NqProblemExtended application.

Description : This program was born to play around the N Queens Problem.
We give a solution which considers the problem from a whole
different aspect and splits the actual calculation from the
real rules of the chess game. In this way, the calculations
of rooks and bishops are also available as well as their
super or awesome versions.
(super means that the pieces can also attack as a knight and)
(awesome means the above but till the edge of the chessborad)
Preplaced chess pieces are also available to solve N Queens
Completion problem.

Published : 01.01.2018

Current version : 1.0

Developed by : Jozsef Kiss
<thehobbypianist@gmail.com>

Changelog : 1.0 - 01.01.2018
Initial release.

A java alkalmazas tervezese es implementalasa elott es soran szandekosan nem tortent kutatasi munka, nehogy masok gondolatai megvezessenek.
A cel egy egyedi megkozelites es mukodo megoldas kidolgozasa volt!
A megkozelites a hagyomanyos volt: ahhoz hogy megszamolhassuk az adott N meretu tablahoz tartozo lerakasokat, le kell tenni az osszes lehetseges modon a sakkfigurakat es azokat megszamoljuk (found++).
Nem volt cel az hogy meglassunk elemi vagy osszetettebb szabalyszerusegeket az egyes lerakasok kozott, hogy ezaltal egy-egy tovabbi lerakast "megsporoljunk" kevesebb vagy tobb processzoridot megtakaritva ezzel. Ebbol kovetkezik hogy minden egyes found++ eseten azonnal, tovabbi kalkulaciok nelkul kiirathato az aktualis sikeres sakkfigura lerakas.

Az alapvetes az volt, hogy ha letezik a hagyomanyos rekurziv-visszalepeses technikanal gyorsabb keresesi megoldas az osszes lerakasok megszamolasara, akkor az nem lehet tetszoleges bonyolultsagu, kulonben az osszetettseggel a program futasi ideje szuksegszeruen novekedne.
Igy a megoldas veges de inkabb "rovid" idon belul megtalalhato kell legyen.

Az elsodleges celkituzesek az alabbiak voltak:
1. n=15 meretu tablan az osszes lerakas megtalalasara a szoftver futasi ideje felezodjon a hagyomanyos modszerhez kepest, amely 1:44s -> 52s alatti legyen! (a hagyomanyos modszer is implementalasra kerul az ugyanazon szamitogepen torteno osszehasonlithatosag miatt)
2. minden kiralynot letenni, nincs csalas! :)
3. ha lehet, olyan megoldast talalni amely szetvalasztja a sakk szabalyait es a tenyleges pozicio keresest
4. elemi adattipusokat hasznalni a mas nyelvekre torteno konnyebb portolas erdekeben
5. a projekt ideje maximum 1 ember honap

Alabb lathato hogy a fontiek mindegyike megvalositasra kerult:
1. a hagyomanyos futas (~1:44s) idejenek hatoda alatt megkeresheto a 15 kiralyno (~17s)
2. minden kiralyno lerakasra kerul
3. a szabalyok es a tenyleges kereses logikajanak szetvalasztasaval konnyeden keresheto barmi a tablan, be is vezettunk uj sakkfigurakat a moka kedveert. tovabba keresheto pl. futo is, amely lerakasait sokkal komplikaltabban lehetne csak keresni a hagyomanyos modon.
4. int-ek, String-ek, boolean-ok, tombok lesznek hasznalva a konzolra iras es a rendszerido lekerdezesere a java.lang.System csomag hasznalatos, am ezek konnyeden cserelhetok
5. teljesult, bar fo munkaidon kivuli tevekenysegkent negyed evig elhuzodott a brutto es osszes tervezesi, implementalasi es tesztelesi tevekenyseg
+1. logika szuletett az osszes (az is szamit hogy melyik figurat tettuk le elobb) es a rendezett (csak a novekvo megoldasokat vesszuk) lerakasok megtalalasara,

   minimalis atalakitassal
+2. a szoftver 9 az 1-ben megoldast ad, 9 fele sakkfigura keresese lehetseges
+3. elore lerakott sakkfigurakat is meg lehet adni igy az N Kiralyno (Figura)
   Kiegeszites problema is megoldhato.

