Binary Search

Jan 1st 2021
Yifan Fei

278. First bad version
1011. Capacity To Ship Packages Within D Days
410. Split Array Largest Sum
1482. Min Number of Days to Make m Bouquets
875. Koko eating bananas
1283. Find the Smallest Divisor Given a Threshold
668. Kth Smallest Number in Multiplication Table
1201. Ugly Number III
719. Find K-th Smallest Pair Distance
1631. Path With Minimum Effort

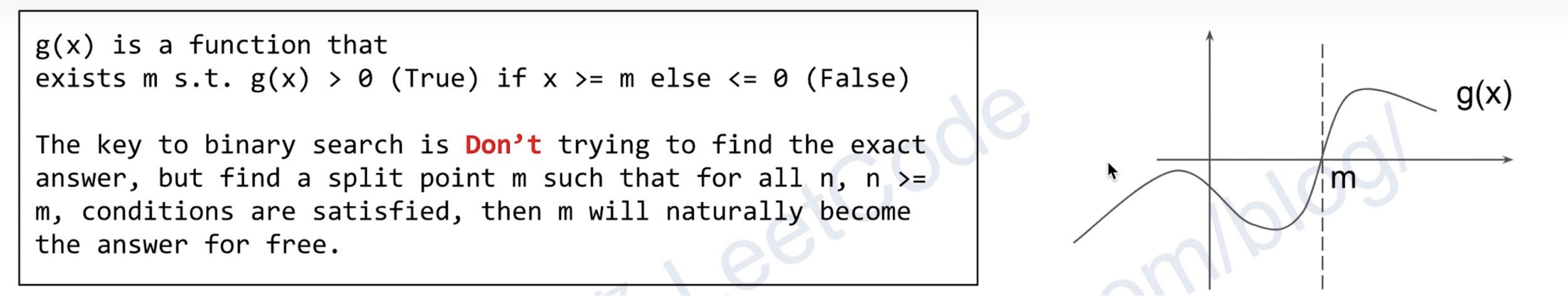
Reference:
LC总结帖：https://leetcode.com/discuss/general-discussion/786126/python-powerful-ultimate-binary-search-template-solved-many-problems

b站古城悦少：https://www.bilibili.com/video/BV1Ea4y1L7HC
huahua：https://www.youtube.com/watch?v=J-IQxfYRTto&list=PLLuMmzMTgVK74vqU7Ukaf70a382dzF3Uo&index=6

Intuition

Given **a boolean check function check(m)**, return the smallest index m such that check(m) is True.

1. [l, r + 1)
2. [l, r]
3. (l - 1, r + 1)

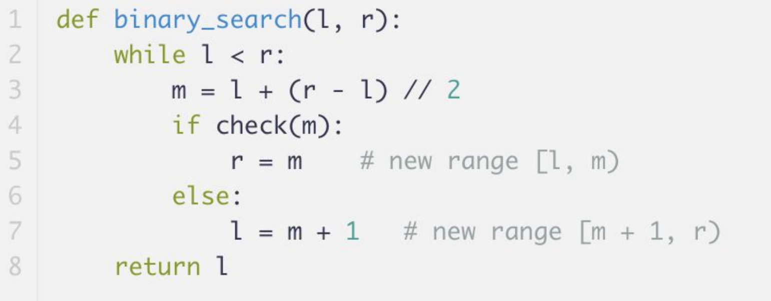


```
g(x) is a function that
exists m s.t. g(x) > 0 (True) if x >= m else <= 0 (False)

The key to binary search is Don’t trying to find the exact
answer, but find a split point m such that for all n, n >=
m, conditions are satisfied, then m will naturally become
the answer for free.
```

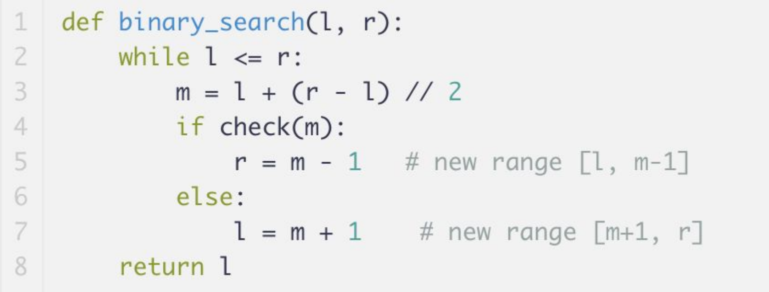
Template

Time complexity will be O(logn) * O(check). Space complexity normally depends on O(check)

1. [l, r + 1), in the end l = r
2. [l, r], in the end l & r swap => [r, l]
3. (l - 1, r + 1), in the end => (l, r)



```python
def binary_search(l, r):
    while l < r:
        m = l + (r - l) // 2
        if check(m):
            r = m     # new range [l, m)
        else:
            l = m + 1   # new range [m + 1, r)
    return l
```



```python
def binary_search(l, r):
    while l <= r:
        m = l + (r - l) // 2
        if check(m):
            r = m - 1   # new range [l, m-1]
        else:
            l = m + 1    # new range [m+1, r]
    return l
```

Take-away

1. 注意开闭区间，会不会出现死循环。
2. check(m)是一个穿越x-axis的函数，本质是找函数零点
3. Python不会有overflow问题，但C++/Java会index溢出
4. return in while loop not recommended

