422,874306105,144027023,994031198,296963410,694407477,906162364
186752,445748738,939955979,213815821,655137137,932543917,200490078,20329112,217674132,60320114,796716888,135642378,235154459,379087126,154677185,951418505,959477082,871211530,314206175,456670722,265043968,233606951,975584257,907689452,4
76063,258557161,13552152,784494195,628171531,765218945,67529934,34711355,408371368,869890528,176203962,886257162,538278905,767013751,41256624,965725216,766987568,822699664,480880236,825505915,826188186,109993795,14236610,958941653,29178
274,51553291,346474640,245432496,702576703,469206489,411976517,988526619,413130601,455620452,533532355,169691569,435667145,418205787,536993281,441580951,412091099,379671274,965407415,768460591,931898393,209582119,326801298,882870949,333
40901,733144132,33654118,361811005,865290870,979726241,251825249,751153622,919728719,891119305,409650698,213194866,721466084,321999156,612852502,234974252,853481816,849854749,290828580,856971036,219051279,572873392,20898051,630353137,50
671705673,377706888,844505449,470015568,323828664,217892702,495265461,439299241,950645625,352576207,54238540,976568491,6180004,351854552,324617236,301643831,353227523,862159010,749011796,943324904,330865956,507943283,306879782,43100298,
3362,597515298,95078898,403307681,904414930,182693638,620910671,428293117,659449543,704999609,471862373,673670167,703800662,602590436,543269536,992118568,674443616,52861246,167393893,793535941,88072581,143685044,230420359,693647228,7912
967194354,135623054,513017398,939356479,896934955,915845670,690693290,666561323,230805461,957719442,187199680,315242262,335324024,342545178,559363739,11203624,299953418,901059833,451756826,57871199,483629697,730012013,258030056,95083849
88934914,681317256,366495238,648074646,513033283,774958274,986578762,628793346,284369783,463776621,874840756,61176596,264402980,790353692,957088754,475826448,398951715,907589922,212241594,668237011,160741020,657714680,369841682,76905831
17681,325314952,550325648,812861543,752846329,877669640,452695731,799287790,282394551,248359758,740186272,985230641,846276117,644996392,462227720,449948499,560855115,561737013,177024831,67237182,48577115,471596852,403852769,724131705,69
19067597,713424296,760854368,104957130,598895498,235882044,198901208,38863632,849256961,799685994,881327905,342907305,229050557,775981208,456122170,987486245,608905395,773882586,966521212,488396965,208061308,776740197,207414855,58605708
2315789,836979813,313041293,773391282,905330058,87220743,896522515,748901081,7964236,845837239,558199645,59786784,378386770,175075889,526844598,256633145,664384564,192884124,473942905,482459883,593587535,473388203,867433899,881355826,16
2984550,546957943,329976986,414773554,587976437,175185734,233231146,556295646,239065407,499000004,240038899,81656434,672810130,815467359,337582112,924210981,463780444,71943652,669437856,908346680,341949313,4411963,43616270,673127170,576
934714,779436691,985637949,782299170,586239292,336340236,742093213,317421545,143110187,278617044,733119066,584021054,698971464,347120272,595966935,154187878,136084739,813520395,927717358,523643195,403543926,232379023,859779522,86513343,
6551,827019499,222059208,494721836,777025129,490163166,241582997,685248333,583860630,536300357,749109810,587163410,410274608,216772795,926277337,869766164,725863351,76095746,160021589,725250928,137318906,87870359,942482098,341976819,587
5,633188403,555261241,649187442,1190868,800125748,741013899,445911525,50306494,880600941,314350429,796555482,177924133,600057759,97606598,157411998,394540604,875049381,777469301,804034528,854292387,872182058,618480144,136819314,36027133
362621683,867278865,47543871,753899832,348414169,345849537,572844727,784281456,447600482,560691278,198153254,531947875,662125234,408904536,695493047,944589764,302579330,184651674,632099134,385063769,48957382,728692804,37949214,779427564
821866,905193641,486291388,304357680,552325847,792004385,281204975,538221890,567610926,559728575,354313025,89241993,908419180,266387682,923623977,964605061,173096540,638921331,440062795,43212903,808871120,904457492,876982854,518392148,2
499836490,990175889,539812215,513179912,20942332,795830234,94473608,673972973,641830147,973763489,937986067,893092719,248405029,134447694,536941184,471633720,707311161,639342724,908967239,383707974,848804024,679728374,924281495,80307461

Program Example

Result:

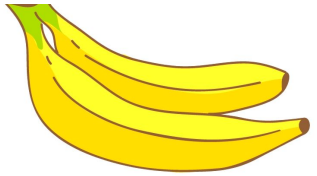
```
python Koko.py < 2.in
```

```
5947
Time:  0.046875
```