Szinte mindenki a kiralynok lerakasara fokuszalt, mert ez a babu az egesz tablat bejarhatja, es jo sok iranyba tud utni.
Ha az 1 lepesre kepes figurakat: gyalog es kiraly nem szamitjuk, vannak meg tovabbi sakkfigurak amelyek az egesz tablan tudnak tamadni, osszesen:

- vezer
- bastya
- futo

Ez a 3 sakkfigura kepes tehat a sakk szabalyai szerint a sakktabla szeleig tamadni. A lo egyelore ugy tunik kimaradt.
Ismeretes viszont a szuperkiralyno fogalma (super). Nevezetesen, olyan vezer, amely tamadhat a sakk hagyomanyos szabalyai szerint, plussz meg L alakban is. Ennek analogiajara bevezetheto ez a tulajdonsag a tobbi altalunk hasznalt figurara is:

- szuper vezer
- szuper bastya
- szuper futo

Igy a lovat is jatekba tudtunk hozni. Viszont miert allnank meg itt?
Adodik a sakktabla szeleig tamadas kepessege, igy bevezetjuk az oruletes (awesome) jelzot a szuper jelzohoz hasonloan. Azzal a kulonbseggel, hogy a szuper sakkfigura 1 tetszoleges lo tavolsagra uthet, az oruletes az elso lo utes iranyaba es egeszen a sakktabla szeleig lepegetve.
Ilyen modon a kovetkezo sakkfigurak is lerakhatok:

- oruletes vezer
- oruletes bastya
- oruletes futo

Az egyes sakkfigurak lehetseges tamadasai tehat:

Queen:

```
+ + + + q + + + +
- - - + + + - - -
- - + - + - + - -
- + - - + - - + -
+ - - - + - - - +
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
```

Super queen:

```
+ + + + q + + + +
- - + + + + + - -
- - + + + + + - -
- + - - + - - + -
+ - - - + - - - +
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
```

Awesome queen:

```
+ + + + q + + + +
- - + + + + + - -
+ - + + + + + - +
- + - - + - - + -
+ - + - + - + - +
- - - - + - - - -
- + - - + - - + -
- - - - + - - - -
+ - - - + - - - +
```

Rook:

```
+ + + + r + + + +
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
```

Super rook:

```
+ + + + r + + + +
- - + - + - + - -
- - - + + + - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
- - - - + - - - -
```

Awesome rook:

```
+ + + + r + + + +
- - + - + - + - -
+ - - + + + - - +
- - - - + - - - -
- - + - + - + - -
- - - - + - - - -
- + - - + - - + -
- - - - + - - - -
+ - - - + - - - +
```

Bishop:

```
- - - - b - - - -
- - - + - + - - -
- - + - - - + - -
- + - - - - - + -
+ - - - - - - - +
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
```

Super bishop:

```
- - - - b - - - -
- - + + - + + - -
- - + + - + + - -
- + - - - - - + -
+ - - - - - - - +
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
```

Awesome bishop:

```
- - - - b - - - -
- - + + - + + - -
+ - + + - + + - +
- + - - - - - + -
+ - + - - - + - +
- - - - - - - - -
- + - - - - - + -
- - - - - - - - -
+ - - - - - - - +
```

A fonti megfontolasokat alkalmazva ezen 9 sakkfigura lerakasat vegzi el az alabbi program.

Az alkalmazott megoldas.

Alapotlet: gondolkodjunk elore es ne visszafele!
Kalkulaljuk ki azt, hogy a lerakas kovetkezteben melyek lesznek a meg
hasznalhato poziciok. Ezaltal azt erjuk el, hogy a kovetkezo lerakas az
biztosan egy helyes pozicioba fog tortenni. A sejtes az, hogy ez sokkal kevesebb
eroforrast fog elvinni. (Szemben a hagyomanyos megoldasokkal, amelyek letesznek egy
vezert valamilyen pozicioba, es nezik hogy azt tamadja-e valamelyik, mar letett figura)

A sakktablat egy rendezett sorozattal reprezentaljuk.
Pl.4x4:
0 1 2 3 -> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
4 5 6 7
8 9 10 11   Egy N dimenzios sakktabla reprezentacioja tehat egy
12 13 14 15   0-tol (N^2 - 1)-ig tarto rendezett sorozat.

Ha egy sakkfigurat leteszunk, akkor a kovetkezo tortenik.
1: adott egy bemeno sorozat amely lehet az eredeti vagy az eredetinek egy
   reszsorozata
2: ennek a sorozatnak kivalasztjuk egy elemet
3: a bemeno sorozat es a kivalasztott elem fuggvenyeben eloallitjuk a bemeno
   sorozat meghatarozott, rendezett reszsorozatat, amely az elokalkulalt
   tamadasi terkepnek megfelel

Ezzel a modszerrel garantalni tudjuk azt hogy a kovetkezo sakkfigura lerakas
biztosan egy helyes pozicioba tortenjen.