Python standard libary bisect

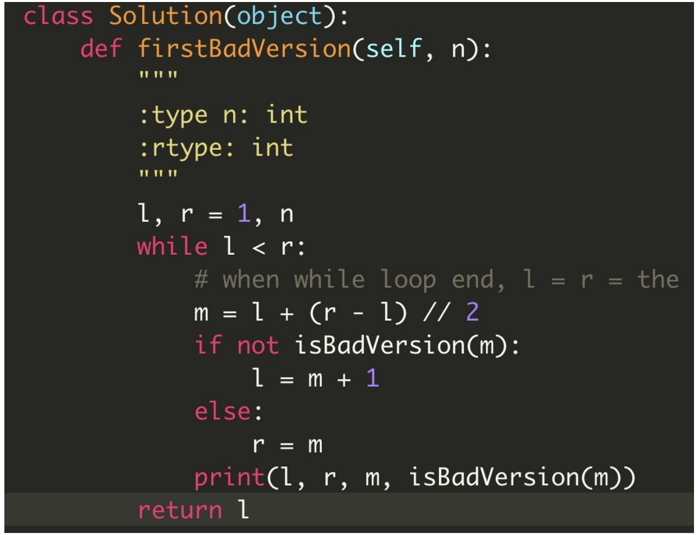
bisect_left vs bisect_right

https://github.com/python/cpython/blob/master/Lib/bisect.py

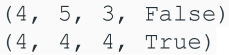
本质是check函数条件不一样，小于 vs 小于或等于

278. First bad version

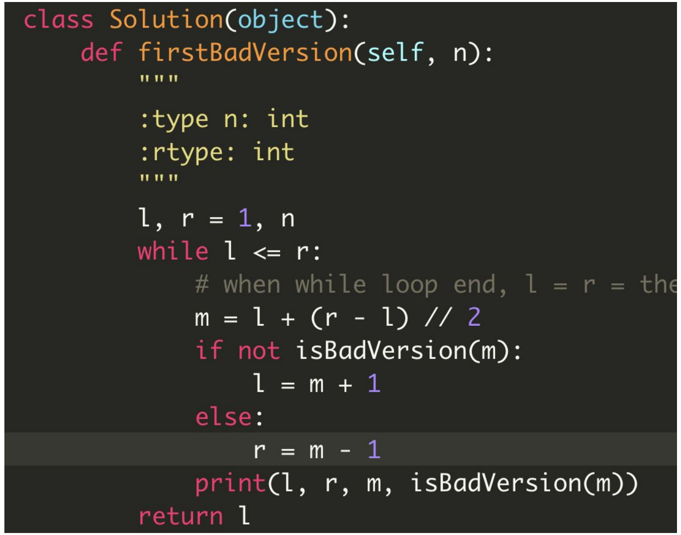
For easy problem normally we don’t need to write the check function ourselves. The check function would be just an embedded one or one line comparison.



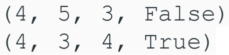
```python
class Solution(object):
    def firstBadVersion(self, n):
        """
        :type n: int
        :rtype: int
        """
        l, r = 1, n
        while l < r:
            # when while loop end, l = r = the
            m = l + (r - l) // 2
            if not isBadVersion(m):
                l = m + 1
            else:
                r = m
            print(l, r, m, isBadVersion(m))
        return l
```



```
(4, 5, 3, False)
(4, 4, 4, True)
```

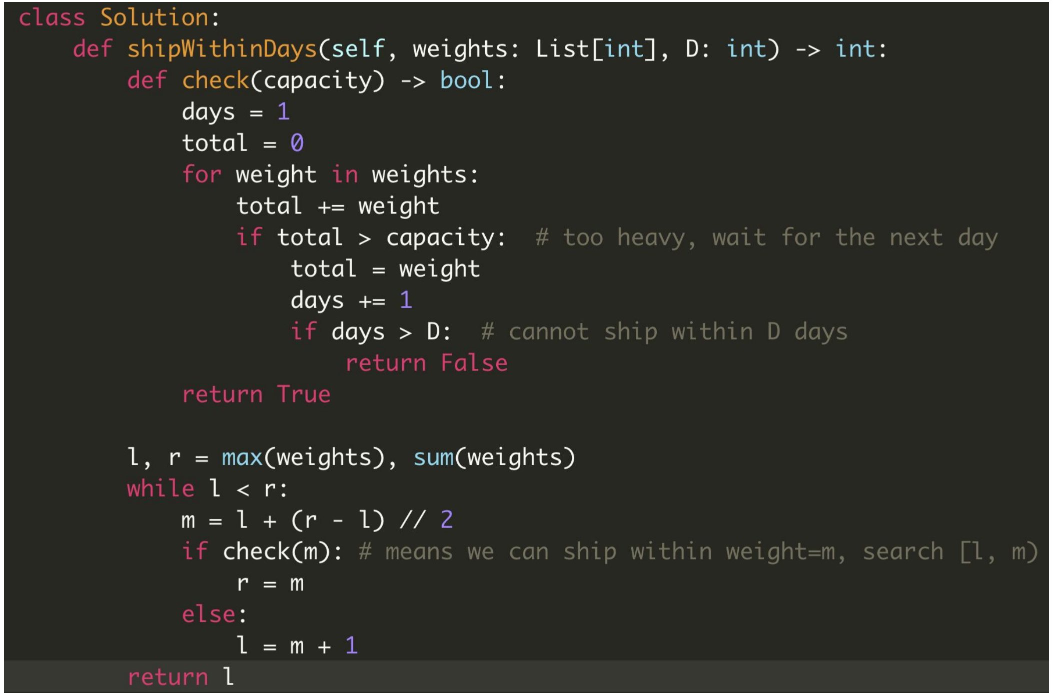


```python
class Solution(object):
    def firstBadVersion(self, n):
        """
        :type n: int
        :rtype: int
        """
        l, r = 1, n
        while l <= r:
            # when while loop end, l = r = the
            m = l + (r - l) // 2
            if not isBadVersion(m):
                l = m + 1
            else:
                r = m - 1
            print(l, r, m, isBadVersion(m))
        return l
```



```
(4, 5, 3, False)
(4, 3, 4, True)
```

