N дами

Дадени са ви N на брой дами. Задачата ви е да ги разположите върху шахматна дъска с размери N X N, така че никоя дама да не дава шах на нито една от другите.

Жокер: Още една класическа задача, която се решава по-лесно с рекурсия.
Най-наивното решение би било да имаме двумерен масив представляващ шахматната дъска, който да показва всяко поле от колко дами се бие. В началото ще го инициализираме с нули. Ще извикаме рекурсивна функция, която въртейки два цикъла да постави текущата дама Ni (равно на нивото на рекурсията) на първото срещнато квадратче, което не се бие от вече поставените дами (се бие от нула дами), и след това да увеличим броят на ударите с едно във всички полета, които текущо сложената дама бие. След това извикваме рекурсивно нашата функция с номер на текущата дама+1. След като се върнем от рекурсията демаркираме всички полета, които тя е биела намаляйки стойностите им с едно. ПРОБЛЕМ 1: Рано ще получим комбинаторен взрив търсейки поле за поредната дама с два вложени цикъла до N. Така порядъкът на сложност на алгоритъма би бил O((N^2)^N) = O(N^(2*N)) без да броя маркиранията на битите полета при слагането на всяка текуща дама на текущо небито поле. ПРОБЛЕМ 2: Като взема предвид и маркиранията на битите полета порядъкът се умножава с още 4*N и става O(4*N*N^(2*N)) = O(4*N^(2*N+1)) = O(N^(2*N+1)), тъй като константите не важат, когато не са в степента. Този проблем не е толкова сериозен като първия тъй, като добавя само една единичка в степента. РЕШЕНИЕ НА ПРОБЛЕМ 1: Когато сложим дама на даден ред, то тя бие целият ред и няма смисъл да пробваме да сложим друга дама на него - следователно на всеки ред има не повече от една дама. Ако пък оставим един ред празен то ще трябва да наместим N дами на N-1 реда, от което следва че на поне един от тези N-1 реда ще има две дами, които ще си дават шах. Така стигаме до притиворечие с условието - следователно не може да има ред без дама. От следствията "всеки ред има не повече от една дама" и "не може да има ред без дама" на свой ред следва, че на всеки ред има точно една дама. Това следствие от своя страна води до оптимизация на алгоритъма ни - Ni-тата дама ще е на Ni-тия ред и ще ни трябва замо един цикъл до N (всички възможни колони) за да и намерим текуща позиция. Така свеждаме порядъка на сложност на нашия алгоритъм до O(N^(N+1)) - отпада "2*" в степента. РЕШЕНИЕ НА ПРОБЛЕМ 2: Нужно ли ни е да ползуваме двумерна матрица за да пазим текущо битите полета от текущо сложените дами? НЕ, РАЗБИРА СЕ. Вместо него можем да използуваме три глобални едномерни масива BeatenColumns, BeatenSlashDiagonals и BeatenBackSlashDiagonals.:
1) bool BeatenColumns[N] - Ще ни казва, кои колони се бият вече от текущо разположените дами на първите Ni реда (нива на рекурсията). Дамите на следващите N-Ni реда (нива на рекурсия) ще се разполагат само на свободните според този масив колони (BeatenColumns[i] == false). Да обобщим - когато поставяме текущата дама Ni на поле (Xi=Ni-1, Yi), ако то е свободно (според BeatenColumns, BeatenSlashDiagonals и BeatenBackSlashDiagonals), въртейки for Yi= 0..(N-1), маркираме че тя вече бие колоната на нейното поле така: BeatenColumns[Yi] = true. Маркираме и диагоналите които тя бие: BeatenSlashDiagonals[...] = true и BeatenBackSlashDiagonals[...] = true. След това викаме рекурсията QueensRecursive(Ni+1). След връщане от рекурсия освобождаваме колоната заета от текущата дама (BeatenColumns[Yi]=false), както и диагоналите които е биела, тъй като вече сме анализирали всички случаи при които тя се намира на тази колона и ще я пробваме на следващите Yi колони.
2) bool BeatenSlashDiagonals[2*N-1] - Ще ни казва, кои / диагонали са заети. Колко са те на брой. За стандартна дъска (N=8) те са 7 (N-1) над главния, главния и седем под него = 15 (2*N-1). Може да пробвате и за други N и ще се убедите че те са винаги (2*N-1). Но как да ги идентифицираме числово. Всички те са {
{(0, N-1)}, Забележете че Xi-Yi = 0 - (N-1) = 1 - N
{(0, N-2), (1, N-1)}, Забележете че Xi-Yi = 0 - (N-2) = 1 - (N-1) = 2 - N
{(0, N-3), (1, N-2), (2, N-1)}, Забележете че Xi-Yi = 0 - (N-3) = 1 - (N-2) = 2 - (N-1) = 3 - N