Ebbol kovetkezik, hogy az osszes letezo megoldas meg lesz keresve, beleertve meg a
lerakas sorrendjet is. Ha az osszes lehetoseg kozul mi csak egyet szeretnenk venni,
akkor ki kell valasztanunk egyet. Ez lesz az adott lerakasokhoz tartozo rendezett
megoldas. Peldaul. (4x4): megtalaljuk majd az
1 7 8 14, 1 7 14 8, 1 8 7 14, 1 8 14 7, 1 14 7 8, 1 14 8 7 ...24 * 2 = 48 megoldast.
Ha csak egyet-egyet szeretnenk megtalalni ezek kozul, akkor csak az 1 7 8 14 es a 2 4 11
13 megoldasok lesznek megtalalva. Ehhez az elokalkulalt tamadasi terkepet modositani
kell. Ez eleg lenne, de egy optimalizalo megoldast is alkalmazunk hogy felgyorsitsuk a
keresest. (A megfelelo poziciokat talalnank meg pusztan a tamadasi terkepet hasznalva,
de tul sok es elkerulheto halott ag lesz.)

Opciok.

```
  Futtatasi mod: [ o , i , t ]
    o: original, hagyomanyos backtrack-rekurziot hasznalo megoldas
       valamivel gyorsabb lesz mert a tamadasi terkepet hasznalja,
       viszont a hagyomanyos backtrack-rekurzios megkozelitesben
    i: improved, a fejlesztett, mind a 9 fontebbi sakkfigurat letenni kepes
       algoritmus implementacioja
    t: testing, azaz az improved es az original megoldasok osszehasonlitasara
  Dimenzio: [ int ]
    ekkora meretu legyen a sakktabla es pontosan ennyi darab sakkfigurat
    szeretnenk letenni a tablara
  ( a tovabbiak csak az improved futasi modban hasznalhatok )
  Sakkfigura: [ q , r , b ]
    q: queen
    r: rook
    b: bishop
  Figura tipus: [ r , s , a ]
    r: regular, sakkfigura hagyomanyos tamadasi kepessegekkel
    s: super, hagyomanyos tamadasi mod + lo alakban is
    a: awesome, hagyomanyos tamadasi mod + lo alakban is de a tabla szeleig
  Talalatok: [ o , a , f ]
    o: ordered, a rendezett megoldasokat keresi
    a: all, az osszes megoldast keresi, igy a lerakasok sorrendje is
       figyelembe lesz veve:
       all solutions == dimenzio! * ordered solutions
    f: first, csak az elso helyes lerakas lesz megkeresve
  Kulonbozok: [ y , n ]
    y: csak a tukrozessel egymasba nem viheto megoldasok szamitanak
       ( lassu mert effort levizsgalni hogy elozoleg mar megtalaltuk-e
         az adott lerakas elforgatottjat )
```

```
    n: minden megoldas szamit, tukrozest nem vizsgalunk a lerakasokra
  Log: [ n , i , d ]
    n: nincs logolas
    i: info, a legelso sorba lerakott kiralynok szerint kiirja a talalatok szamat
       es itt a keresessel eltoltott idot
    d: debug, minden informacio kiirasra kerul a konzolra a kereses soran
  Lerakas: [ int ertekek , karakterrel elvalasztva ]
    olyan sakkfigurak amelyeket elore le kell tenni, uresen hagyhato
```

A sok opcio miatt a core algoritmus sok helyre lesz duplikalva, a sok opcio okozza a kod hosszusagat.

A core algoritmus az i futasi modra:

0. isFiltered 2 dimenzios tomb elokalkulacioja a megadott szabalyrendszer alapjan:
   - dimenzio (n)
   - milyen sakkfigurakat hasznalunk (q , r , b)
   - ezek mely valtozatat (r , s , a)
   - talalatok (o , a , f)
1. filterezo fuggveny amely megadja azt, hogy az eredeti sorozatot az adott elem hogyan filterezi (melyek maradnak a lerakas kovetkezteben tovabbra is szabad poziciok)
   applyFiltersXXX (2):

```
boolean [ ] a = isFiltered [ currPiecePos ] ;
for ( int i = from ; i < to ; i ++ )
{
  if ( ! a [ workingArray [ i ] ] )
  {
    workingArray [ currIndToWrite ] = workingArray [ i ] ;
    currIndToWrite ++ ;
  }
}
return currIndToWrite - to ;
```

2. rekurziv lerako metodus, amely az elozo filterezes kovetkezteben szabad poziciokra leteszi a kovetkezo figurat.
   putPiecesXXX (18):

```
if ( pieceToPlace < dimension )
{
  if ( to - from >= dimension - pieceToPlace )
  {
    int count ;
    for ( int i = from ; i < to ; i ++ )
    {
      currPath [ pieceToPlace ] = workingArray [ i ] ;
      currIndToWrite = to ;
      count = applyFilters ( currPath [ pieceToPlace ] , from , to ) ;
      putPiecesXXX ( pieceToPlace + 1 , currIndToWrite-count , currIndToWrite ) ;
    }
  }
  else
  {
    deads ++ ;
  }
}
else
{
  found ++ ;
}
```

A fonti megoldasnak 8-10 variansa kiprobalasra kerult (pl. egyszerre 2-t tesz le, nem az elso sorba teszi le a sakkfigurakat hanem a kozepsobe, stb.), de ezek egyike sem futott gyorsabban. Ugy talaltuk hogy az elvet megvalosito algoritmus fonti valtozatanal van a futasi idonek minimuma.

