Introduction to Discrete Math

Felipe P. Vista IV

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - Ex: 202054321 Juan Dela Cruz

Not changing your name to this format

* you **will** be marked Absent * → absent?

Course Outline

- Mathematical Thinking
  - Convincing Arguments, Find Example, Recursion, Logic, Invariants
- Probability & Combinatronics
  - Counting, Probability, Random Variables
- Graph Theory
  - Graphs (cycles, classes, parameters)
- Number Theory & Cryptography
  - Arithmetic in modular form
  - Intro to Cryptography

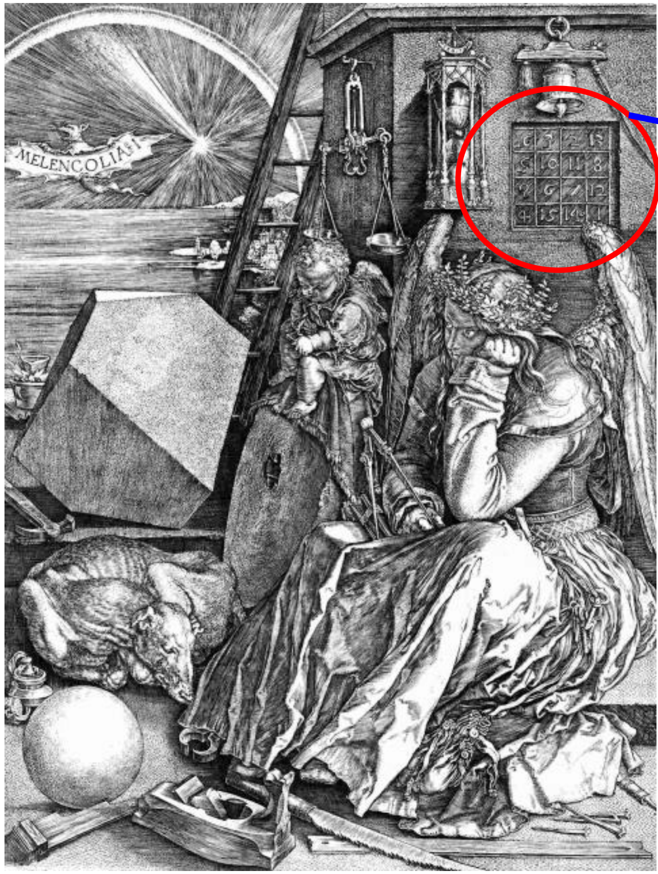
Mathematical Thinking –Find Example

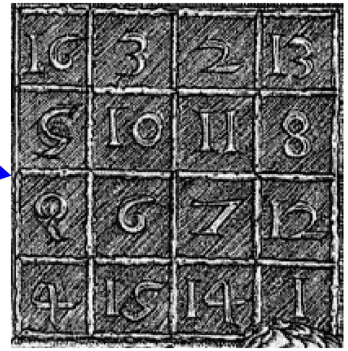
HOW TO FIND AN EXAMPLE

How To Find Example

- Magic Square
- Narrow Search
- Multiplicative Magic Squares
- Additional Puzzles
- Integer Linear Combinations
- Paths in a Graphs

Find ways!





magic square!

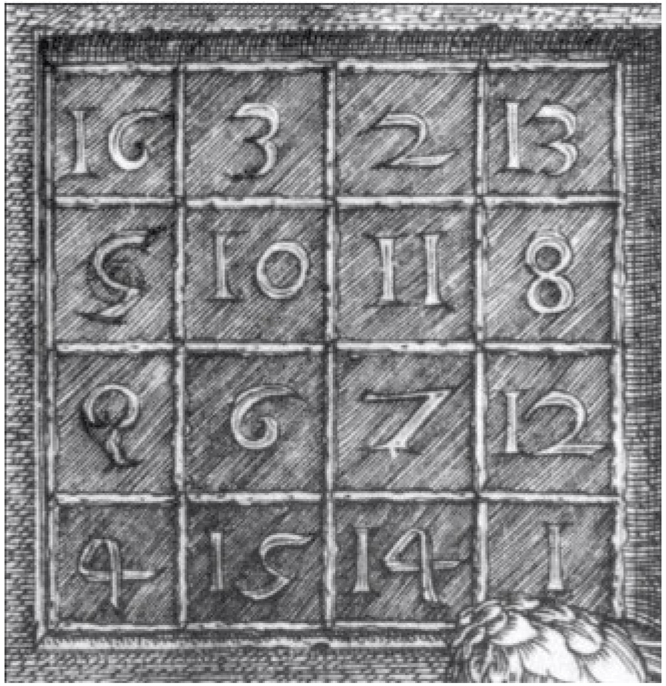
Melancholia (1514)
An engraving by
Albrecht Durer

* melancholia – very deep sadness, depression, withdrawal , apathy
* apathy – lack of interest, concern or enthusiasm

Magic Square

definition:

- a table with unique numbers whose sum for all 4 rows, 4 columns, and 2 diagonals (for a 4x4) are the same
- For a *n x n* square
  - 1, 2, 3, …., 15, 16
  - 1, 2, 3, …, n^2



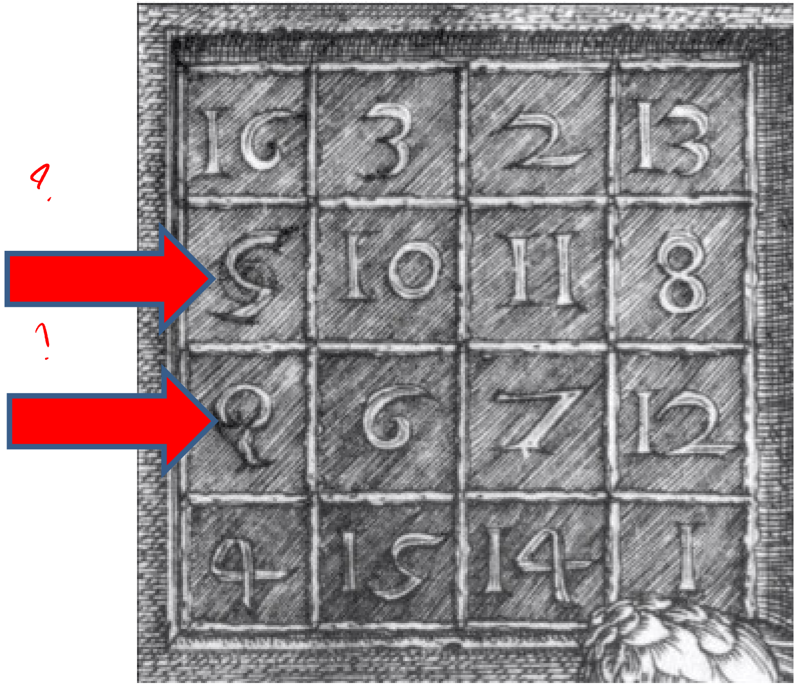
Magic Square

Taking a good look

- You can see some numbers are not clear, can you identify the numbers?
  - 2^{nd} row, 1^{st} col
    - It is 5
  - 3^{rd} row, 1^{st} col
    - It is 9

9.