1011. Capacity To Ship Packages Within D Days



```
class Solution:
    def shipWithinDays(self, weights: List[int], D: int) -> int:
        def check(capacity) -> bool:
            days = 1
            total = 0
            for weight in weights:
                total += weight
                if total > capacity:  # too heavy, wait for the next day
                    total = weight
                    days += 1
                    if days > D:  # cannot ship within D days
                        return False
            return True

        l, r = max(weights), sum(weights)
        while l < r:
            m = l + (r - l) // 2
            if check(m): # means we can ship within weight=m, search [l, m)
                r = m
            else:
                l = m + 1
        return l
```

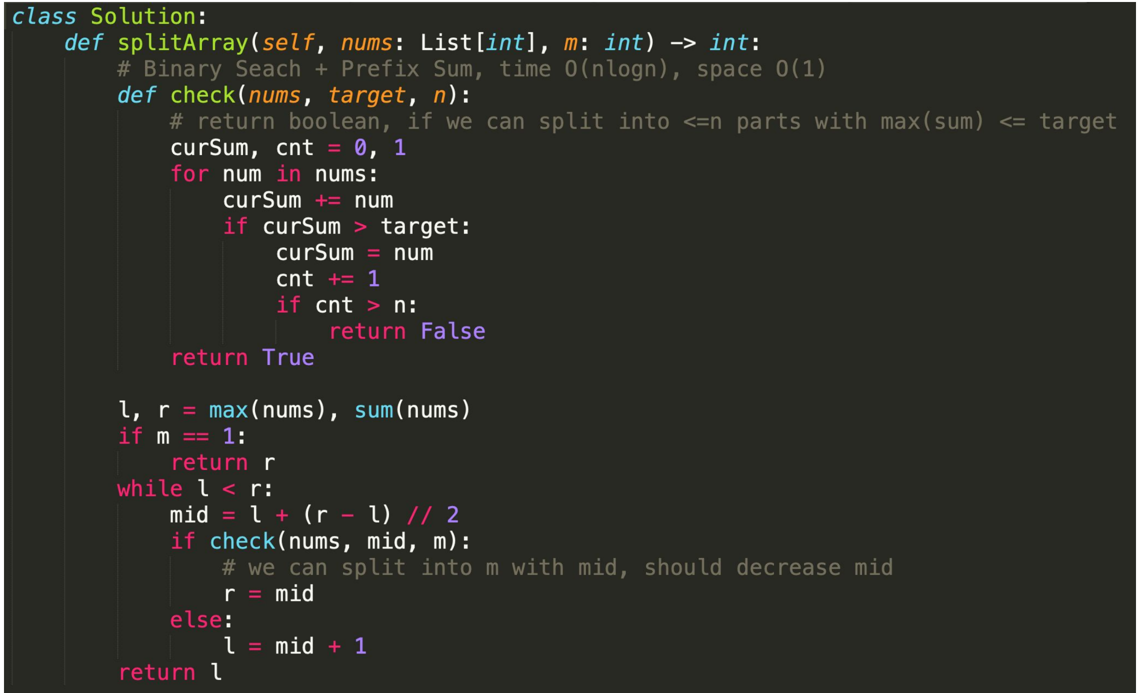
Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within `D` days.

`weights = [1,2,3,4,5,6,7,8,9,10], D = 5`

`Output: 15`

- `1 <= D <= weights.length <= 50000`
- `1 <= weights[i] <= 500`

410. Split Array Largest Sum



```python
class Solution:
    def splitArray(self, nums: List[int], m: int) -> int:
        # Binary Seach + Prefix Sum, time O(nlogn), space O(1)
        def check(nums, target, n):
            # return boolean, if we can split into <=n parts with max(sum) <= target
            curSum, cnt = 0, 1
            for num in nums:
                curSum += num
                if curSum > target:
                    curSum = num
                    cnt += 1
                    if cnt > n:
                        return False
            return True

        l, r = max(nums), sum(nums)
        if m == 1:
            return r
        while l < r:
            mid = l + (r - l) // 2
            if check(nums, mid, m):
                # we can split into m with mid, should decrease mid
                r = mid
            else:
                l = mid + 1
        return l
```

minimize the largest sum among these m subarrays.

