**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：714: The Copying Books**

**解題者：朱仲誠**

**解題日期：2025年5月22日**

**題意：**

給定 k 位抄寫員、m 本書，各有, …, 頁。目標：m 本書分配給 k 位抄寫員抄寫。

要求每位抄寫員必須分到一段連續編號的書、每位抄寫員至少要分到一本書以及負責總頁數最多的抄寫員，其總頁數要盡量小。

輸出滿足要求的分配方式，若有多組解，則讓較後面的抄寫員分配較多的書。

**題意範例：**

 輸入：

2 // 2 個 case

9 3 // 9 本書、3 個抄寫員

100 200 300 400 500 600 700 800 900 // 9 本書頁數

5 4 // 5 本書、4 個抄寫員

100 100 100 100 100

 輸出：

100 200 300 400 500 / 600 700 / 800 900

100 / 100 / 100 / 100 100 // 讓後面的人分配較多的書

**解法：**

定義：給定一種滿足要求的分配方式。若每位抄寫員分到的總頁數皆不超過 x，則稱 x 為合法的頁數總和。

使用 greedy method 檢查 x 頁數總和是否合法。

使用Binary Search 尋找最小合法頁數總和。

最後依照此值分配給每位抄寫員即可。

**解法範例：**

範例：3 位抄寫員，9 本書各有 9, 8, 1, 7, 6, 2, 3, 4, 5 頁。

步驟一，設定兩個變數。Reject：其值為當下不合法的頁數總和最大值。比 Reject 值小的值皆非法。初始化為所有書中最小頁數。Accept：其值為當下合法的頁數總和最小值。比 Accept 值大的值皆合法。初始化為所有書總頁數之和。以此範例為例，初始化 Reject = 1；初始化 Accept = 9 + 8 + 1 + 7 + 6 + 2 + 3 + 4 + 5 = 45。

步驟二，對 Reject 到 Accept 之間做 Binary Search。若中間值 mid 為合法頁數總和，則將 mid 設為下一個 Accept。若中間值 mid 為非法頁數總和，則將 mid 設為下一個 Reject。以此範例為例，mid = (1 + 45) / 2 = 23。依照 9 / 8 / 1 6 2 3 4 5 的分法可以分配給三個抄寫員，故mid = 23 為合法頁數總和，所以設下一個 Accept = 23。下一個 mid = (1 + 23) / 2 = 12，無法分配給三位抄寫員，故 mid = 12 為非法頁數總和，所以設 Reject = 12。持續做步驟二直到符合步驟三的條件。

步驟三，當 Reject + 1 == Accept 時，停止 Search，此時 Accept 值即為最小合法頁數總和。以此範例為例，持續做 Step2 直到 Reject + 1 == Accept ，得到 Reject = 16； Accept = 17。故 17 即為最小合法頁數總和。

步驟四，從後往前根據最小最大值重建分配方案，使用 stack 儲存輸出順序，並標記 / 區隔每位抄寫員的書。以此範例為例，得到 stack 中儲存的元素由底部到頂端依序為 5, 4, 3, 2, /, 6, 7, 1, /, 8, 9。

步驟五，依序將 stack 中存放的元素 pop 出來後印出即可。以此範例為例，依序 pop stack 後印出 9 8 / 1 7 6 / 2 3 4 5。

**討論：**

給定 k 位抄寫員、m 本書，各有 , …, 。使用 Binary Search 從 到 尋找最小合法頁數總和。每一次確認是否為合法頁數總和需要 ，故時間複雜度為 。

若從 一路暴力檢查到 檢查到的第一個合法頁數總和即為最小合法頁數總和。時間複雜度會達到為 。

**程式：**

#include <iostream>

#include <vector>

#include <stack>

using namespace std;

int N, M, K;  // N 個 case, M 本書, K 個工人

int main()

{

    cin >> N;

    while(N--) {

        cin >> M >> K;

        vector<int> books(M);

        long long reject = 0, accept = 0, mid;

        for (int i = 0; i < M; i++) {

            cin >> books[i];

            if (books[i] > reject) reject = books[i];

            accept += books[i];

        }

        // 用 binary search 找

        while(accept-1 > reject) {

            mid = (accept + reject + 1) / 2;

            int temp = 0, mans = K-1;

            for(int i = 0; i < M; i++) {

                if (books[i] > mid) {  // 單本書超過可接受範圍

                    mans = -1; break;

                }

                if (temp + books[i] <= mid) {

                    temp += books[i];

                } else {

                    temp  = books[i]; mans--;

                }

            }

            if(mans >= 0) accept = mid;

            else reject = mid;

        }

        stack<string> S;

        long long temp = 0, j = K-1;

        for (int i = M-1; i >= 0; i--) {

            if (i+1 == j) {

                S.push("/");

                j--;

            } else if (temp + books[i] <= accept) {

                temp += books[i];

            } else {

                S.push("/");

                temp  = books[i];

                j--;

            }

            S.push(to\_string(books[i]));

        }

        int first = 1;

        while(!S.empty()) {

            if(!first) cout << " ";

            first = 0;

            cout << S.top();

            S.pop();

        }

        cout << endl;

    }

}