Allocate Minimum Pages - GeeksforGeeks

Source: https://www.geeksforgeeks.org/allocate-minimum-number-pages/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Allocate Minimum Pages Last Updated : 12 Aug, 2025 Given an array arr[] , where arr[i] represents the number of pages in the i-th book, and an integer k denoting the total number of students, allocate all books to the students such that: Each student gets at least one book. Books are allocated in a contiguous sequence. The maximum number of pages assigned to any student is minimized. If it is not possible to allocate all books among k students under these conditions, return -1. Examples: Input: arr[] = [12, 34, 67, 90], k = 2 Output: 113 Explanation: Books can be distributed in following ways: [12] and [34, 67, 90] - The maximum pages assigned to a student is 34 + 67 + 90 = 191. [12, 34] and [67, 90] - The maximum pages assigned to a student is 67 + 90 = 157. [12, 34, 67] and [90] - The maximum pages assigned to a student is 12 + 34 + 67 = 113. The third combination has the minimum pages assigned to a student which is 113. Input: arr[] = [15, 17, 20], k = 5 Output: -1 Explanation: Since there are more students than total books, it's impossible to allocate a book to each student. Input: arr[] = [22, 23, 67], k = 1 Output: 112 Explanation: Since there is only 1 student, all books are assigned to that student. So, maximum pages assigned to a student is 22 + 23 + 67 = 112. Try it on GfG Practice Table of Content [Naive Approach] By Iterating Over All Possible Page Limits [Expected Approach] Using Binary Search [Naive Approach] By Iterating Over All Possible Page Limits The approach is to search for the minimum possible value of the maximum number of pages that can be assigned to any student. The lowest possible page limit is the maximum number of pages in a single book, since that book must be assigned to some student. The highest possible page limit is the total number of pages across all books, which would happen if one student gets all the books. To check if a given page limit can work, we simulate the process of assigning books to students: We start with the first student and keep assigning books one by one until adding the next book would exceed the current page limit. At that point, we assign books to the next student, and continue this process. As soon as we find the smallest page limit that allows us to allocate all books to exactly k students, we return it. C++ #include <iostream> #include <vector> #include <algorithm> #include <numeric> using namespace std ; // function to check if books can be allocated to // all k students without exceeding 'pageLimit' bool check (vector < int > & arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . size (); i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } int findPages (vector < int > & arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . size ()) return -1 ; // minimum and maximum possible page limits int minPageLimit = * max_element (arr . begin (), arr . end ()); int maxPageLimit = accumulate (arr . begin (), arr . end (), 0); // iterating over all possible page limits for (int i = minPageLimit ; i <= maxPageLimit ; i ++) { // return the first page limit with we can // allocate books to all k students if (check (arr , k , i)) return i ; } return -1 ; } int main () { vector < int > arr = { 12 , 34 , 67 , 90 }; int k = 2 ; cout << findPages (arr , k); return 0 ; } C #include <stdio.h> #include <stdbool.h> // function to check if books can be allocated to // all k students without exceeding 'pageLimit' bool check (int arr [], int n , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < n ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned

to exactly k students as well return (cnt <= k); } int findPages (int arr [], int n , int k) { // if number of students are more than total books // then allocation is not possible if (k > n) return -1 ; // minimum and maximum possible page limits int minPageLimit = arr [0]; int maxPageLimit = 0 ; for (int i = 0 ; i < n ; i ++) { if (arr [i] > minPageLimit) minPageLimit = arr [i]; maxPageLimit += arr [i]; } // iterating over all possible page limits for (int i = minPageLimit ; i <= maxPageLimit ; i ++) { // return the first page limit with we can // allocate books to all k students if (check (arr , n , k , i)) return i ; } return -1 ; } int main () { int arr [] = { 12 , 34 , 67 , 90 }; int k = 2 ; int n = sizeof (arr) / sizeof (arr [0]); printf ("%d \n " , findPages (arr , n , k)); return 0 ; } Java import java.util.Arrays ; class GfG { // function to check if books can be allocated to // all k students without exceeding 'pageLimit' static boolean check (int [] arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i] ; } else { pageSum += arr [i] ; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } static int findPages (int [] arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . length) return - 1 ; // minimum and maximum possible page limits int minPageLimit = Arrays . stream (arr). max (). getAsInt (); int maxPageLimit = Arrays . stream (arr). sum (); // iterating over all possible page limits for (int i = minPageLimit ; i <= maxPageLimit ; i ++) { // return the first page limit with we can // allocate books to all k students if (check (arr , k , i)) return i ; } return - 1 ; } public static void main (String [] args) { int [] arr = { 12 , 34 , 67 , 90 }; int k = 2 ; System . out . println (findPages (arr , k)); } } Python def check (arr , k , pageLimit): # starting from the first student cnt = 1 pageSum = 0 for pages in arr : # if adding the current book exceeds the page # limit, assign the book to the next student if pageSum + pages > pageLimit : cnt += 1 pageSum = pages else : pageSum += pages # if books can assigned to less than k students then # it can be assigned to exactly k students as well return cnt <= k def findPages (arr , k): # if number of students are more than total books # then allocation is not possible if k > len (arr): return - 1 # minimum and maximum possible page limits minPageLimit = max (arr) maxPageLimit = sum (arr) # iterating over all possible page limits for i in range (minPageLimit , maxPageLimit + 1): # return the first page limit with we can # allocate books to all k students if check (arr , k , i): return i return - 1 if __name__ == "__main__" : arr = [12 , 34 , 67 , 90] k = 2 print (findPages (arr , k)) C# using System ; using System.Linq ; class GfG { // function to check if books can be allocated to // all k students without exceeding 'pageLimit' static bool check (int [] arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . Length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } static int findPages (int [] arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . Length) return - 1 ; // minimum and maximum possible page limits int minPageLimit = arr . Max (); int maxPageLimit = arr . Sum (); // iterating over all possible page limits for (int i = minPageLimit ; i <= maxPageLimit ; i ++) { // return the first page limit with we can // allocate books to all k students if (check (arr , k , i)) return i ; } return - 1 ; } static void Main () { int [] arr = { 12 , 34 , 67 , 90 }; int k = 2 ; Console . WriteLine (findPages (arr , k)); } } JavaScript // function to check if books can be allocated to // all k students without exceeding 'pageLimit' function check (arr , k , pageLimit) { // starting from the first student let cnt = 1 ; let pageSum = 0 ; for (let i = 0 ; i < arr . length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } function findPages (arr , k) { // if number of students are more than total books // then allocation is not possible if (k > arr . length) return - 1 ; // minimum and maximum possible page limits const minPageLimit = Math . max (... arr); const maxPageLimit = arr . reduce ((a , b) => a + b , 0); // iterating over all possible page limits for (let i = minPageLimit ; i <= maxPageLimit ; i ++) { // return the first page limit with we can // allocate books to all k students if (check (arr , k , i)) return i ; } return - 1 ; } // Driver Code const arr = [12 , 34 , 67 , 90]; const k = 2 ; console . log (findPages (arr , k)); Output 113 Time Complexity: O(n × (sum(arr) - max(arr))), where n is the total number of books, sum(arr) is the total number of pages in all the books and max(arr) is maximum number of pages in any book. Auxiliary Space: O(1) [Expected Approach] Using Binary Search The maximum number of pages (page limit) that a student can be allocated has a monotonic property: If at a page limit p , books cannot be allocated to all k students, then we need to reduce the page limit to ensure more students receive

books. If at a page limit p , we can allocate books to more than k students, then we need to increase the page limit so that fewer students are allocated books. Therefore, we can apply binary search to minimize the maximum pages a student can be allocated. To check the number of students that can be allotted books for any page limit, we start assigning books to the first student until the page limit is reached, then move to the next student. C++ #include <iostream> #include <vector> #include <algorithm> #include <numeric> using namespace std ; // function to check if books can be allocated to // all k students without exceeding 'pageLimit' bool check (vector < int > & arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . size (); i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } int findPages (vector < int > & arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . size ()) return -1 ; // search space for Binary Search int lo = * max_element (arr . begin (), arr . end ()); int hi = accumulate (arr . begin (), arr . end (), 0); int res = -1 ; while (lo <= hi) { int mid = lo + (hi - lo) / 2 ; if (check (arr , k , mid)){ res = mid ; hi = mid - 1 ; } else { lo = mid + 1 ; } } return res ; } int main () { vector < int > arr = { 12 , 34 , 67 , 90 }; int k = 2 ; cout << findPages (arr , k); return 0 ; } C #include <stdio.h> #include <stdbool.h> // function to check