**算法训练 接水问题**

时间限制：1.0s   内存限制：64.0MB

问题描述

　　学校里有一个水房，水房里一共装有m 个龙头可供同学们打开水，每个龙头每秒钟的 供水量相等，均为1。 现在有n 名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1 到n 编号，i 号同学的接水量为wi。接水开始时，1 到m 号同学各占一个水龙头，并同时打 开水龙头接水。当其中某名同学j 完成其接水量要求wj 后，下一名排队等候接水的同学k 马上接替j 同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即 j 同学第x 秒结束时完成接水，则k 同学第x+1 秒立刻开始接水。若当前接水人数n’不足m， 则只有n’个龙头供水，其它m−n’个龙头关闭。 现在给出n 名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。

输入格式

　　第1 行2 个整数n 和m，用一个空格隔开，分别表示接水人数和龙头个数。 第2 行n 个整数w1、w2、……、wn，每两个整数之间用一个空格隔开，wi 表示i 号同 学的接水量。

输出格式

　　输出只有一行，1 个整数，表示接水所需的总时间。

样例输入

5 3  
4 4 1 2 1

样例输出

4

样例输入

8 4  
23 71 87 32 70 93 80 76

样例输出

163

输入输出样例 1 说明

　　第1 秒，3 人接水。第1 秒结束时，1、2、3 号同学每人的已接水量为1，3 号同学接完  
　　水，4 号同学接替3 号同学开始接水。  
　　第2 秒，3 人接水。第2 秒结束时，1、2 号同学每人的已接水量为2，4 号同学的已接  
　　水量为1。  
　　第3 秒，3 人接水。第3 秒结束时，1、2 号同学每人的已接水量为3，4 号同学的已接  
　　水量为2。4 号同学接完水，5 号同学接替4 号同学开始接水。  
　　第4 秒，3 人接水。第4 秒结束时，1、2 号同学每人的已接水量为4，5 号同学的已接  
　　水量为1。1、2、5 号同学接完水，即所有人完成接水。  
　　总接水时间为4 秒。

数据规模和约定

　　1 ≤ n ≤ 10000，1 ≤m≤ 100 且m≤ n；  
　　1 ≤ wi ≤ 100。

锦囊1

模拟即可，要加速可以使用堆优化。

锦囊2

本题的数据范围比较小，可以直接按照题库模拟，或者也可以使用堆来优化算法。

本题的C++参考代码如下：

#include "iostream"

#include "string"

#include "stdio.h"

#include "ctype.h"

#include "algorithm"

#include "stack"

#include "list"

#include "math.h"

using namespace std;

const int N =101;

struct X

{

int total;

int now;

X(int t=0,int n=0):total(t),now(n){}

}a[10001],b[201];

int main()

{

int n,m;

scanf("%d%d",&n,&m);

for(int i=0;i<n;i++)

scanf("%d",&a[i].total);

int index=0;

for(int i=0;i<m;i++)

{

b[i]=a[ index++];

}

int time=0;

while(true)

{

time++;

for(int i=0;i<m;i++)

b[i].now++;

for(int i=0;i<m;i++)

{

if(b[i].now>=b[i].total)

{

if(index<n)

{

b[i]=a[index++];

}

}

}

int i;

for ( i=0;i<m;i++)

{

if(b[i].now<b[i].total)

break;

}

if(i==m)

break;

}

cout<<time<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int max(int \*p,int n)

{

int Max = p[0];

int i;

for(i=1;i<n;i++)

{

if(p[i]>Max)

Max = p[i];

}

return Max;

}

int main(int argc, char \*argv[]) {

int m,n; //m个水龙头 n个同学

int i,j,second,min,num;

int \*p;

scanf("%d%d",&n,&m);

p = (int \*)malloc(n\*sizeof(int));

if( NULL == p )

{

printf("no enough memory!\n");

return 0;

}

for(i=0;i<n;i++)

{

scanf("%d",&p[i]);

}

if( n<=m)

{

printf("%d",max(p,n));

}

if( n>m)

{

second = 0;

num = 0;

while(n>m)

{

min = p[0];

j = 0;

for(i=0;i<m;i++)

{

if(min>p[i])

{

min = p[i];

j=i;

}

}

for(i=0;i<m;i++)

{

p[i] -= min;

}

second += min;

p[j]=p[m+num]; //第m+num同学取代取完水的同学

num++;

n--;//取水同学减少一个

if( n<=m)

{

printf("%d",max(p,n)+second);

}

}

}

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(

new InputStreamReader(System.in));

String[] str1 = br.readLine().split(" ");

String[] str2 = br.readLine().split(" ");

int[] first = new int[2];

for (int i = 0; i < first.length; i++)

first[i] = Integer.parseInt(str1[i]);

int[] student = new int[first[0]];

for (int i = 0; i < student.length; i++) {

student[i] = Integer.parseInt(str2[i]);

}

if (first[0] > first[1])

System.out.println(getwater(student, first[1]));

else

System.out.println(getmax(student));

}

public static int getwater(int[] student, int watercome) {

int time = 0;

int[] firststudents = new int[watercome];

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

firststudents[i] = student[i];

}

int k = firststudents.length;

for (int i = 0; i <= student.length - watercome; i++) {

if (i == student.length - watercome)

time += getmax(firststudents);

else {

int min = getmin(firststudents);

for (int j = 0; j < firststudents.length; j++) {

firststudents[j] -= min;

if (firststudents[j] == 0 && k<student.length) {

firststudents[j] = student[k++];

}

}

time += min;

}

}

return time;

}

public static int getmin(int[] arr) {

Arrays.sort(arr);

return arr[0];

}

public static int getmax(int[] arr) {

Arrays.sort(arr);

return arr[arr.length - 1];

}

}