A fonti modszer futasi idejeire legjobban illesztheto gorbek:

```
D  elapsed (ms)
13 536
14 2941
```

15 17580
16 117029
17 820977
18 6081743

5th polynomial:
y = -17692570000 + 5994815000*x - 811125900*x^2 + 54785620*x^3 - 1847318*x^4 + 24879.31*x^5

exponential:
y = 0.289471 + 1.398972e-9*e^(+2.000462*x)

A program hasznalatanak eredmenyei.

Nehany esetet kiszamitottunk es az eredmenyek alabb olvashatok.

A hasznalt konfiguracio ez volt
 - Windows 10 x64 OS
 - Intel Celeron N2840 (2 logical CPU cores, 2.16GHz)
 - 8GB 1333MHz DDR3

Az oruletes kiralynok kalkulacioja 28 esetre mas volt, masik szamitogepet hasznaltunk.
Annak parameterei ott olvashatok.

////////// **T E S T I N G M O D E** //////////

Ez a teszt (t) futasi mod a fejlesztett (i) es az eredeti (o) megoldas
osszevetesere szuletett.

A parancs ez volt:
  java -jar NqProblemExtended.jar t 18

mode o -> mode i (improved version versus original solution)
mode o won't be run,
  the times have to be written manually to the source
rate1: i elapsed / o elapsed
rate2: i (k) elapsed / i (k-1) elapsed

```
n   count      elapsed      elapsed2             -> elapsed        elapsed2           count        rate1   rate2
13  c73712     2377     ms ( 2s 377ms )          -> 536       ms ( 536ms )          c73712     ( 0.225 |       )
14  c365596    15116    ms ( 15s 116ms )         -> 2941      ms ( 2s 941ms )       c365596    ( 0.194 | 5.486 )
15  c2279184   104357   ms ( 1m 44s 357ms )      -> 17580     ms ( 17s 580ms )      c2279184   ( 0.168 | 5.977 )
16  c14772512  766915   ms ( 12m 46s 915ms )     -> 117029    ms ( 1m 57s 29ms )    c14772512  ( 0.152 | 6.656 )
17  c95815104  6040290  ms ( 1h 40m 40s 290ms )  -> 820977    ms ( 13m 40s 977ms )  c95815104  ( 0.135 | 7.015 )
18  c666090624 48165728 ms ( 13h 22m 45s 728ms ) -> 6081743   ms ( 1h 41m 21s 743ms ) c666090624 ( 0.126 | 7.407 )
```

////////// **A W E S O M E Q E E N S O N 4 0 T H R E A D S** //////////

Ez a szamitas egy masik laptopon futott, amelynek parameterei:
 - Dell 3521
 - Intel i3-3227 (4 logical CPU cores, 1.9GHz)
 - 2x4 GB 1333MHz DDR3

A thread pool 40 szalon volt hasznalva mert ugy talaltuk hogy a futasi idok igy a
legkisebbek.

Az utolso szamitas parancsa ez volt:
  java -jar NqProblemExtended.jar i 28 q a o 40 n i