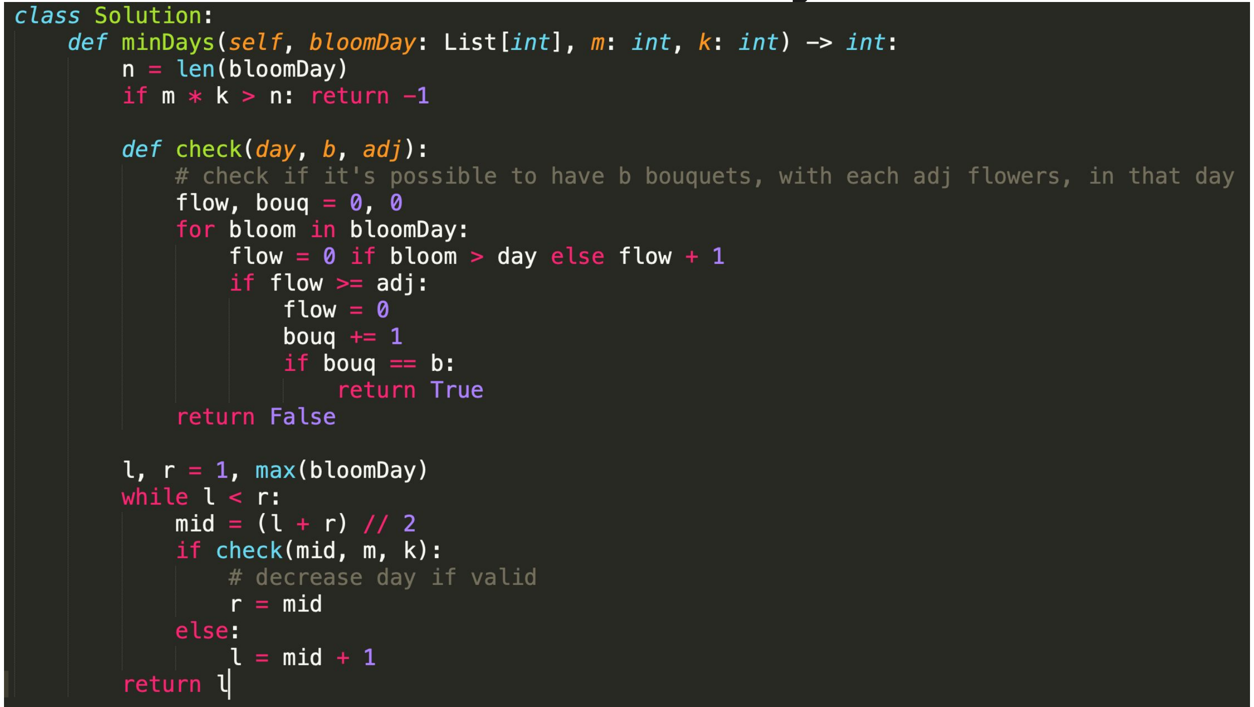
- `1 <= nums.length <= 1000`
- `0 <= nums[i] <= 106`
- `1 <= m <= min(50, nums.length)`

DP barely solves this problem but much much slower!!

Top down DP + Prefix Sum, Time O(mn^2), Space O(n)

Binary Seach + Prefix Sum, time O(nlogn), space O(1)

1482. Min Number of Days to Make m Bouquets



```python
class Solution:
    def minDays(self, bloomDay: List[int], m: int, k: int) -> int:
        n = len(bloomDay)
        if m * k > n: return -1

        def check(day, b, adj):
            # check if it's possible to have b bouquets, with each adj flowers, in that day
            flow, bouq = 0, 0
            for bloom in bloomDay:
                flow = 0 if bloom > day else flow + 1
                if flow >= adj:
                    flow = 0
                    bouq += 1
                    if bouq == b:
                        return True
            return False

        l, r = 1, max(bloomDay)
        while l < r:
            mid = (l + r) // 2
            if check(mid, m, k):
                # decrease day if valid
                r = mid
            else:
                l = mid + 1
        return l
```

We need to make m bouquets. To mak
a bouquet, you need to use k
adjacent flowers from the garden.