?



Find Magic Squares

- Durer: gave proof that magic square of size 4 (composed of 1,2,…,16) exists
- but! a magic square of size 2 (composed of 1,2,3,4) does not exist
  - why?

a	*b*
c	*d*

$\cancel{a} + b = \cancel{a} + c \Rightarrow b = c$

- which violates uniqueness of items

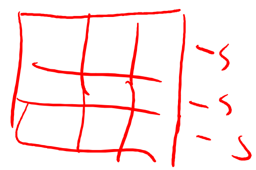
What about size 3?
Can we make a magic square with items 1,2,…,9?

The Search for the 3x3 Magic Square

- a magic square exists if $n > 2$
- brute force for 3 x 3 feasible?

  * brute force – use all combinations possible by trial & error

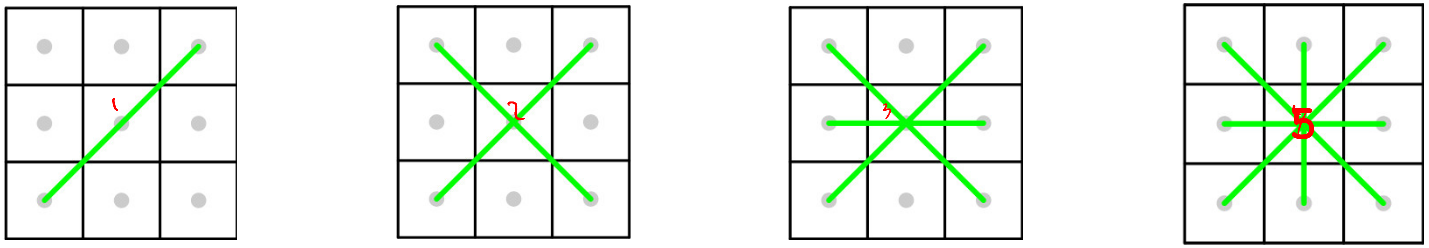
  - all permutations (possible mix) for 9 digits
    - 9 x 8 x 7 x … x 1 = 362880
    - no problem for computers, but challenging for most humans
- what is row/column/diagonal sum *s*?

  - ts ⇒ 1 +2 + 3 + … + 9 = 45; ts = total sum
  - ts = 3s
  - s ⇒ 45 / 3 = 15 per row/col, since there are 3 rows & 3 cols

The Search for the 3x3 Magic Square

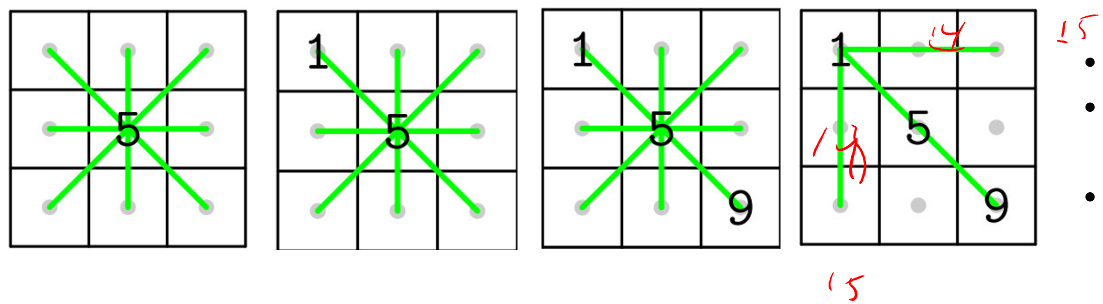
- **hint**: focus on the center
  - summing up 4 lines passing through the center



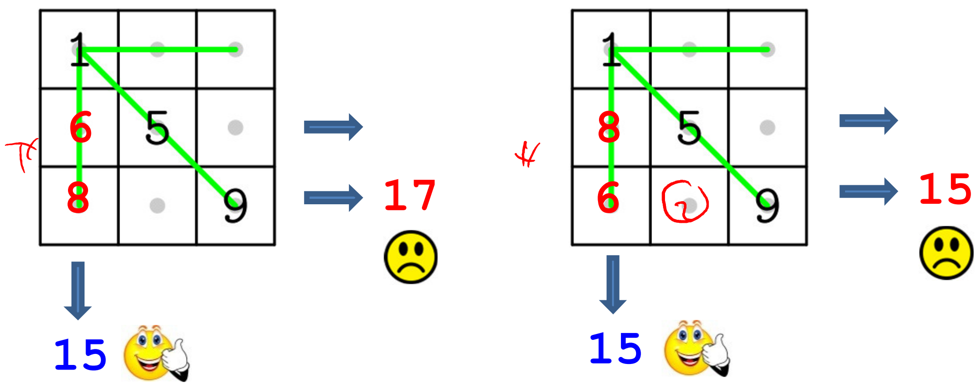
  - let **4s** represent the total sum for 4 lines
  - note that we used all the numbers once to get 4s except the center one which was used 4 times
    - **ts** (sum up all numbers) + **3 x C** (center number, only 3 times since used once already in ts), therefore
  - $4s = ts + 3C$, recall $ts = 3s$
  - $4s - 3s = 3C \Rightarrow s = 3C$
  - $C = s/3$, recall $s = 15$, hence $\Rightarrow C = 5$

The Search for the 3x3 Magic Square

- Now let us analyze where we can put “1”, how about corner?