dimension	solutions	elapsed
1	1	47ms
2 -> 9	0	47 -> 64ms
10	4	78ms
11	33	78ms
12	6	78ms
13	59	79ms
14	8	94ms
15	12	109ms
16	18	187ms
17	180	250ms
18	124	594ms
19	361	1s 687ms
20	516	6s 860ms
21	689	25s 728ms
22	2092	2m 0s 498ms
23	5639	8m 32s 988ms
24	22794	41m 52s 252ms
25	68044	3h 9m 59s 430ms
26	275732	15h 45m 42s 275ms
27	767820	3d 3h 3m 47s 476ms

```
  | 28        |   3698242 | 17d 10h 58m 14s 423ms |
  +-----------+-----------+-----------------------+
  oruletes kiralynok 40 szalon

//////////     D E B U G   M O D E   O F   4   Q U E E N S       //////////

Nezzunk egy peldat a debug modra!
Ez az eset megmutatja a core logikat, a bemeno es kimeno sorozatokat mutatja az eppen
lerakott sakkbabu es az elokalkulalt tamadasi terkep fuggvenyeben.

Parancs:
  java -jar NqProblemExtended.jar i 4 q r o 1 n d
  [ ]
                          | ----------+-----------------------------------
                          | to filter : [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8
                          |             , 9 , 10 , 11 , 12 , 13 , 14 , 15 ]
                          | Piece     : 0
                          | filtered  : [ 6 , 7 , 9 , 11 , 13 , 14 ]
  [ 0   ]
                          | ----------+-----------------------------------
                          | to filter : [ 6 , 7 , 9 , 11 , 13 , 14 ]
                          | Piece     : 6
                          | filtered  : [ 13 ]
  [ 0   6   ]
        0
                          | ----------+-----------------------------------
                          | to filter : [ 6 , 7 , 9 , 11 , 13 , 14 ]
                          | Piece     : 7
                          | filtered  : [ 9 , 14 ]
  [ 0   7   ]
                          | ----------+-----------------------------------
                          | to filter : [ 9 , 14 ]
                          | Piece     : 9
                          | filtered  : [ ]
  [ 0   7   9   ]
          0
        0
    0
                          | ----------+-----------------------------------
                          | to filter : [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8
                          |             , 9 , 10 , 11 , 12 , 13 , 14 , 15 ]
                          | Piece     : 1
                          | filtered  : [ 7 , 8 , 10 , 12 , 14 , 15 ]
  [ 1   ]
                          | ----------+-----------------------------------
                          | to filter : [ 7 , 8 , 10 , 12 , 14 , 15 ]
                          | Piece     : 7
                          | filtered  : [ 8 , 12 , 14 ]
  [ 1   7   ]
                          | ----------+-----------------------------------
                          | to filter : [ 8 , 12 , 14 ]
                          | Piece     : 8
                          | filtered  : [ 14 ]
  [ 1   7   8   ]
                          | ----------+-----------------------------------
                          | to filter : [ 14 ]
                          | Piece     : 14
                          | filtered  : [ ]
  [ 1   7   8   14  ]

  | 1 | 7 | 8 | 14 |
  * q * *
  * * * q
  q * * *
  * * q *
              1
          1
        1
    1
                          | ----------+-----------------------------------
                          | to filter : [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8
                          |             , 9 , 10 , 11 , 12 , 13 , 14 , 15 ]
                          | Piece     : 2
                          | filtered  : [ 4 , 9 , 11 , 12 , 13 , 15 ]
  [ 2   ]
                          | ----------+-----------------------------------
                          | to filter : [ 4 , 9 , 11 , 12 , 13 , 15 ]
                          | Piece     : 4
                          | filtered  : [ 11 , 13 , 15 ]
  [ 2   4   ]
                          | ----------+-----------------------------------
                          | to filter : [ 11 , 13 , 15 ]
                          | Piece     : 11
                          | filtered  : [ 13 ]
  [ 2   4   11  ]
                          | ----------+-----------------------------------
                          | to filter : [ 13 ]
                          | Piece     : 13
                          | filtered  : [ ]
  [ 2   4   11  13  ]

  | 2 | 4 | 11 | 13 |
  * * q *
  q * * *
  * * * q
  * q * *
              1
          1
```

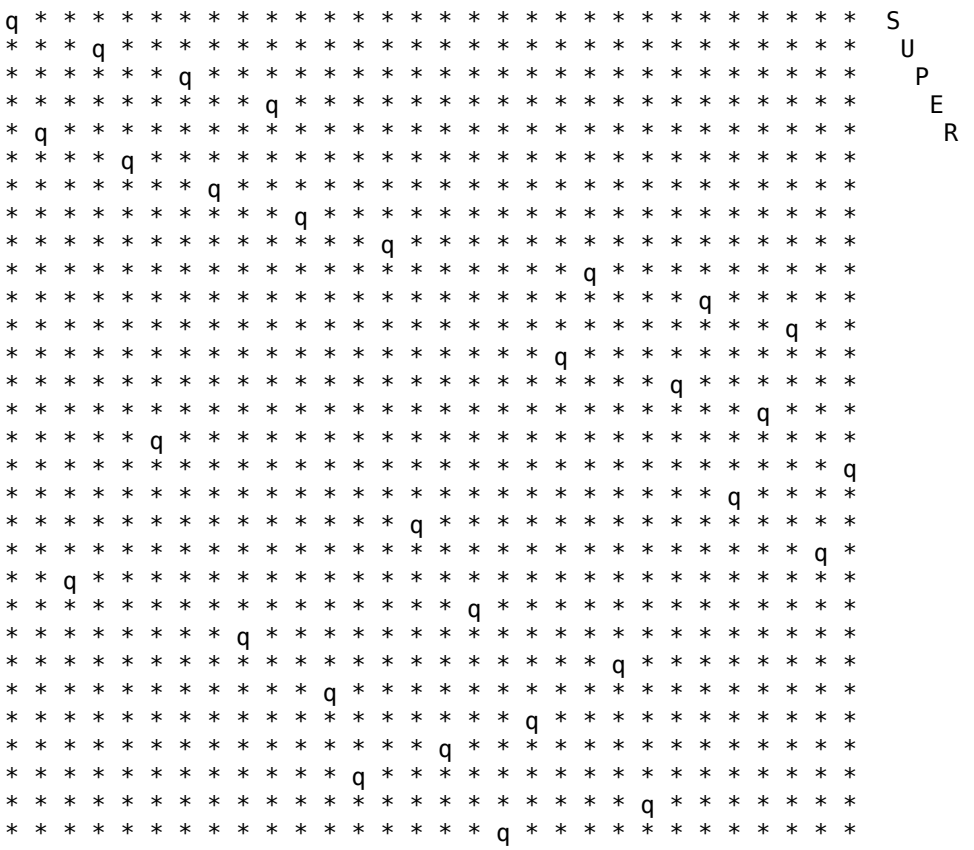
```
      1
    1
                          | ----------+-----------------------------------
                          | to filter : [ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8
                          |             , 9 , 10 , 11 , 12 , 13 , 14 , 15 ]
                          | Piece     : 3
                          | filtered  : [ 4 , 5 , 8 , 10 , 13 , 14 ]
  [ 3   ]
                          | ----------+-----------------------------------
                          | to filter : [ 4 , 5 , 8 , 10 , 13 , 14 ]
                          | Piece     : 4
                          | filtered  : [ 10 , 13 ]
  [ 3   4   ]
                          | ----------+-----------------------------------
                          | to filter : [ 10 , 13 ]
                          | Piece     : 10
                          | filtered  : [ ]
  [ 3   4   10  ]
          0
      0
                          | ----------+-----------------------------------
                          | to filter : [ 4 , 5 , 8 , 10 , 13 , 14 ]
                          | Piece     : 5
                          | filtered  : [ 14 ]
  [ 3   5   ]
      0
    0
2

  2
  positions have been found
  all attempts: 6
             ( 33.33% success )
  in 279ms
   ( 279 )
////////// F I R S T   Q U E E N S   O N   3 0   D I M   B O A R D ////////
```

