# beer 1



Разполагаме с S лева (1 ≤ S ≤ 1 000 000), които искаме да похарчим за бира. Заведението, в който се намираме, предлага N вида бира (2 ≤ N ≤ 20 000), всяка на цена Li (1 ≤ Li ≤1 000 000). Ще купуваме комплекти от по 2 вида бира. Кои точно комплекта ще купим, решаваме след като прегледаме всички възможности. Разбира се, само с наличните пари. От колко различни комплекта трябва да направим избора си?

ЖОКЕР: Ако пробваме сумата от цените на всички двойки бири ще извършим 20000 \* 19999 / 2 = 199,990,000 прости операции. Това все още е в позволените рамки от 1 милиард прости операции. Но след като умножим това по максималния брой тестове 12 ще получим 199,990,000 \* 12 = 2,399,880,000, което е повече от 1 милиард и вашата програма най-вероятно няма да мине по време освен, ако автора на тестовете за задачата не е прекалено добродушен и/или несъобразителен.

За да влезете със сигурност по време, първо сортирайте видовете бира по цена в масив prices. След това използувайте два 0-базирани индекса IMAX и IMIN в масива със сортираните цени. Първият индекс IMAX ще ви дава позицията на по-скъпата от двойката бири - инициализирате го с индекса на най-скъпата бира (IMAX=N-1) в сортирания масив prices и го намаляйте с едно докато prices[IMAX] + prices[0] > S, т.е инициализираме го с индекса на най-скъпата бира, която може да се комбинира с поне една (ако не може да се комбинира с най-евтината, няма да може и с която и да е друга). Вторият индекс IMIN ще ви дава позицията на най-скъпата втора бира (но по евтина от тази на позиция IMAX), която можем да си позволим при положение, че другата бира е с цена prices[IMAX]. Инициализирайте го с 0 и го увеличавайте с едно докато prices[IMAX] + prices[IMIN] <= S. След като сте инициализирали двата индекса правилно вече знате че за вида бира с цена prices[IMAX] може да си позволите всички бири с цена от prices[0] до prices[IMIN]. Те са IMIN+1 на брой. Инициализирайте променлива в която ще трупаме отговора на задачата с 0.

НАЧАЛО ЦИКЪЛ: Ако все още IMAX > IMIN, прибавете към променлива в която ще трупаме отговора на задачата стойността (IMIN+1) - това е броят бири с който може да се комбинира тази с цена prices[IMAX текущо], иначе преминете към 'ФИНАЛНА ЧАСТ'. След това намалете IMAX с едно и намерете новата максимална стойност на IMIN (увеличавайте го с едно, толкова пъти, колкото е възможно, докато prices[IMAX] + prices[IMIN] <= S - както при инизиализацията на IMIN). Повторете цикъла от 'НАЧАЛО ЦИКЪЛ'.

ФИНАЛНА ЧАСТ: Когато това условие (IMAX > IMIN) вече не е изпълнено вие можете да комбинирате всичките бири с индекси от 0 до IMAX(текущо) всяка с всяка без да надвишите цената S. Те са IMAX(текущо) + 1 на брой. Броят на всички двойки комбинации от К елемента е К\*(K-1)/2. В нашия случай K=IMAX+1, така че те са (IMAX+1)\*IMAX/2. Добавете тази стойност към променливата в която трупате отговора за дадения тест и това е крайният отговор за него.

Сложността на алгоритъма е O(N\*logN + 2 \* N) = O(N\*(logN+2)) = O(N\*logN), тъй като имаме едно сортиране в началото, а след това движим IMAX и IMIN максимум N пъти. '+2' е константа затова можем да я махнем при пресмятане на O.

#### Input Format

Входът съдържа Т тестови примера. Първият ред на всеки от тях започва с целите числата N и S. Следват N реда, съдържащи целите числа Li. – цената на i-тия вид бира. Входът завършва с две нули.

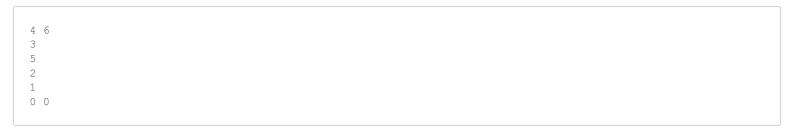
#### Constraints

 $1 \le T \le 12$   $1 \le S \le 1$  000 000  $2 \le N \le 20$  000  $1 \le Li \le 1$  000 000

#### **Output Format**

За всеки пореден тестов пример трябва да се изведе търсения брой различни комплекти от по 2 вида бира, които бихме могли да си купим с наличните пари, и съответно да изпием след това.

## Sample Input 0



### Sample Output 0