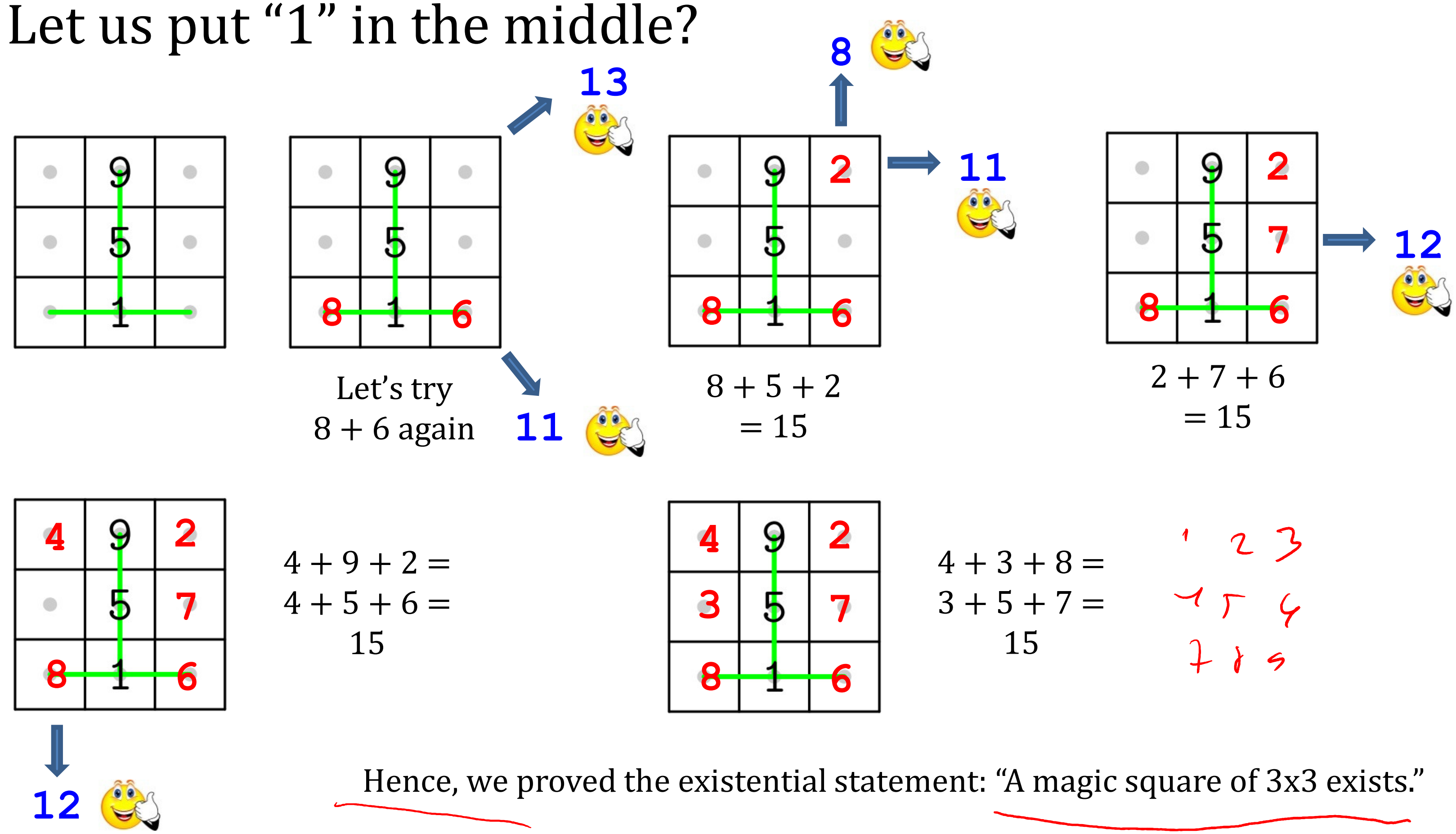
- recall s = 15, therefore we need 14
- 14 = 5 + 9, 6 + 8; only possible combinations
- Let’s try 6 + 8



- Hence, “1” cannot be in the corner

The Search for the 3x3 Magic Square

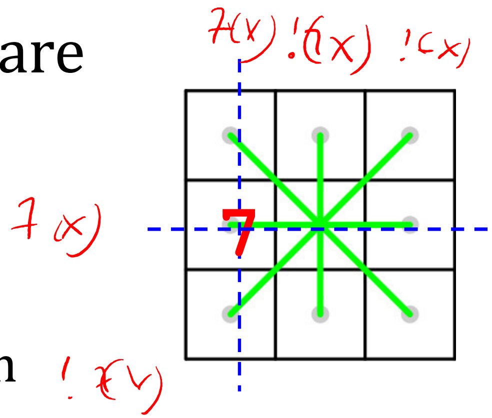
- Let us put "1" in the middle?



Hence, we proved the existential statement: "A magic square of 3x3 exists."

Magic Square, Product not Sum

- magic square: **sums** for rows, cols, diagonals are same
- how about the **product**?
  - cannot use sequential numbers, ex: 1, 2, 3, 4, 5, 6, 7, 8, 9
  - why? take for example “7” in a 3x3 square
    - we put “7” anywhere
    - 7 is part of product with dash lines
    - all the others do not have 7
    - No other number can give 7 in combination



- use of arbitrary positive integers are allowed
- is it possible though?

Magic Square, Product not Sum

2^3	2^8	2^1
2^2	2^4	2^6
2^7	2^0	2^5

$2^0 = 1$

- $2^{x+y} = 2^x * 2^y$
- exponentiation: addition → multiplication

sum: 15 ⟸

4	9	2
3	5	7
8	1	6

→

2^4	2^9	2^2
2^3	2^5	2^7
2^8	2^1	2^6

256

⟹ **product:**
$2^{15} = 2^4 * 2^9 * 2^2$
(32,768)

product:
$\cancel{2^{14}} = 2^3 * 2^8 * 2^1$
2^{12}
(4,096)

- numbers are big, $2^9 = 512$
- how about get numbers less than 300?
  - divide numbers by 2, hence largest would be $2^9 / 2 = 256$ (2^8)
- how about less than 40?

Magic Square, Product not Sum

- numbers less than 40

sum

1	2	0
0	1	2
2	0	1

→

product

2	4	1
1	2	4
4	1	2

x 2

base 2: (from sum magic square in the left)

2^1 (2) : 2^2 (4): 2^0 (1)

1 x 2 x 4 = 8

4 x 1 x 2 = 8

✖

0	2	1
2	1	0
1	0	2

→

1	9	3
9	3	1
3	1	9

x 3

base 3: (from sum magic square in the left)

3^0 (1) : 3^2 (9): 3^1 (3)

9 x 3 x 1 = 27

3 x 1 x 9 = 27

=

2	36	3
9	6	4
12	1	18

from elements of the 2 product magic square above

2 x 1 (2) : 4 x 9 (36): 1 x 3 (3);

9 x 6 x 4 = 216; 216 = 8 x 27

12 x 1 x 18 = 216

100??? Divisible by 9127

- a 6-digit number starting with "100" & divisible by 9127?
  - not that many candidates
- lazy(?) programmers way, (brute force version)

```
for i = 100000 to 100999
    if i is multiple of 9127
        print (i)
```

1st

- mathematical way, paper & pencil (aka hard way :D)
  - 100,000/9127 = 10.956503 ≈ 11 (round to 11)
    - why not 10? Checking, 10 x 9,127 = 91,270 (incorrect) ×
  - 11 x 9,127 = 100,397 (candidate solution) ✓
    - try 12 x 9, 127 = 109,524 (above limit)
  - therefore 11 is the correct answer