... ,
{(N-2, 0), (N-1, 1)}, Забележете че Xi-Yi = (N-2) - 0 = (N-1) - 1 = N-2
{(N-1, 0))}Забележете че Xi-Yi = (N-1) - 0 = N-1
}
Сигурно забелязвате че разликите между X и Y координатите на всички полета принадлежащи на един и същи диагонал е равна на едно и също число в интервала [1-N, N-1]. Следователно тази разлика можем да ползуваме за идентификация на диагонала, като индекс в масива BeatenSlashDiagonals. Тук имаме незначителен проблем: индекса може да стане отрицателен. Той се решава лесно като към него прибавим N-1 (защото 1-N е най-малката отрицателна стойност). Да обобщим - когато поставяме текущата дама Ni на поле (Xi=Ni-1, Yi) маркираме че тя вече бие и / диагонала (освен BeatenColumns[Yi] и BeatenBackSlashDiagonals[...]) на нейното поле така: BeatenSlashDiagonals[Ni-1-Yi+N-1] = true. След това викаме рекурсията QueensRecursive(Ni+1). Като се върнем освобождаваме диагонала BeatenSlashDiagonals[Ni-1-Yi+N-1] = false (освен BeatenColumns[Yi] и BeatenBackSlashDiagonals[...]) и продължаваме цикъла по Yi.
3) bool BeatenBackSlashDiagonals[2*N-1] - Ще ни казва, кои \ диагонали са заети. Тук положението е абсолютно аналогично, както при BeatenSlashDiagonals. За вас остава да намерите каква е функцията F(Xi,Yi)=const за всички полета лежащи на един и същ обратен диагонал. След като я откриете ще индексирате в BeatenBackSlashDiagonals така: BeatenBackSlashDiagonals[F(Xi,Yi)]=true/false. F(Xi,Yi) няма да дава отрицателни индекси и няма да прибавяме (N-1) към нея.
Условието за край на рекурсията е Ni > N. Тогава печатаме всички двойки координати на поставените дами. X координатата на всяка двойка е поредното число от 0 до N-1, а Yi координатата записваме по време на рекурсията и вземаме от помощен масив по подобие на масива Word В задачите "Пълно Комбиниране" и "Комбинации C(K,N)". След като разпечатаме първата валидна комбинация нямаме повече работа и вдигаме глобален флаг exit = true и го проверяваме в рекурсията (след като се върнем от по-долно ниво) за да можем да излезем аварийно и нашата програма да не се провали по "Time Out". Втори вариант за аварийно излизане е да хвърлим изключение и да го хванем на най-горно ниво при първото извикване на функцията QueensRecursive(1).
Първото извикване на рек. функция е QueensRecursive(1). Не забравяйте да нинициализирате 3-те помощни глобални масива с false.
РЕШЕНИЕТО НА ПРОБЛЕМ 2 сваля сложността на алгоритъма по време от O(N^(N+1)) на O(N^N). Не е много, но все пак си е полезна гимнастика за ума и може да ни помогне да влезем по време, ако тестовете са критични и/или програмираме на бавно изпълними езици, като JAVA, C#, JavaScript или Python.

Input Format

Дадено е само едно цяло число N представляващо броя на дамите и размера на дъската N X N едновременно.

Constraints

5 <= N <= 20

Output Format

Изпечатайте координатите на дамите - по една двойка координати X и Y на ред (общо N реда) за всяка дама - така че никои две дами да не си дават шах. От всички възможни такива позиции, трябва да разпечате тази при която поредната изпечатана координата (започвайки от X-а на първата дама, Y-ка на първата дама, X-а на втората и т.н до Y-ка на последната) е най-малката възможна.

Sample Input 0

```
4
```

Sample Output 0

```
1 2
2 4
3 1
4 3
```