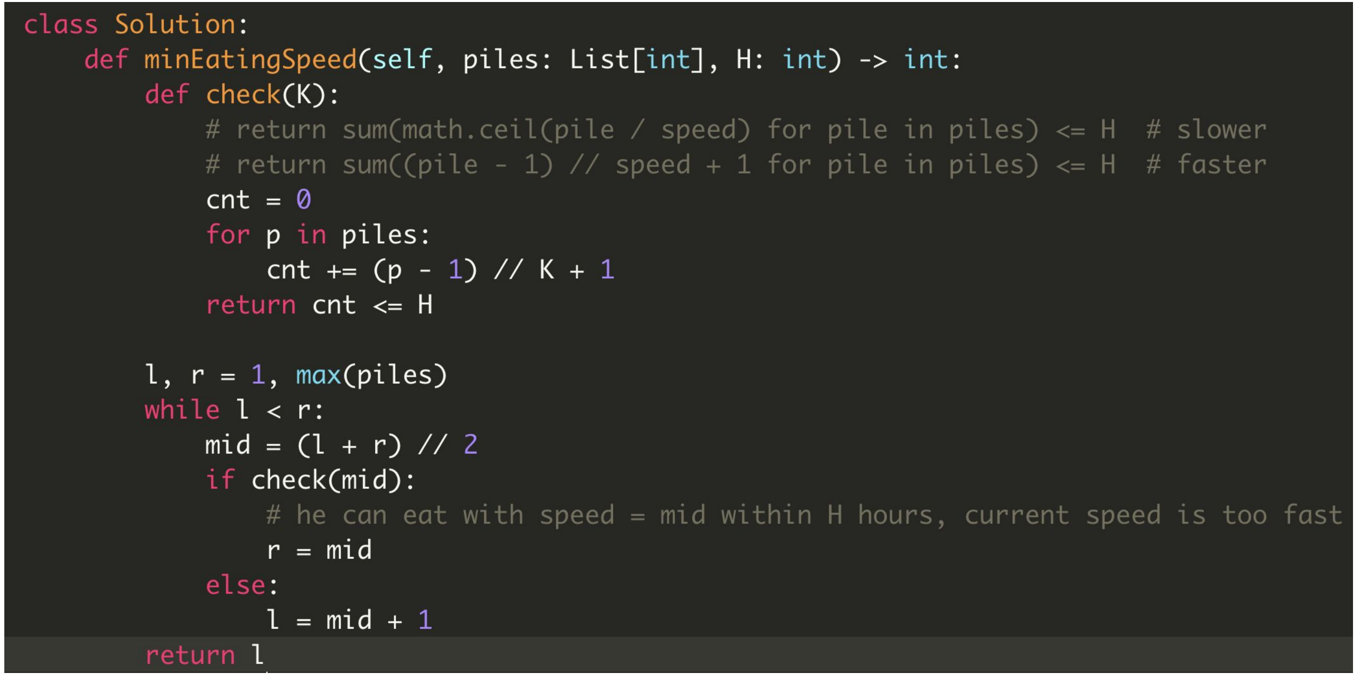
Return *the minimum number of days*
you need to wait to be able to make
m bouquets from the garden.

```
Input: bloomDay = [1,10,3,10,2
m = 3, k = 1
```

```
Output: 3
```

- bloomDay.length == n
- 1 <= n <= 10^5
- 1 <= bloomDay[i] <= 10^9
- 1 <= m <= 10^6

875. Koko eating bananas



```
class Solution:
    def minEatingSpeed(self, piles: List[int], H: int) -> int:
        def check(K):
            # return sum(math.ceil(pile / speed) for pile in piles) <= H  # slower
            # return sum((pile - 1) // speed + 1 for pile in piles) <= H  # faster
            cnt = 0
            for p in piles:
                cnt += (p - 1) // K + 1
            return cnt <= H

        l, r = 1, max(piles)
        while l < r:
            mid = (l + r) // 2
            if check(mid):
                # he can eat with speed = mid within H hours, current speed is too fast
                r = mid
            else:
                l = mid + 1
        return l
```

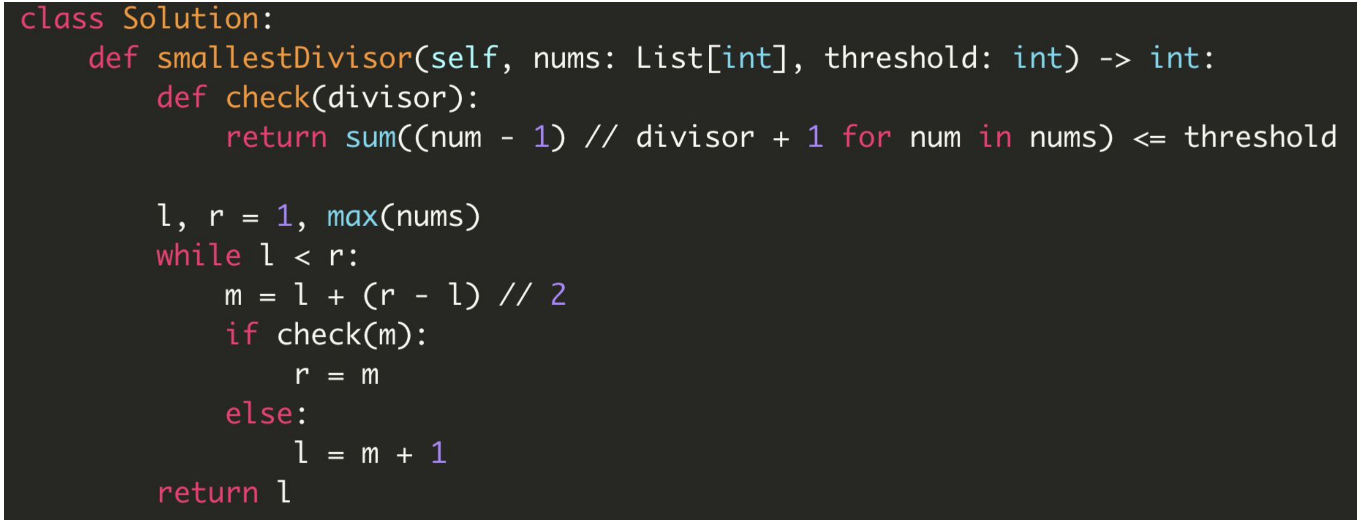
Koko can decide her bananas-per-hour eating speed of K