2nd

3-digit number N, Remainder One

- a 3-digit number N that gives a remainder of 1 when divided by 2, 3, 4, 5, 6, & 7?
  - we set 3-digits since if not, we can say answer is “1”
    - recall, 1 divided by any number from 2 will give a remainder of 1 (1/N = X rem 1)
  - take note that N/{2,3,4,5,6,7} will give remainder 1
    - hence, (N-1)/{2,3,4,5,6,7} will give us remainder “0”
  - so taking the multiple of 2,3,4,5,6,7
  - 2 x 2 x 3 x 5 x 7 = 420, ; why 4 & 6 are not used?
    - 4 has factors (1, 2 & 4), 6 has (1,2 ,3 & 6)
  - N - 1 = 420; N = 421
  - Try other candidates:
    - 420 x 2 + 1 = 841
    - 420 x 3 + 1 = 1261, not three digits any more; N={421, 841}

Perfect Square That Starts With 31415

- an integer n such that $n^2 = 31415...?$
- note: finite decimal fraction is good enough, $(10x)^2 = 100x^2$ $10^2 x ?$
  - YYY.YYY x 10 ⇒ YYYY.YY, move decimal pt one place to the right
  - $(\text{YYY.YYY})^2$ ⇒ YYYYY.Y, move decimal pt two places to the right
- Just take for now $\sqrt{(31415)} = 177.242771$ (calculator)
  - $177.242771^2 = 31\,414.999872...$ (calculator)
  - $177.243^2 = 31\,415.081049$, round off 3 decimal pts left hand side
  - for left hand side of eqn: move dec pt 3 places to right then square
    - $177.243 \rightarrow 177\,243.00^2$
  - then for right hand side: move dec pt double (3x2=6) places to right
    - $31\,415.081049 \rightarrow 31\,415\,081049.00$
  - Hence, $177\,243^2 = 31\,415\,081\,049$

Two Perfect Squares That Starts With 31415

- another integer n with different first digit such that $n^2 =$ 31415…?
- can we use same method?
- let's try with 3141.5
    - √(3141.5) = 56.0490856… (calculator)
    - 5605^2 = 31 416 025; too big (calculator)
    - $560\,491^2$ = 314 150 161 081; Ok
- hence the two perfect squares starting with 31415 are:
    - 177 243 & 560 491

Just 7 & 13

Imagine a country with currency of only 7 & 13 ewan coins

Two person with same amount of coins for each type

- possible for one person to pay 6 ewans to the other?
  - Yes: 6 = 1 x 13 ewans – 1 x 7 ewans; easy
- how about paying 1 ewan?
  - Yes: 2 x 7 ewans – 1 x 13 ewans = 1 ewan; or 7 – 6 = 1 (using prev knowledge above)
- 2 ewans?
  - Yes: 4 x 7 ewans – 2 x 13 ewans = 2 ewans ; or 2 x 1 = 2 (use prev knowledge again)

- mathematically speaking, for any integer amount (for any integer c)
  - $7x + 13y = c$

Now just 15 & 21

What if coins were changed to 15 & 21 only   3·5   3·7

- Possible to pay 6 ewans?   9
  - 6 = 1 x 21 ewans – 1 x 15 ewans   2·3
- how about paying 8 ewans?   2·2
  - No: obstacle, coins are multiples of 3, cannot get 8 or 1
- 3 ewans?
  - Yes: 6 = 1 x 21 – 1 x 15 → 9 = 1 x 15 – 6 → 3 = 9 – 6; or
- Unfolding to find out how we paid for 3 ewans:
  - 9 = 2 x 15 ewans – 1 x 21 ewans, 3 = 3 x 15 ewans – 2 x 21 ewans
- Hence, any multiple of 3 can be paid

- mathematically speaking
  - 15x + 21y = *c* ⇔ multiple of 3 (has integer solutions iff *c* multiple of 3)

Ewan challenge (Assignment)

- With 7 & 13 ewan coins, is it possible to pay 5 ewans?
- How about with 15 & 21 ewan coins?

Lotsa Hotel

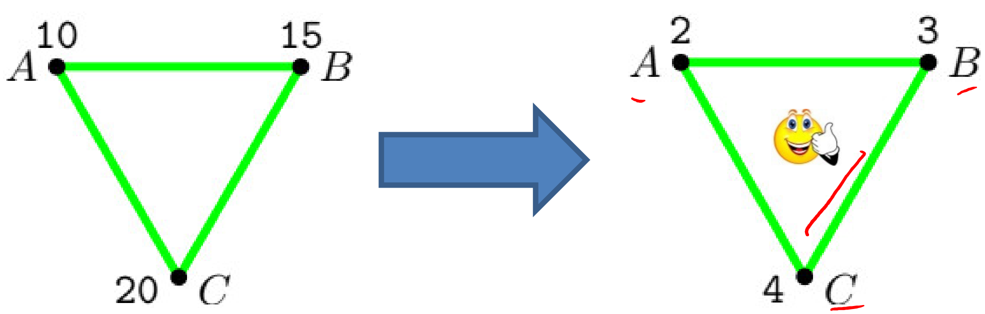
- One night stay vouchers for 3 hotels
  - one voucher for one night
    - voucher → can be used to stay in the hotel (like coupon, to stay free)
  - cannot use/stay two consecutive/successive nights in one hotel
  - Hotel A (10 vouchers)
  - Hotel B (15 vouchers)
  - Hotel C (20 vouchers)

- Can you use all 45 vouchers (10+15+20)
  - for 45 consecutive nights changing hotels each night?

Hotels & Paths

Let us now shift from Number Theory to Graph Theory :D

- Hotels A(10), B(15), & C (25)
  - change hotel every night for 45 (10 + 15 +20) nights



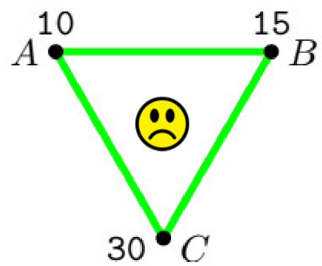
  - 10, 15 & 20 multiples of 5; we can simplify them into 2, 3 & 4
    - hence, total path should be: length of 9 repeated 5 times
  - must be different end points: ex: A→B, B→C; not A→B, B→A
  - since 4 C's, let us set C as every second point

- A→
- C→
- A→
- C→
- B→
- C→
- B→
- C→

A Path Does Not Always Exist

25

- Hotels A(10), B(15), & C (30)
  - change hotel every night for 55 (10 + 15 + 30) nights



  - Is it possible?
  - Obstacle: too many vouchers for C
  - How to prove?
  - With 30 vouchers for C, there should at least be 29 hotels in between to obey condition, but we only have 25 (A:10, B:15), hence not possible
    - **C→A→C→A→C→A→C→A→C→A→C→A→C→A→C→A→C→A→C→A→**
    - **C→B→C→B→C→B→C→B→C→B→C→B→C→B→C→B→C→B→C→B→**
    - **C→B→C→B→C→B→C→B→C→B→C→ →C→ →C→ →C→ →C→ →**