| 0 | 32 | 64 | 91 | 123 | 158 | 190 | 222 | 254 | 276 | 322 | 355 | 387 | 414
| 441 | 473 | 509 | 536 | 568 | 585 | 611 | 639 | 667 | 695 | 737 | 769 | 796
| 823 | 860 | 888 |

```
q * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  R
* * q * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   E
* * * * q * * * * * * * * * * * * * * * * * * * * * * * * * * *    G
* q * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     U
* * * q * * * * * * * * * * * * * * * * * * * * * * * * * * * *      L
* * * * * * * * q * * * * * * * * * * * * * * * * * * * * * * *       A
* * * * * * * * * * q * * * * * * * * * * * * * * * * * * * * *        R
* * * * * * * * * * * * q * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * q * * * * * * * * * * * * * * * * *
* * * * * * q * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * q * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * q * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * q * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * q * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * q * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * q * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * q * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * q * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * q * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * q * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * q * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * q * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * q * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * q * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * q * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * q * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * q * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * q * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * q * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * q * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * q * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * q * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * q * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * q * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * q * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * q * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* q * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

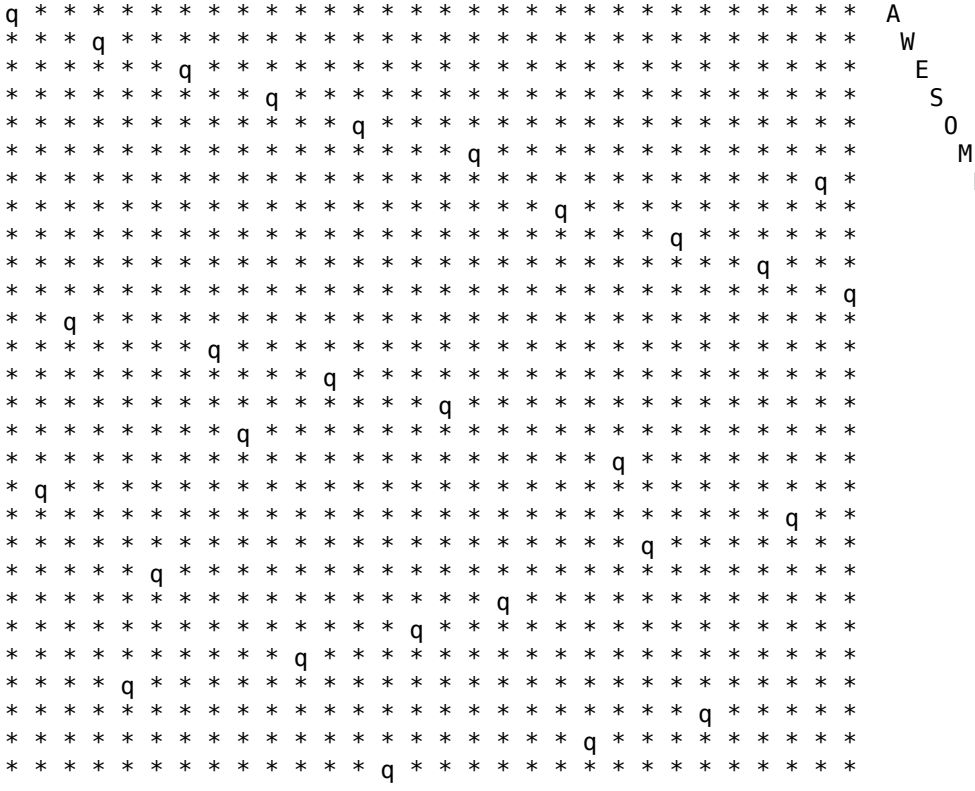
```
  1
  position has been found,
  all attempts: 20256683
              ( 0.0% success )
  in 12s 542ms
   ( 12542 )
```

| 0 | 33 | 66 | 99 | 121 | 154 | 187 | 220 | 253 | 290 | 324 | 357 | 379 | 413
| 446 | 455 | 509 | 535 | 554 | 598 | 602 | 646 | 668 | 711 | 731 | 768 | 795
| 822 | 862 | 887 |



1
position has been found,
all attempts: 16589947
(0.0% success)
in 11s 999ms
(11999)

| 0 | 33 | 66 | 99 | 132 | 166 | 208 | 229 | 263 | 296 | 329 | 332 | 367 | 401
| 435 | 458 | 501 | 511 | 567 | 592 | 605 | 647 | 674 | 700 | 724 | 774 | 800
| 823 | 865 | 888 |