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

Return the minimum integer K such that she can eat all the bananas within H hours.

```
Input: piles =
[3,6,7,11], H = 8
Output: 4
```

1283. Find the Smallest Divisor Given a Threshold



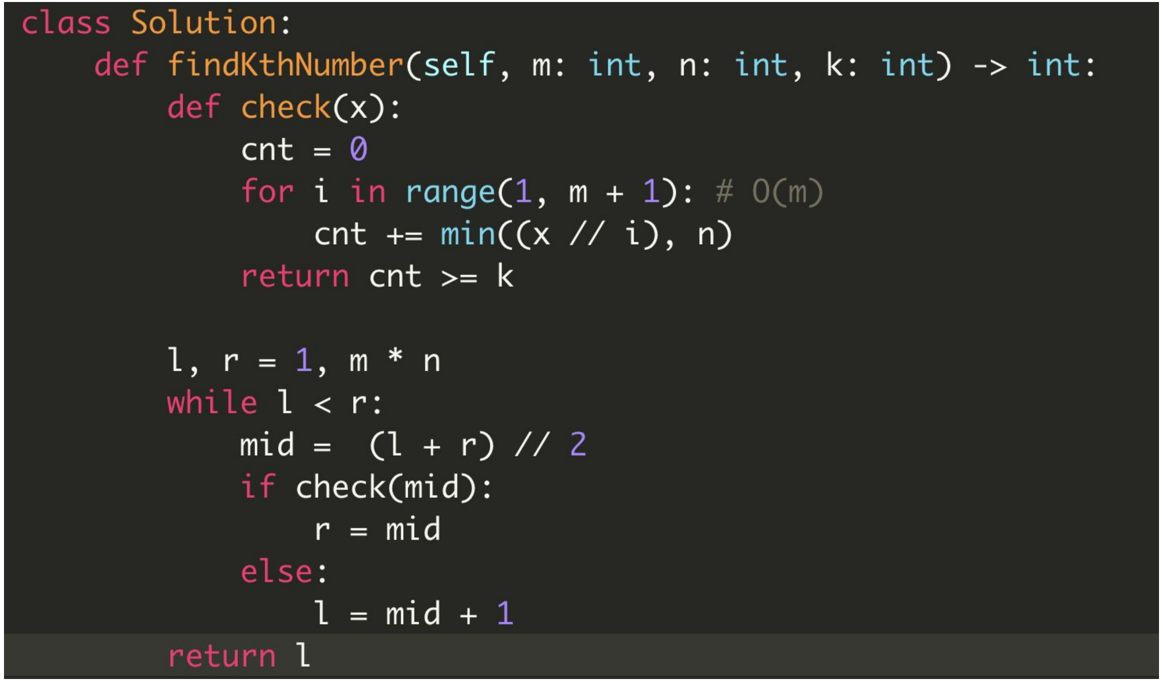
```python
class Solution:
    def smallestDivisor(self, nums: List[int], threshold: int) -> int:
        def check(divisor):
            return sum((num - 1) // divisor + 1 for num in nums) <= threshold

        l, r = 1, max(nums)
        while l < r:
            m = l + (r - l) // 2
            if check(m):
                r = m
            else:
                l = m + 1
        return l
```

Choose a positive integer divisor and divide all the array by it and sum the result of the division. Find the smallest divisor such that the result mentioned above is less than or equal to threshold.

Input: nums = [1,2,5,9], threshold = 6 Output: 5

- 1 <= nums.length <= 5 * 104
- 1 <= nums[i] <= 106
- nums.length <= threshold <= 106

668. Kth Smallest Number in Multiplication Table



```python
class Solution:
    def findKthNumber(self, m: int, n: int, k: int) -> int:
        def check(x):
            cnt = 0
            for i in range(1, m + 1): # O(m)
                cnt += min((x // i), n)
            return cnt >= k

        l, r = 1, m * n
        while l < r:
            mid =  (l + r) // 2
            if check(mid):
                r = mid
            else:
                l = mid + 1
        return l
```