Mathematical Thinking –Find Example

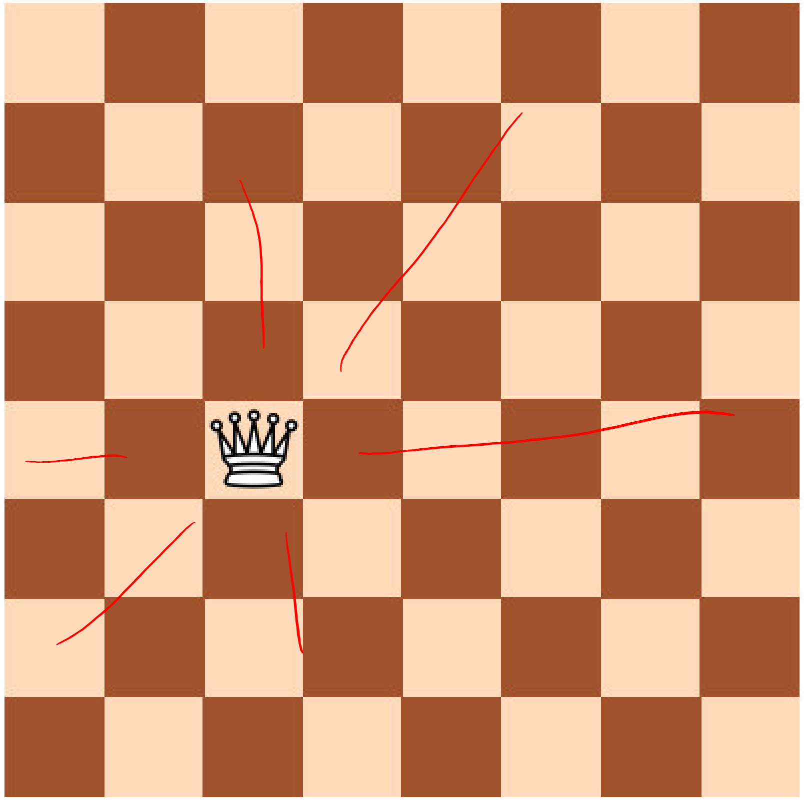
COMPUTER SEARCH

Computer Search

- There are times when solution exists but not easy to implement manually
- Utilize computers!
- We discuss a puzzle
  - n queens
  - design program to solve puzzle fast on your laptop

- n-Queens: Brute Force Search
- n-Queens: Backtracking

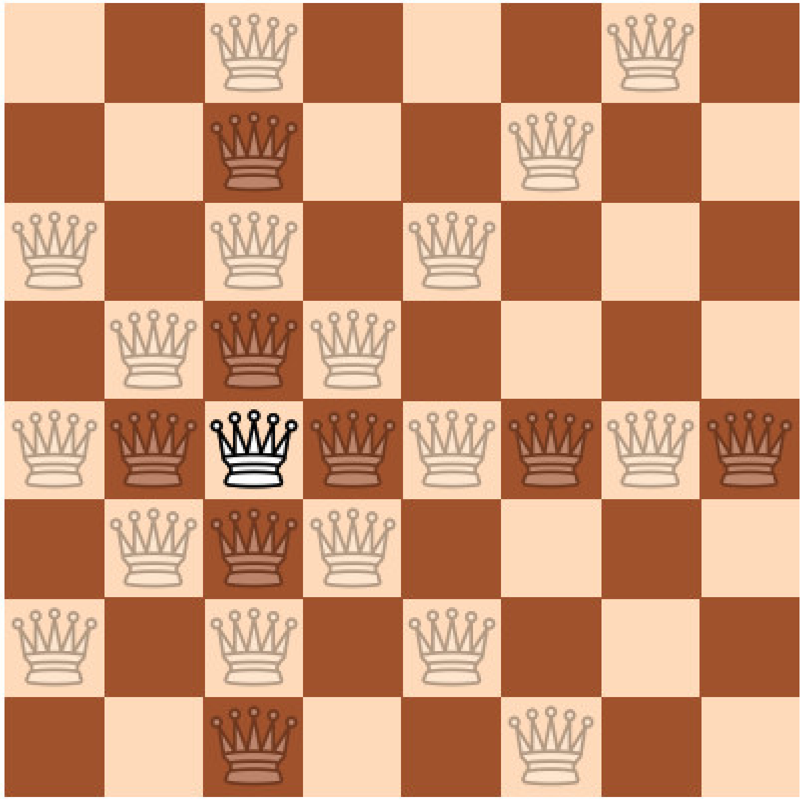
Chess Queen



Chess queen:

- horizontally
- vertically
- diagonally

Chess Queen



Chess queen:

- horizontally
- vertically
- diagonally

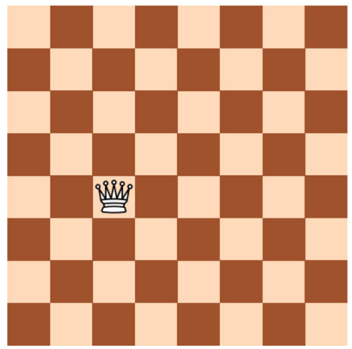
n-Queens Challenge

The Challenge

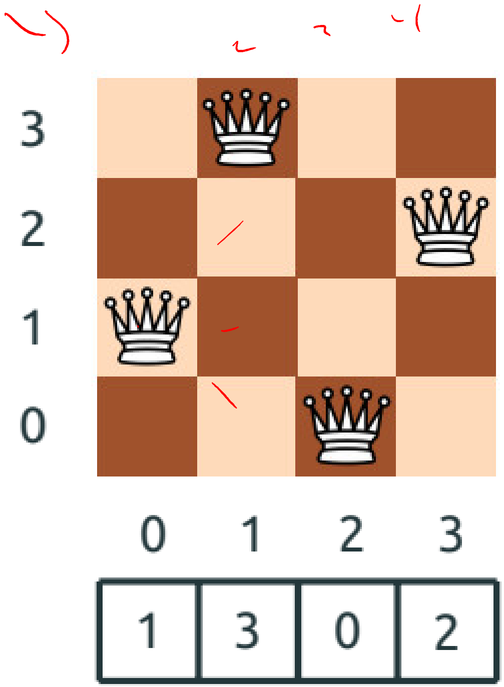
- Is it possible to place n Queens on an n x n chessboard wherein no two or more queens attack each other?
  - it is known that this is possible for all $n \geq 4$
  - although, it is already difficult to manually implement when $n = 8$

Assumptions

- Since *n* queens on an *n x n* chessboard, there must be exactly one queen in each column
  - two queens in one column will attack each other
  - if there is a column with no queen, then the total number of queens is less than *n*
- At the same time and for the same reason, there should be one queen for each row.