Comparison

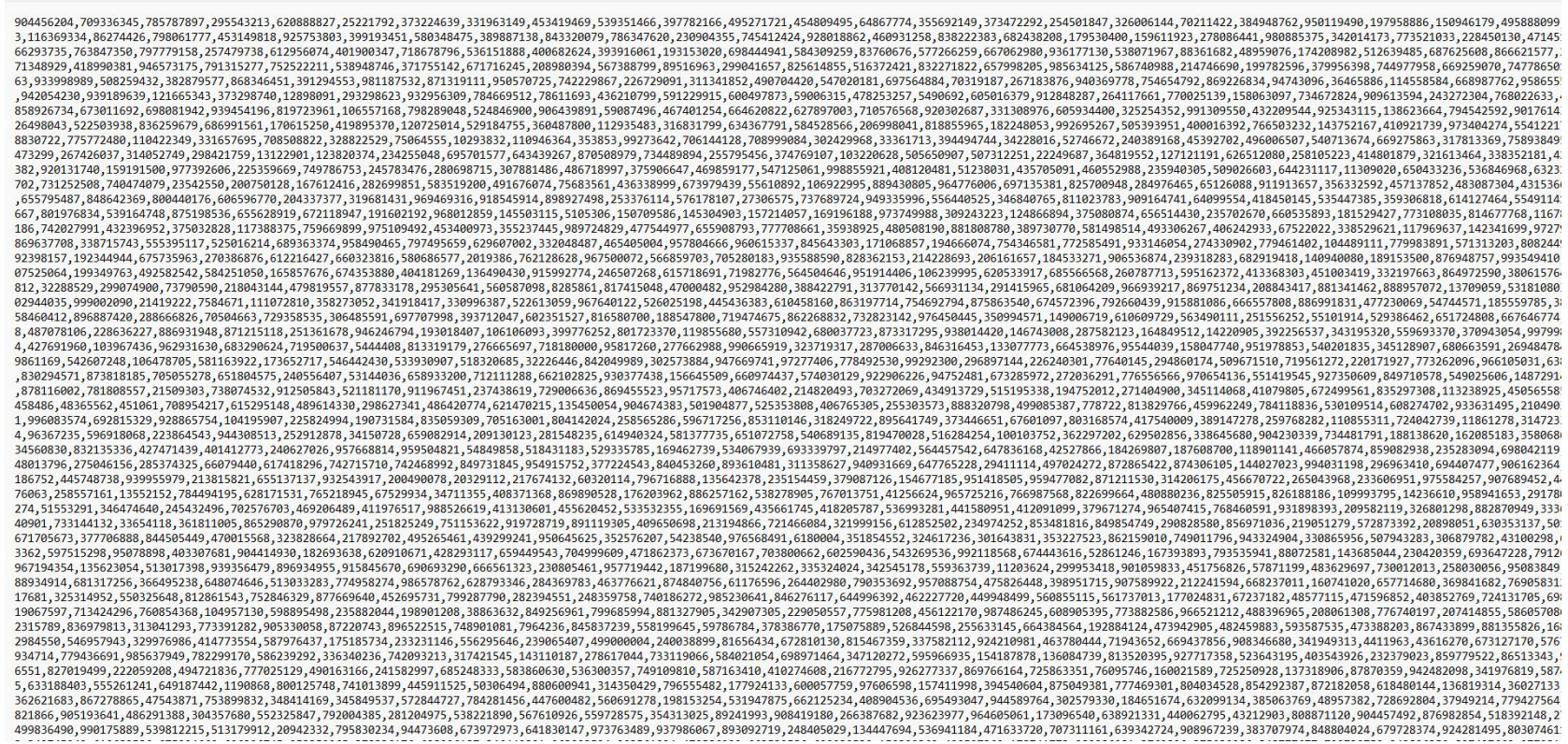
Time Submitted	Status	Runtime	Memory	Language
02/23/2022 11:03	Accepted	886 ms	15.6 MB	python3

Time Submitted	Status	Runtime	Memory	Language
02/23/2022 18:00	Time Limit Exceeded	N/A	N/A	python3



Comparison

10000 piles



Comparison

Brute Force Result:

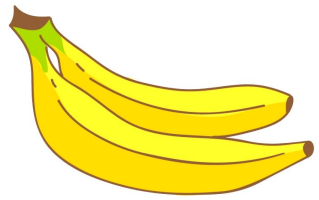
```
python Monke.py < 2.in
```

```
5947
Time: 6.734375
```

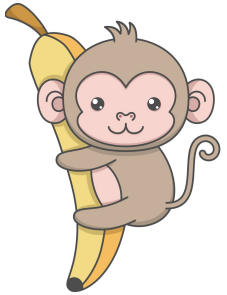
Binary Search Result:

```
python Koko.py < 2.in
```

```
5947
Time: 0.046875
```



Thank You!



References

- https://leetcode.com/problems/koko-eating-bananas/solution/
- https://www.geeksforgeeks.org/binary-search/