Променливи

Дадени са n променливи a1, a2, ..., an. За всяко фиксирано k, 1 ≤ k ≤ n, да образуваме израза, който е сума от всички различни произведения на k променливи, взети измежду дадените. Например при n = 3 и k = 2 този израз е a1a2+a1a3+a2a3. Напишете програма, която да намери най-голямата по модул m стойност на всичи такива изрази за k = 1,2,...,n, при зададени стойности на променливите.

Жокер: Ново динамично програмиране - нов късмет. Модул от m е даден от авторите само за да ви спаси от препълването на long long и нуждата от използуване на класове за BigInteger, каквито няма в C++ STL. Просто след всяко изчисление на израз в програмата ще му правите % m и това ще ви реши проблема с него. В тази задача динамичната ни таблица ще е двумерна. Първото измерение (1..n) ще отразява от колко променливи ще образуваме суми (многочлени), а второто измерение (1..n) ще означава, колко от променливите (k) ще умножаваме във всеки едночлен на дадения многочлен. Последния ред ще съдържа всичките многочлени (по %m) от n-те променливи. В първата му колонка ще имаме многочлена с едночлени от една променлива (a1 + a2+ .. an) % m. Във втората му колонка ще имаме многочлена, който е сума от всики възможни произведения от две от променливите (a1*a2 + a1*a3 ... +a1*an + a2*a3 + a2*a4 + ... + ... + $a_{n-1}a_n$) % m и т.н. *Последната колонка ще съдържа само един едночлен:* ($a1$a2*a3*...*a_{n-1}*a_n)%m. Явно максимума на елементите от последния ред е отговора на задачата. Ще разпишем таблицата за n = 4. В нея забравихме да напишем (...)%m във всяка от клетките. Добавете го в кода си.

n \ k	0	1	2	3	4
0	1	0	0	0	0
1	1	0+a1*(1) = a1	0	0	0
2	1	a1+a2*1 =a1+a2	0 + a2*(a1) =a1*a2	0	0
3	1	(a1+a2) +a3*1 = a1+a2+a3	(a1*a2) + a3*(a1+a2) = a1*a2+a2*a3+a1*a3	0 + a3 * (a1 * a2) = a1 * a2 * a3	0
4	1	(a1+a2+a3)+a4*1 = a1+a2+a3+a4	(a1*a2+a2*a3+a1*a3) + a4* *(a1+a2+a3) = a1*a2 + a1*a3+a1*a4+ +a2*a3+a2*a4+a3*a4	a1*a2*a3+a4*(a1*a2+ +a2*a3+a1*a3) = a1*a2*a3+a1*a2*a4+ +a1*a3*a4+a2*a3*a4	0+ a4 * (a1*a2*a3) = a1*a2*a3*a4

Началната инициализация е DP[0][i]=0 (i=1..n) и DP[j][0]=1 (j=0..n). Гледайки таблицата, открийте сами каква е целевата функция DP[i][j] = ? и я приложете за попълване на таблицата по редове.

Паметта от 512MB няма да ви стигне за цялата таблица (12345*12345*sizeof(unsigned int) > 521MB). За да изчислите даден ред ви трябва само предния. Така че пазете само 2 реда - този който попълвате (curr_row) и предния (prev_row). След като попълните текущия ред правите prev_row=curr_row и продължавате с попълването на следващия ред. Самите елементи на таблицата ще са от тип unsigned int, но когато изчислявате стойността на следващия елемент от таблицата работете в unsigned long long. След като го изчислите и накрая му приложите модул от m, резултатът спокойно може да бъде конвертиран (неявно) към unsigned int за да бъде записан в DP, тъй като m и (нещо % m) е < MAX_UNSIGNED_INT по условие.

Input Format

Програмата трябва да прочете от стандартния вход няколко (не повече от 10) тестови примера. Всеки тестов пример съдържа стойностите n (0 < n < 12345), m (1 < m < 12345678) и a1, a2, ..., an (ai – цели, 0 < ai < 12345, i = 1, 2, …, n). Програмата трябва да спре, когато прочете на мястото на n и m стойности, равни на нула.

Constraints

тестови примери <= 10
0 < n < 12345
1 < m < 12345678
0 < ai < 12345

Output Format

За всеки тестов пример програмата трябва да изведе на отделен ред на стандартния изход най-голямата по модул m стойност на сумите от разглеждания вид.

Sample Input 0

```
3 1000000
1 2 3
4 10
1 2 3 4
0 0
```

Sample Output 0

11
5