Permutation is the Answer



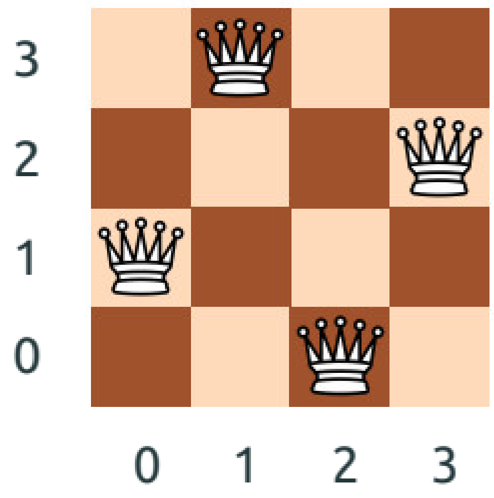
- Permutation– order of selection is a factor
- Combination – order of selection not a factor

Ex: Permutation/Combination pairs from the set {A, B, C, D, E}

Permutation:
AB AC AD AE
BA BC BD BE
CA CB CD CE
DA DB DC DE
EA EB EC ED

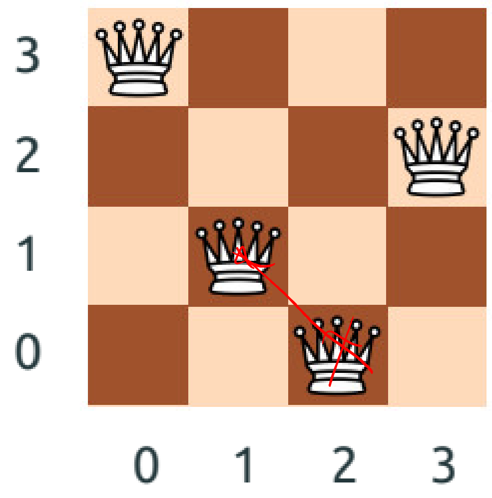
Combination:
AB AC AD AE BC BE CD CE DE

But, Not Every Permutation is Correct Answer



1	3	0	2

✓

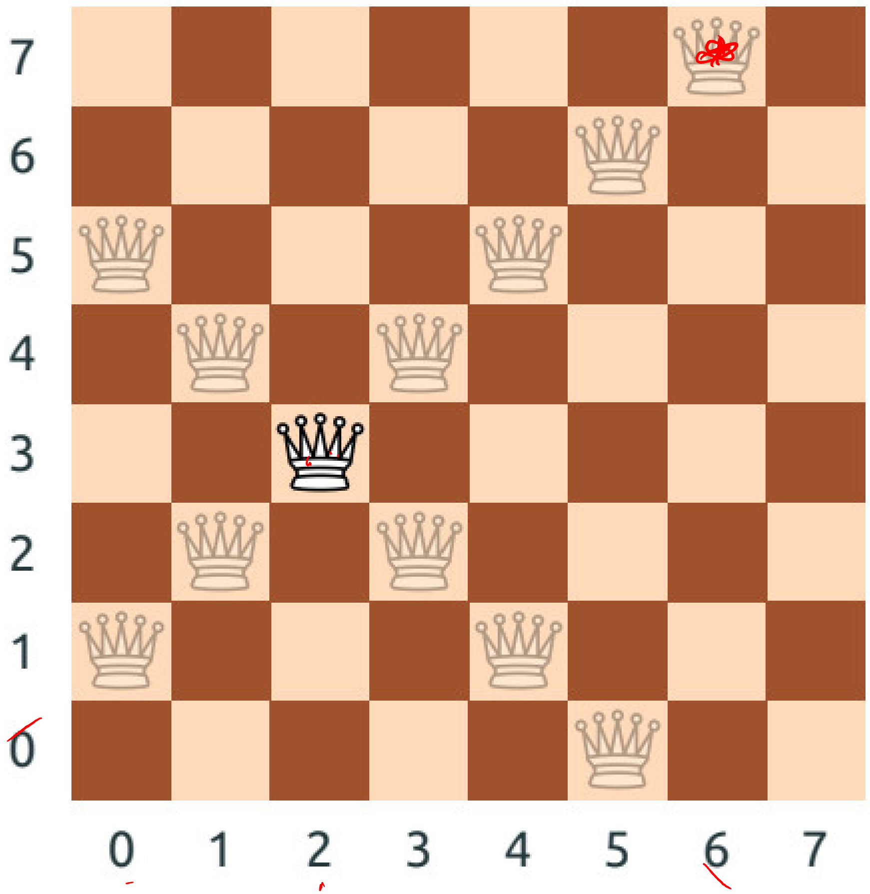


3	1	0	2

✗

How to check if a given permutation is a correct answer?

When Occupied Cells are in the Same Diagonal



How to check if a given permutation is a correct answer?

- when cell $[i_1, j_1]$ & cell $[i_2, j_2]$ are on the same diagonal iff $|i_1 - i_2| = |j_1 - j_2|$
- example: [4,2] & [0,6]
  - $|4-0| = |2-6|$
  - $4 = 4$

The Answer?

- Using python

```
import itertools as it

def is_answer(perm):
    for  (i1, i2) in it.combinations(range(len(perm)), 2):
        if abs(i1 - i2) == abs (perm[i1] - perm[i2]):
            return False
        return True

assert(is_answer([1, 3, 0, 2]) == True)
sssert(is_answer([3, 1, 0, 2]) == False)
```

- itertools– functions for efficient looping
- it.combinations(input) – return length "2" from input set "perm"
- range(n) – generate sequence of numbers start from 0 to n-1
- len(var) – return length of var(string, array, list, etc.)
- assert() – debugging tool to test conditions

Complete Program Using Brute Force Search

- Using python

Online python compilers:

- https://repl.it/languages/python3
- https://www.tutorialspoint.com/execute_python_online.php
- https://www.onlinegdb.com/online_python_compiler

```
import itertools as it

def is_answer(perm):
   for  (i1, i2) in it.combinations(range(len(perm)), 2):
      if abs(i1 - i2) == abs (perm[i1] - perm[i2]):
         return False
      return True


for perm in it.permutations(range(8)):
   if is_answer(perm):
      print(perm)
      exit()
```

- it.permutations – return permutations of input length for given input set

Conclusion

- The program can immediately give an answer for $n=8$
- Although it takes too long when n = 13
  - more so if $n = 20$

- Next section
  - optimize the program so that it will be able to quickly find an answer even if n = 20

