

$$SUM_N(n) = SUM_S(n);$$

$$SUM_W(n) = SUM_E(n);$$

$$SUM_N(i) \geq SUM_S(i) \text{ за всяко } i, i = 1, 2, \dots, n;$$

$$SUM_W(i) \geq SUM_E(i) \text{ за всяко } i, i = 1, 2, \dots, n.$$

Един ден Иванчо, гледайки поредната подредба на слънчогледите, се зачудил колко други подредби има, запазващи бабината логика. Той решил да счита, че саксиите с еднакво гледащи слънчогледи са неразличими, т.е. ако размени местата на две саксии с еднакво гледащи слънчогледи, няма да получи нова подредба. Помогнете на Иванчо – напишете програма **flower**, която прочита една конфигурация от слънчогледи и определя колко още други конфигурации съществуват.

Входните данни се състоят от два реда. На първия ред се намира четното естествено число n – броя саксии на балкона на баба Кунка. Този брой не надвишава 100. На следващия ред се намират n символа, разделени със по един интервал – това са типовете слънчогледи, разгледани от ляво надясно. С „N“ ще означаваме северен слънчоглед, с „S“ – южен, с „W“ – западен и с „E“ – източен. Изходът трябва да се състои от единствен ред с едно число – търсеният брой.

ПРИМЕР 1:

Вход:

6

N W E W S E

Изход:

29

ПРИМЕР 2:

Вход:

8

N N W E S S N S

Изход:

139

Решение:

Задачата може да се разглежда като просто обобщение на известната „задача за скобите“ (ако са ни дадени n двойки скоби, колко е броя на „правилните аритметични изрази“, които може да образуваме с тях). Тази задача има връзка с числата на Каталан, а именно, K_n изразява броя изрази с n двойки скоби.

Иначе казано, в дадената задача се пита по колко начина можем да образуваме израз с два различни типа скоби (например, кръгли и квадратни) – ако имаме a двойки кръгли и b двойки квадратни скоби, така че всеки тип скоби поотделно образуват правилен израз. Тъй като едните скоби не влияят на „правилността“ на другите, то имаме просто умножението на Каталановите им числа: $K_a * K_b$. Трябва, обаче, да вземем предвид как скобите са „размесени“. Тъй като целия израз има $2*a + 2*b$ символа, може просто да приемем, че започваме от $2(a+b)$ „празни“ клетки, в които трябва да попълним $2*a$ полета със скоби от първия тип, а останалите полета запълним със другия тип. Това, очевидно, става по $C(2(a+b), 2a)$ начина.

Т.е. задачата може да се реши чрез следната формула:

$$A = K_a * K_b * C(2(a+b), 2a) = C(2a, a) * C(2b, b) * C(2(a+b), 2a) / ((a+1)*(b+1))$$

Където a е броят „северни“ слънчогледи (спрямо условието), а b е броят „западни“.

Решение 1 („Формулата“)

Това е първият начин да се реши задачата – достатъчно е да се намери формулата и да се смята по нея.

От формулата е достатъчно очевидно, че бройката расте много бързо, и дори при малки стойности на **a** и **b**, броят надхвърля обхвата на 32-битов тип. В тестовите има само 3 примера, които се „хващат“ с 32- (че дори и 64-) битова аритметика. Най-големият отговор, който се получава при **a** = 25 и **b** = 25, е с 55 цифри. Явна е необходимостта от реализиране на „дълги числа“ под някаква форма. Тук сложността откъм писане на задачата много зависи от пътя за реализация, които ще избере състезателя. Формулата горе, макар и проста, изисква тежките операции „умножение“ и „деление“, които са доста обемисти за писане. Лесно се вижда, обаче, че човек може да мине и без тях, защото множителите и делителите навсякъде са числа до 100. Казано на „програмистски жаргон“, необходими са само операции на умножение/деление на „дълги с къси“ числа.

Един по-ефикасен и удобен метод е през цялото време да пазим числата разложени в каноничен вид (т.е. всяко число X представяме във вида $X = a^x b^y c^z \dots$). Така, например, за да умножим едно число X , което е вече в каноничен вид, с друго, Y , което не е, просто трябва разложим Y и да добавим простите множители и степените им в разлагането на X .

Пример:

$$X = 90 = 2 \cdot 3^2 \cdot 5, Y = 14 = 2 \cdot 7$$

$$X \cdot Y = 2^2 \cdot 3^2 \cdot 5 \cdot 7$$

Така пресмятането на формулата горе се свежда до разлагане на множители и добавяне/изваждане в 100-елементов масив. Разбира се, накрая числото трябва да се превърне от каноничен вид в „нормален“. Тук отново се стига до „умножение на дълго с късо“ число, което не вярвам да затрудни състезателите.