```
 * * * * * * * * * * * * * * * * * * * * * * * * * q * * * *
 * * * * * * * * * * * * * * * * * * * * q * * * * * * * * * *

  1
  position has been found,
  all attempts: 64652148
              ( 0.0% success )
  in 50s 572ms
   ( 50572 )
```

////////// T H E A T T A C K I N G M A P //////////

A tamadasi lehetosegek vizualizacioja.
Ez a tablazat reprezentalja mindegyik babut mindegyik babuval. A kozepen talalhato csokornyakkendok szama: dimenzio − 1. r || b == q.
“+” : a babuk amelyek az adott sorban es oszlopban talalhatok, utik egymast
“ “ : nem utik egymast.
Pelda parancs hogy ezt lathassuk: java -jar NqProblemExtended.jar i 5 q r f 1 n n

```
   0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
0  +  +  +  +  +  +  +           +     +        +        +     +           +
1  +  +  +  +  +  +  +  +           +     +        +        +     +
2  +  +  +  +  +     +  +  +     +     +     +        +              +
3  +  +  +  +  +        +  +  +     +     +     +        +              +
4  +  +  +  +  +           +  +        +     +     +        +  +           +
5  +  +           +  +  +  +  +  +  +           +     +        +        +
6  +  +  +        +  +  +  +  +  +  +  +           +     +        +        +
7     +  +  +     +  +  +  +  +     +  +  +     +     +     +        +
8        +  +  +  +  +  +  +  +        +  +  +     +     +     +        +
9           +  +  +  +  +  +  +           +  +        +     +     +        +
10 +     +        +  +           +  +  +  +  +  +  +           +     +
11    +     +     +  +  +        +  +  +  +  +  +  +  +           +     +
12 +     +     +     +  +  +     +  +  +  +  +     +  +  +     +     +     +
13    +     +           +  +  +  +  +  +  +  +        +  +  +     +     +
14       +     +           +  +  +  +  +  +  +           +  +        +     +
15 +        +     +     +        +  +           +  +  +  +  +  +  +
16    +        +     +     +     +  +  +        +  +  +  +  +  +  +  +
17       +        +     +     +     +  +  +     +  +  +  +  +     +  +  +
18 +        +        +     +           +  +  +  +  +  +  +  +        +  +  +
19    +        +        +     +           +  +  +  +  +  +  +           +  +
20 +           +  +        +     +     +        +  +           +  +  +  +  +
21    +              +        +     +     +     +  +  +        +  +  +  +  +
22       +              +        +     +     +     +  +  +     +  +  +  +  +
23          +     +        +        +     +           +  +  +  +  +  +  +  +
24 +           +     +        +        +     +           +  +  +  +  +  +  +
```

///// O R D E R E D A N D N O T U N I Q U E S O L U T I O N S /////

Ime nehany tovabbi pelda kalkulacio.

ordered and not unique (by mirroring) queen solutions

dimension	regular	super	awesome
1	1	1	1
2	0	0	0
3	0	0	0
4	2	0	0
5	10	0	0
6	4	0	0
7	40	0	0
8	92	0	0
9	352	0	0
10	724	4	4
11	2680	44	33
12	14200	156	6
13	73712	1876	59
14	365596	5180	8
15	2279184	32516	12

ordered and not unique (by mirroring) rook solutions

dimension	regular	super	awesome
1	1	1	1
2	2	2	2
3	6	2	1
4	24	8	8
5	120	20	10
6	720	94	22
7	5040	438	38
8	40320	2766	276
9	362880	19480	475
10	3628800	163058	2304
11	39916800	1546726	4884
12	479001600	16598282	24528

ordered and not unique (by mirroring) bishop solutions

dimension	regular	super	awesome
1	1	1	1
2	4	4	4
3	26	6	6
4	260	86	86
5	3368	854	293
6	53744	9556	2824
7	1022320	146168	12098
8	22522960	2660326	234450
9	565532992	56083228	1465563

//////// A L L A N D N O T U N I Q U E S O L U T I O N S ////////

Nem egyedi megoldas alatt itt azt ertjuk hogy a lerakas sorrendje szamit.

all and not unique (by mirroring) queen solutions

dimension	solutions	elapsed
1	1 (== 1 * 1!)	32ms
2	0 (== 0 * 2!)	47ms
3	0 (== 0 * 3!)	40ms
4	48 (== 2 * 4!)	47ms
5	1200 (== 10 * 5!)	31ms
6	2880 (== 4 * 6!)	62ms
7	201600 (== 40 * 7!)	125ms
8	3709440 (== 92 * 8!)	1s 884ms
9	127733760 (== 352 * 9!)	1m 3s 108ms