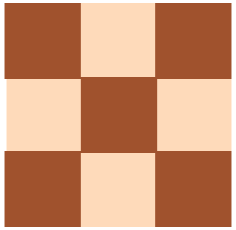
Computer Search Outline

- n-Queens: Brute Force Search
- n-Queens: Backtracking

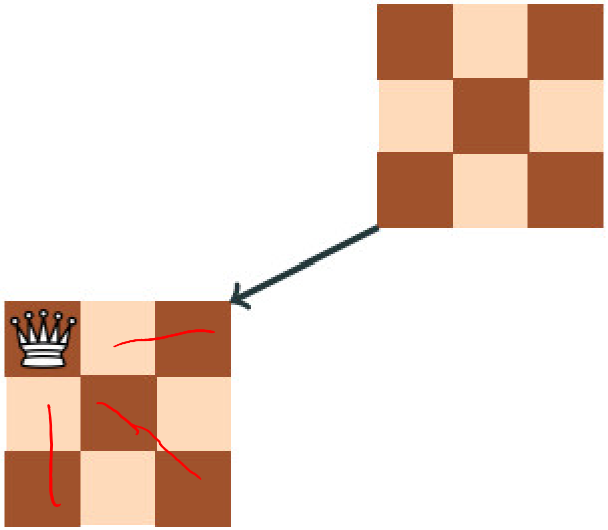
Main Idea

- Construct permutation piece by piece
- Backtrack if the current partial permutation cannot be extended to a valid answer