Решение 2 (Динамично оптимиране)

Вторият начин за решаване на задачата се основава на техниката „динамично оптимиране“. За човек, който не е отлично запознат с числата на Каталан, биномните коефициенти и т.н. е по-логично да тръгне по този начин. Дефинира се четириаргументната функция $f(N, S, W, E)$, с която означаваме броя начини да „довършим правилно“ израза, ако вече сме разположили N северни, S южни, W западни и E източни слънчогледа. Функцията много лесно се дефинира:

$$f(N, S, W, E) :=$$

$$0, \quad \text{ако } N > \mathbf{a} \text{ или } S > \mathbf{a} \text{ или } W > \mathbf{b} \text{ или } E > \mathbf{b} \text{ или } S > N \text{ или } E > W.$$

$$1, \quad \text{ако } N = S = \mathbf{a} \text{ и } W = E = \mathbf{b}.$$

$$f(N+1, S, W, E) + f(N, S+1, W, E) + f(N, S, W+1, E) + f(N, S, W, E+1), \text{ иначе.}$$

Последния ред може да обясним със следното разсъждение: Ако вече сме разположили известен брой слънчогледи, то на текущата позиция можем да сложим някой от четирите типа слънчогледи и отговора е сумата от броя начини да довършим правилно израза при вече зафиксирани слънчоглед на текущата позиция.

Отговорът на задачата е стойността на $f(0, 0, 0, 0)$. Директното прилагане на рекурсивната дефиниция по-горе, обаче, е непрактично, поради експоненциалната сложност на изчислението: всеки екземпляр от функцията (в общия случай) поражда 4 извиквания на същата функция. За целта ще използваме таблица `dyn[51][51][51][51]`, в която ще записваме вече пресметнатите стойности на функцията (*memoization*).

Както вече споменахме, задачата изисква „дълги числа“ в някаква форма. Тук, обаче, за разлика от комбинаторното решение, нещата са по-приятни, тъй като се изисква

само операцията „събиране“. Съществен проблем обаче започва да става обема на използваната памет: таблицата `dyn` е с 6,764,201 елемента и ако за всеки елемент отделим (например) 150 байта за едно дълго число, то заемащата памет ще стане към 1 GB. Има няколко начина да се разреши този проблем, между които:

- Да се определи каква е максималната големина на числата, които ще се съхраняват в таблицата. Това може да стане и експериментално. При една достатъчно пестелива реализация на дълги числа, необходимата памет за таблицата става около 200 MB.
- Лесно се забелязва, че не всички клетки в таблицата ще се използват. Например, при $a = 25$, $b = 25$, таблица `dyn[26][26][26][26]` би ни свършила работа, а при $a = 48$, $b = 2$, дори по-малка таблица стига: `dyn[49][49][3][3]`. Това наблюдение може да се приложи по два начина: единият е да се задели толкова голяма таблица, колкото е нужно; вторият е, в таблицата да се съхраняват само указатели към числа, вместо самите числа, и паметта за всяка клетка да се заделя само когато е нужно.

Решение 3 (Прекалкулиране)

С използването на кой да е от двата начина горе, но без да се решават всички тънкости, е лесно да се изчислят отговорите за всеки възможен входен пример. Тъй като отговорите са симетрични за a и b , достатъчно е да се прекалкулират около 600 отговора.

Веселин Георгиев

Задача А3. ХРАНА

Гошо Ламерът е още ученик. Но вместо да ходи редовно на училище, той ходи редовно в близкия игрален клуб, където играе докато му свършат джобните. Един ден, когато това се случи, вече беше следобед и Гошо беше огладнял. Той отиде в близката верига за бързо хранене и застана пред вкусното меню с празни джобове. В заведението имаше много различни храни и напитки както и много промоции, с които ако човек си вземе например пица и кола спестява два лева. Може Гошо да е с празни джобове, но не е глупав. Той забеляза, че системата, която смята сметката първо прибавя цената на всички закупени храни и после изважда промоциите, за които има необходимите продукти. Гошо установи, че системата не проверява дали някой продукт вече е участвал в промоция и само с една кола например може да се хванат няколко промоции наведнъж (пица с кола, бургер с кола и т.н.)

Имайки всички продукти и промоциите за комбинации от тях, Гошо иска да му помогнете (като напишете програма **food**) да избере тези храни и напитки, за които не само няма да има нужда да плаща, но и да вземе максимално "ресто", за да се върне в игралния клуб.

Не забравяйте, че Гошо няма никакви пари, така че ако не може да поръча нещо без да плати, той няма да си купи нищо и ще се прибере вкъщи.

Входните данни се четат от стандартния вход. На първия ред има две числа N ($3 \leq N \leq 75$) и M ($1 \leq M \leq 75$), съответстващи на броя различни храни и броя промоции във веригата за бързо хранене. Храните и промоциите са номерирани от 1 до техния съответен брой. Следват N реда, съдържащи по едно цяло число C_i ($1 \leq C_i \leq 1000000$) – цената на съответната храна. Следват M реда започващи с числата P_i ($1 \leq P_i \leq 1000000$) и F_i ($1 \leq F_i \leq 10$) – отстъпката за съответната промоция както и броят храни, които участват в нея. Всеки от M -те реда завършва с F_i различни числа – от 1 до N – храните в съответната промоцията.

Вашата програма трябва да изведе на стандартния изход едно число – максимално ресто, което Гошо може да получи при подходящо подбрана поръчка. Ако няма храни, такива, че закупувайки ги не се налага да се плаща – изведете 0.