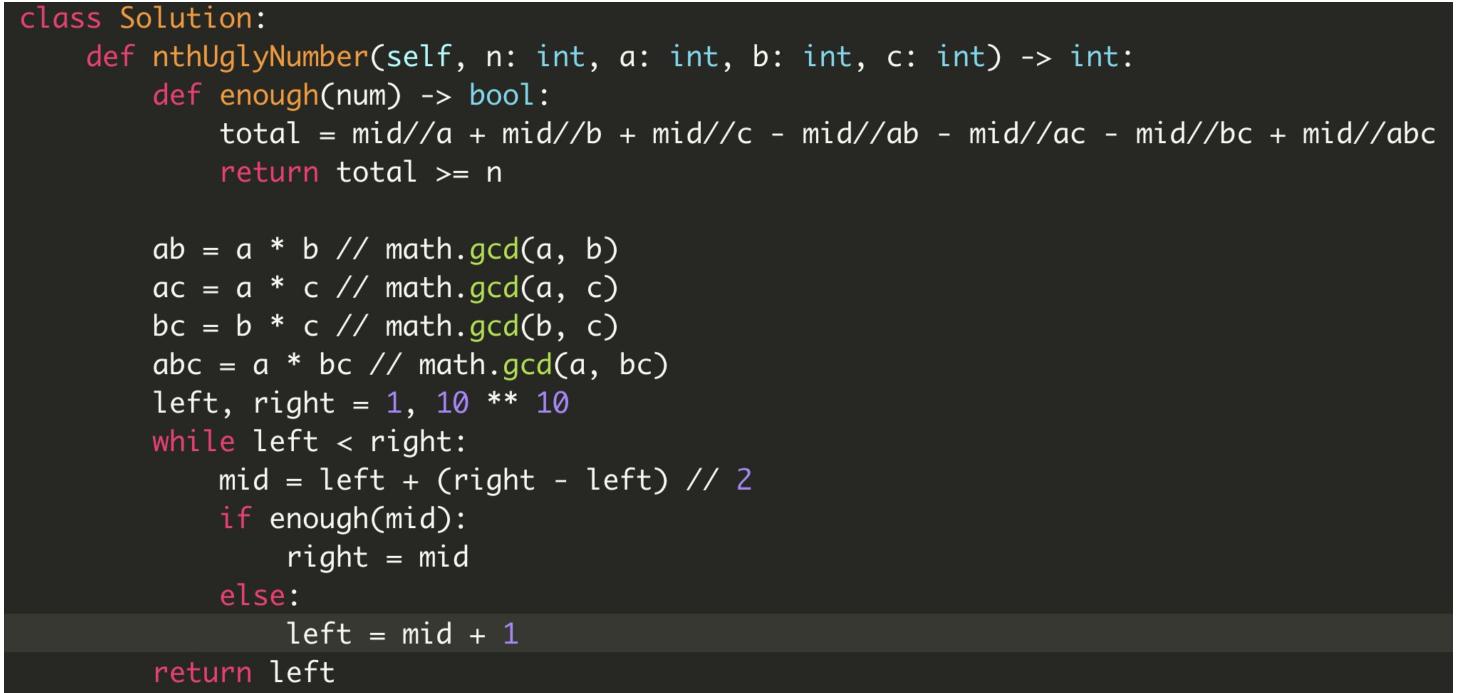
```
Input: m = 3, n = 3, k = 5

The Multiplication Table:

1	2	3

2	4	6

3	6	9
```

```
The 5-th smallest number is 3

(1, 2, 2, 3, 3).
```

1201. Ugly Number III

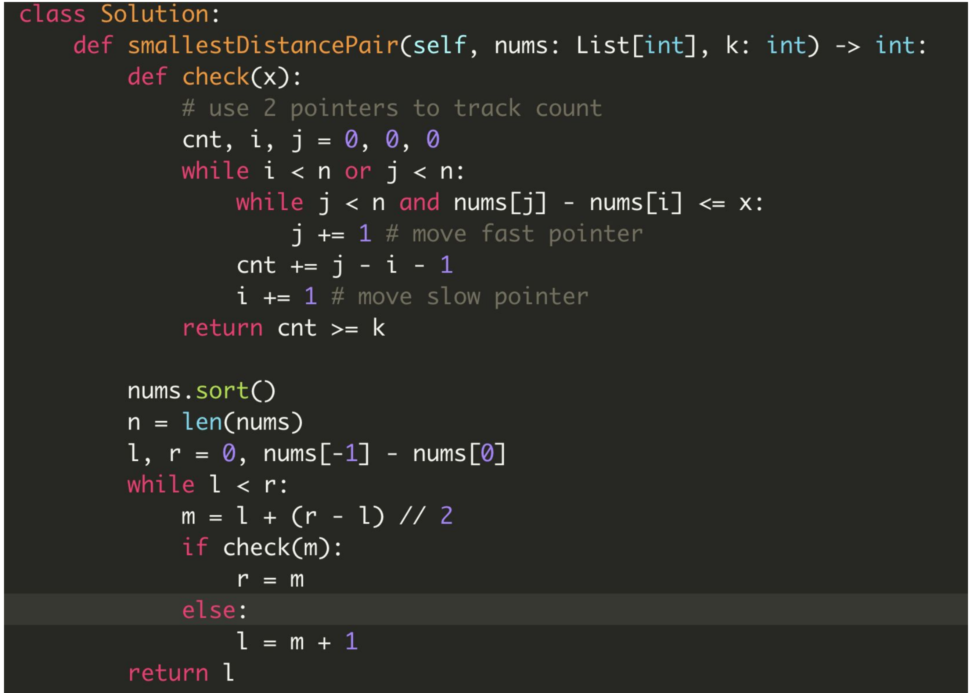
Ugly numbers are positive integers which are divisible by `a` or `b` or `c`.

least common multiple: `F(N) = N/a + N/b + N/c - N/lcm(a, c) - N/lcm(a, b) - N/lcm(b, c) + N/lcm(a, b, c)`



```python
class Solution:
    def nthUglyNumber(self, n: int, a: int, b: int, c: int) -> int:
        def enough(num) -> bool:
            total = mid//a + mid//b + mid//c - mid//ab - mid//ac - mid//bc + mid//abc
            return total >= n

        ab = a * b // math.gcd(a, b)
        ac = a * c // math.gcd(a, c)
        bc = b * c // math.gcd(b, c)
        abc = a * bc // math.gcd(a, bc)
        left, right = 1, 10 ** 10
        while left < right:
            mid = left + (right - left) // 2
            if enough(mid):
                right = mid
            else:
                left = mid + 1
        return left
```