if books can be allocated to // all k students without exceeding 'pageLimit' bool check (int arr [], int n , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < n ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } int findPages (int arr [], int n , int k) { // if number of students are more than total books // then allocation is not possible if (k > n) return -1 ; // maximum element of the array is minimum page limit int lo = arr [0]; for (int i = 1 ; i < n ; i ++) if (arr [i] > lo) lo = arr [i]; // summation of all element is maximum page limit int hi = 0 ; for (int i = 0 ; i < n ; i ++) hi += arr [i]; int res = -1 ; while (lo <= hi) { int mid = lo + (hi - lo) / 2 ; if (check (arr , n , k , mid)){ res = mid ; hi = mid - 1 ; } else { lo = mid + 1 ; } } return res ; } int main () { int arr [] = { 12 , 34 , 67 , 90 }; int k = 2 ; int n = sizeof (arr) / sizeof (arr [0]); printf ("%d \n " , findPages (arr , n , k)); return 0 ; } Java import java.util.Arrays ; class GfG { // function to check if books can be allocated to // all k students without exceeding 'pageLimit' static boolean check (int [] arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i] ; } else { pageSum += arr [i] ; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } static int findPages (int [] arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . length) return - 1 ; // search space for Binary Search int lo = Arrays . stream (arr). max (). getAsInt (); int hi = Arrays . stream (arr). sum (); int res = - 1 ; while (lo <= hi) { int mid = lo + (hi - lo) / 2 ; if (check (arr , k , mid)){ res = mid ; hi = mid - 1 ; } else { lo = mid + 1 ; } } return res ; } public static void main (String [] args) { int [] arr = { 12 , 34 , 67 , 90 }; int k = 2 ; System . out . println (findPages (arr , k)); } } Python # function to check if books can be allocated to # all k students without exceeding 'pageLimit' def check (arr , k , pageLimit): # starting from the first student cnt = 1 pageSum = 0 for pages in arr : # if adding the current book exceeds the page # limit, assign the book to the next student if pageSum + pages > pageLimit : cnt += 1 pageSum = pages else : pageSum += pages # if books can assigned to less than k students then # it can be assigned to exactly k students as well return cnt <= k def findPages (arr , k): # if number of students are more than total books # then allocation is not possible if k > len (arr): return - 1 # search space for Binary Search lo = max (arr) hi = sum (arr) res = - 1 while lo <= hi : mid = lo + (hi - lo) // 2 if check (arr , k , mid): res = mid hi = mid - 1 else : lo = mid + 1 return res if __name__ == "__main__" : arr = [12 , 34 , 67 , 90] k = 2 print (findPages (arr , k)) C# using System ; using System.Linq ; class GfG { // function to check if books can be allocated to // all k students without exceeding 'pageLimit' static bool check (int [] arr , int k , int pageLimit) { // starting from the first student int cnt = 1 ; int pageSum = 0 ; for (int i = 0 ; i < arr . Length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } static int findPages (int [] arr , int k) { // if number of students are more than total books // then allocation is not possible if (k > arr . Length) return - 1 ; // search space for Binary Search int lo = arr . Max (); int hi = arr . Sum (); int res = - 1 ; while (lo <= hi) { int mid = lo + (

hi - lo) / 2 ; if (check (arr , k , mid)){ res = mid ; hi = mid - 1 ; } else { lo = mid + 1 ; } } return res ; } static void Main () { int [] arr = { 12 , 34 , 67 , 90 }; int k = 2 ; Console . WriteLine (findPages (arr , k)); } } JavaScript // function to check if books can be allocated to // all k students without exceeding 'pageLimit' function check (arr , k , pageLimit) { // starting from the first student let cnt = 1 ; let pageSum = 0 ; for (let i = 0 ; i < arr . length ; i ++) { // if adding the current book exceeds the page // limit, assign the book to the next student if (pageSum + arr [i] > pageLimit) { cnt ++ ; pageSum = arr [i]; } else { pageSum += arr [i]; } } // if books can assigned to less than k students then // it can be assigned to exactly k students as well return (cnt <= k); } function findPages (arr , k) { // if number of students are more than total books // then allocation is not possible if (k > arr . length) return - 1 ; // search space for Binary Search let lo = Math . max (... arr); let hi = arr . reduce ((a , b) => a + b , 0); let res = - 1 ; while (lo <= hi) { let mid = lo + Math . floor ((hi - lo) / 2); if (check (arr , k , mid)){ res = mid ; hi = mid - 1 ; } else { lo = mid + 1 ; } } return res ; } // Driver Code const arr = [12 , 34 , 67 , 90]; const k = 2 ; console . log (findPages (arr , k)); Output 113 Time Complexity: O(n × log(sum(arr) - max(arr))), where n is the total number of books, sum(arr) is the total number of pages in all the books and max(arr) is maximum number of pages in any book. Auxiliary Space: O(1) Comment Article Tags: Article Tags: Divide and Conquer Searching DSA Google Binary Search + 1 More