```
////////    F I R S T   T O   P R E P L A C E D   B I H S H O P S    ////////

Let's  calculate the positions of the chess
pieces  when they don't  attack each other!

  0 mode      (original,improved,testing) : i
  1 dimension (a positive integer)        : 8
  2 pieces    (queen,rook,bishop)         : b
  3 kinds     (regular,super,awesome)     : r
  4 hits      (ordered,all,first)         : f
  5 threads   (a positive integer)        : 1
  6 uniques   (no,yes)                    : n
  7 log       (no,info,debug)             : n
  8 placings  (ints separated by , char)  : 43,44,45,46

Started : Sun Dec 17 15:39:23 CET 2017

Attacking map  is too  large so it
could be printed under dimension 7

 Preplaced chess pieces are: 43 44 45 46

 ( case : improved 16 )

 | 43 | 44 | 45 | 46 | 2 | 3 | 4 | 5 |
 * * b b b b * *
 * * * * * * * *
 * * * * * * * *
 * * * * * * * *
 * * * * * * * *
 * * * B B B B *
 * * * * * * * *
 * * * * * * * *

  1
  position has been found,
  in 78ms
   ( 78 )
```

Tovabbi megfigyelesek a regular lerakasokra.

```
Rook kereses:
 - mindig 100%-os
 - minden elso poziciohoz ugyanannyi lerakas tartozik
 - helyes lerakasok szama: dimenzio!

Bishop kereses:
 - >70% talalati arany
 - olyan poziciok is lehetsegesek amelyek a tabla kozepen, vegen kezdodnek
 - teljesen valid megoldast ad a legelso vagy legutolso sor, oszlop telerakasa

Queen kereses:
 - par %-os talalati arany
 - n = 8 eseten a talalatok szama az egyes poziciokra:
      4   8   16  18  18  16  8   4
      8   16  14  8   8   14  16  8
      16  14  4   12  12  4   14  16
      18  8   12  8   8   12  8   18
      18  8   12  8   8   12  8   18
      16  14  4   12  12  4   14  16
      8   16  14  8   8   14  16  8
      4   8   16  18  18  16  8   4
      - barmelyik sorban vagy oszlopban levo szamok osszege:
        92 amely a helyes lerakasok szamat adja
      - szimmetrikus a megoldasok szama, nem kerult kihasznalasra
        ( Jk : n == 2k + 1 eseten ki kell szamolni a kozepvonalig a talalatokat,
          meg kell szorozni 2-vel es ehhez hozzaadni az elso sor kozepso elemere
          eso talalatokat
          egyeb esetre: ki kell szamolni a kozepvonalig a talalatokat,
```

meg kell szorozni 2-vel)

- a logikabol kovetkezik hogy minden pillanatban tudhato hogy mennyi szabad hely all meg rendelkezesre a sakkfigurak letetelere
  vezer eseten:
  - 0. figura lerakas elott: n^2 (minden pozicio szabad)
  - 1. figura lerakasa elott: n^2 - 3n + 2 (az elso sorba barhova is tegyuk)
  - a tovabbi helyeken: fugg a poziciotol es mennyi hely volt mar elozoleg kitakarva
  - az (n-1). figura lerakasa elott: pontosan 1 szabad hely van.

  egyelore nem kerult kihasznalasra.