719. Find K-th Smallest Pair Distance



```python
class Solution:
    def smallestDistancePair(self, nums: List[int], k: int) -> int:
        def check(x):
            # use 2 pointers to track count
            cnt, i, j = 0, 0, 0
            while i < n or j < n:
                while j < n and nums[j] - nums[i] <= x:
                    j += 1 # move fast pointer
                cnt += j - i - 1
                i += 1 # move slow pointer
            return cnt >= k

        nums.sort()
        n = len(nums)
        l, r = 0, nums[-1] - nums[0]
        while l < r:
            m = l + (r - l) // 2
            if check(m):
                r = m
            else:
                l = m + 1
        return l
```

Given an integer array, return the k-th smallest distance among all the pairs. The distance of a pair (A, B) is defined as the absolute difference between A and B.

```
Input: nums = [1,3,1] k = 1
```

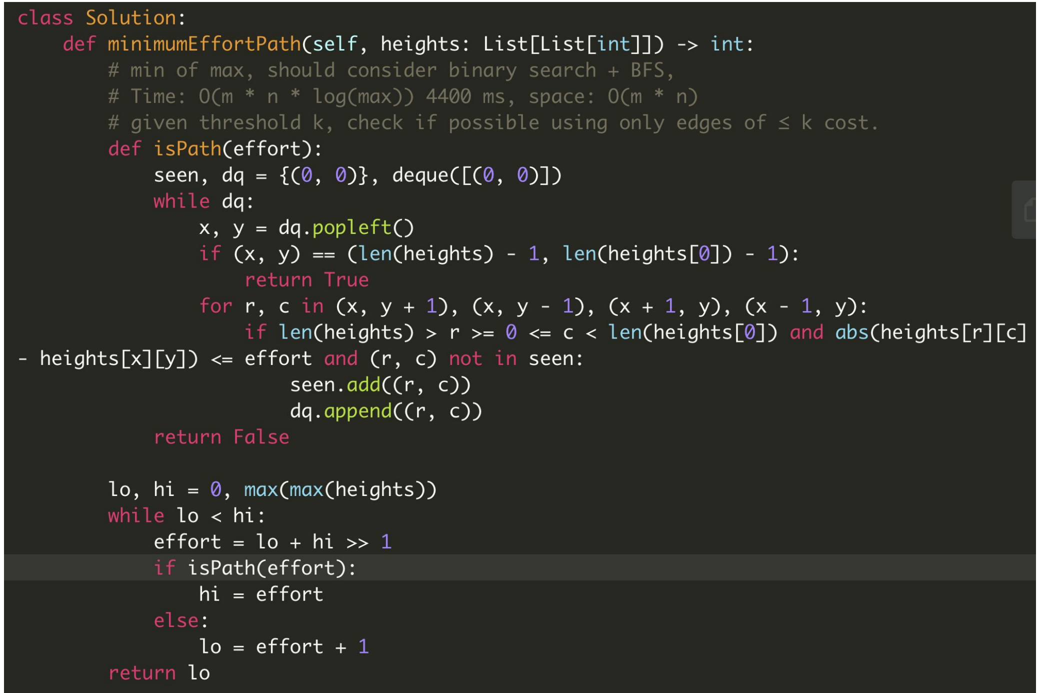
```
Output: 0
```

Here are all the pairs:

(1,3) -> 2, (1,1) -> 0, (3,1) -> 2

Then the 1st smallest distance pair is (1,1), and its distance is 0.

1631. Path With Minimum Effort



```
class Solution:
    def minimumEffortPath(self, heights: List[List[int]]) -> int:
        # min of max, should consider binary search + BFS,
        # Time: O(m * n * log(max)) 4400 ms, space: O(m * n)
        # given threshold k, check if possible using only edges of ≤ k cost.
        def isPath(effort):
            seen, dq = {(0, 0)}, deque([(0, 0)])
            while dq:
                x, y = dq.popleft()
                if (x, y) == (len(heights) - 1, len(heights[0]) - 1):
                    return True
                for r, c in (x, y + 1), (x, y - 1), (x + 1, y), (x - 1, y):
                    if len(heights) > r >= 0 <= c < len(heights[0]) and abs(heights[r][c]
- heights[x][y]) <= effort and (r, c) not in seen:
                        seen.add((r, c))
                        dq.append((r, c))
            return False

        lo, hi = 0, max(max(heights))
        while lo < hi:
            effort = lo + hi >> 1
            if isPath(effort):
                hi = effort
            else:
                lo = effort + 1
        return lo
```

A route's effort is the maximum absolute difference in heights between two consecutive cells of the route.

Return *the minimum effort required to travel from the top-left cell to the bottom-right cell.*

- `rows == heights.length`
- `columns == heights[i].length`
- `1 <= rows, columns <= 100`
- `1 <= heights[i][j] <= 106`

Consider the grid as a graph, where adjacent cells have an edge with cost of the difference between the cells.

If you are given threshold k, check if it is possible to go from (0, 0) to (n-1, m-1) using