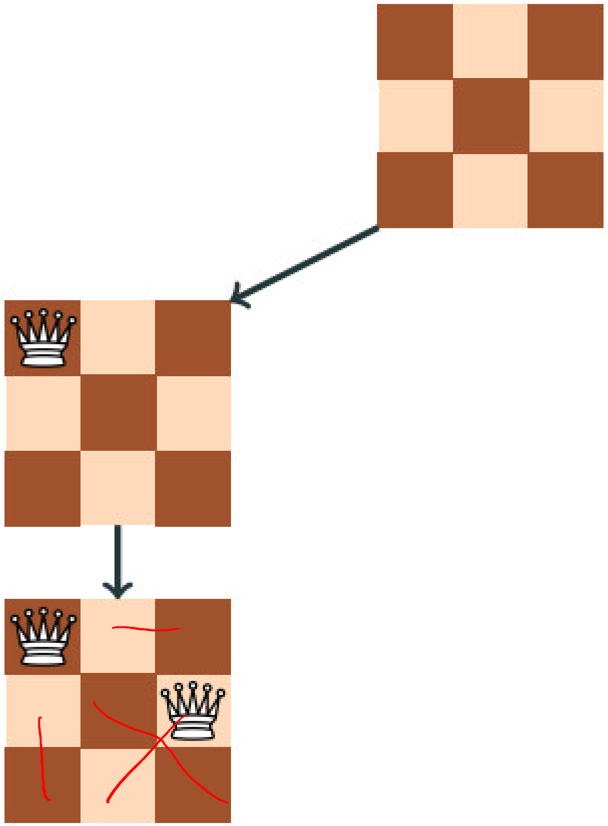
Backtracking on a 3 x 3 Board



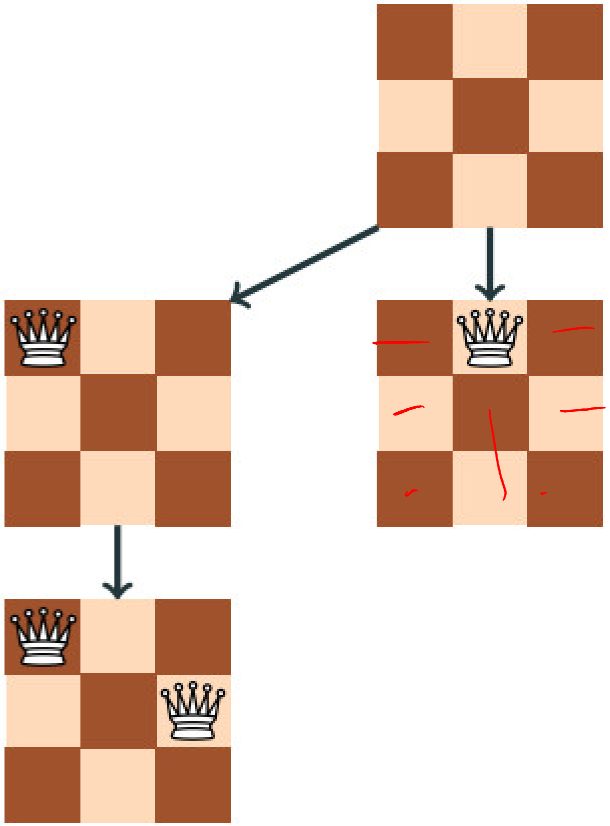
Backtracking on a 3 x 3 Board



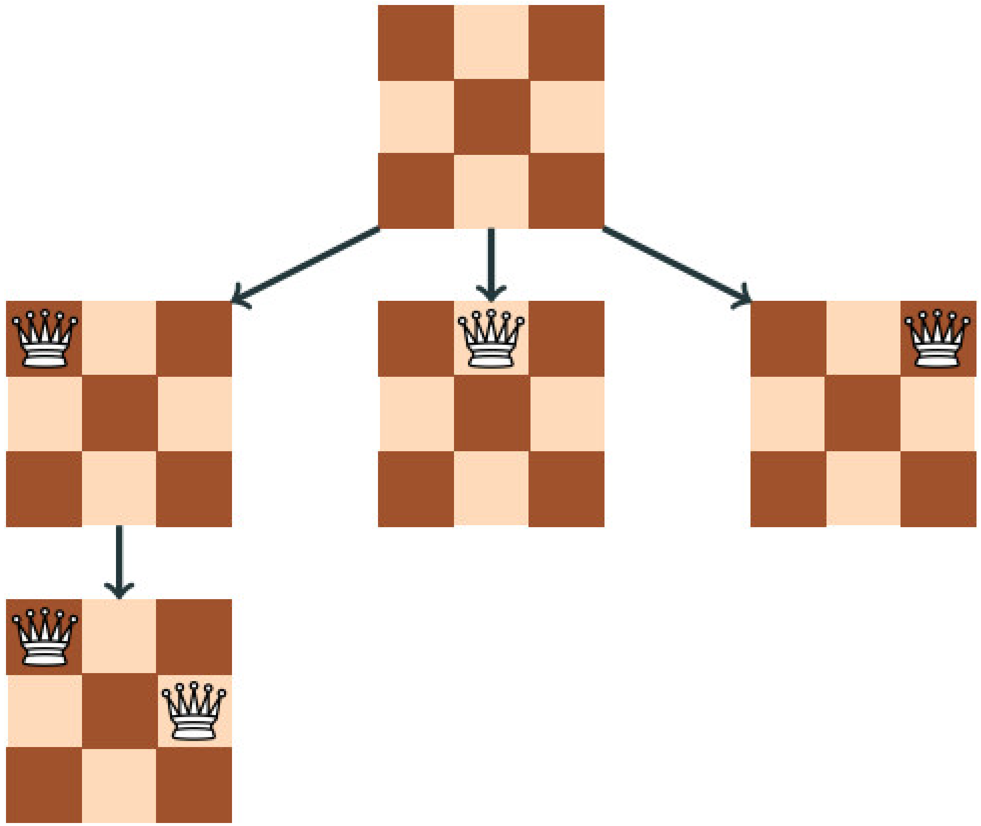
Backtracking on a 3 x 3 Board



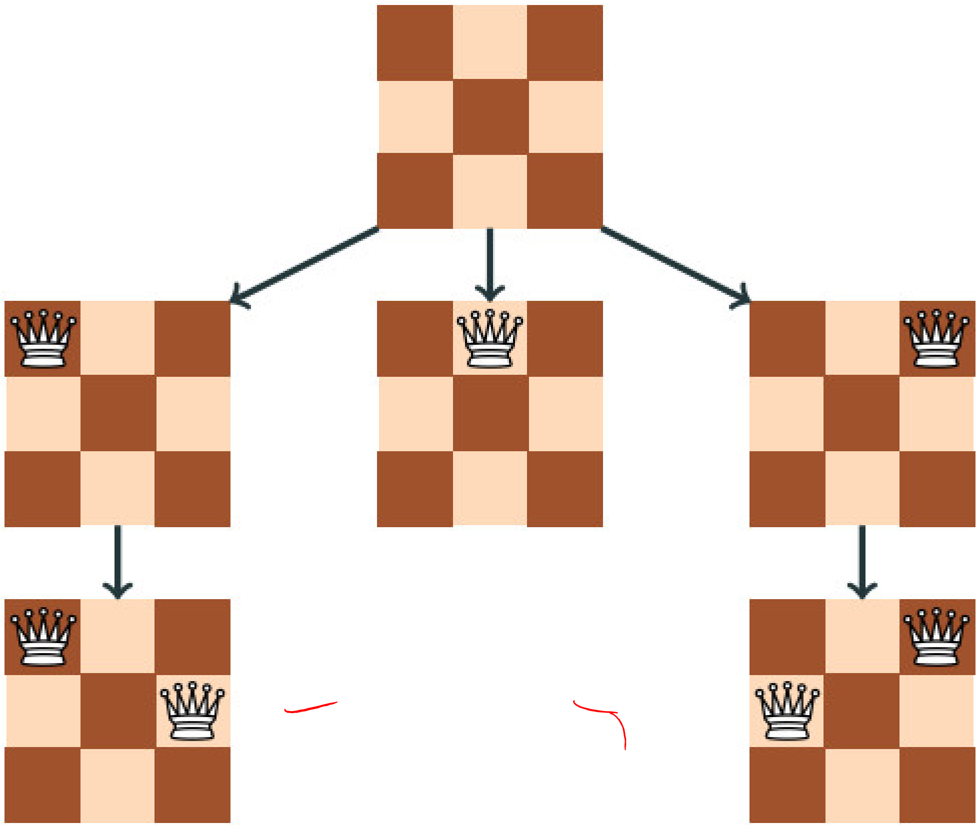
Backtracking on a 3 x 3 Board



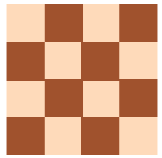
Backtracking on a 3 x 3 Board



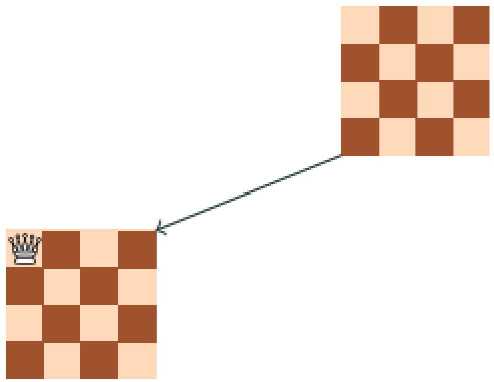
Backtracking on a 3 x 3 Board



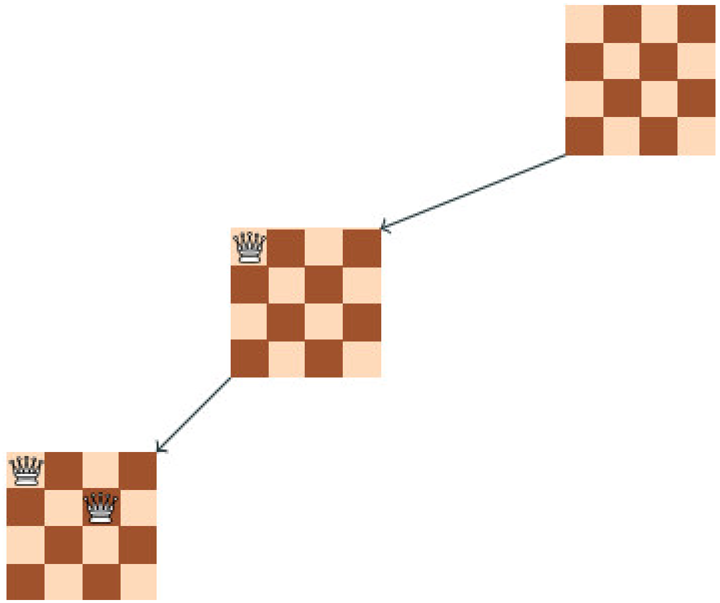
Computer Search – Backtracking



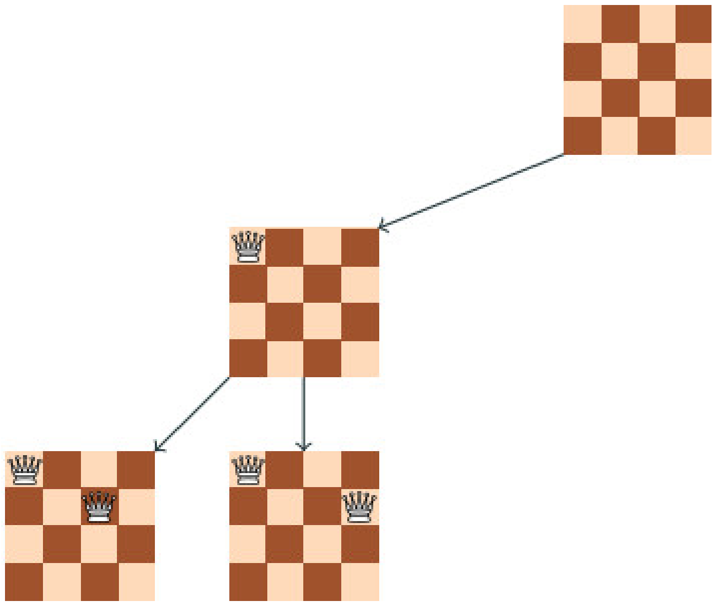
Computer Search – Backtracking



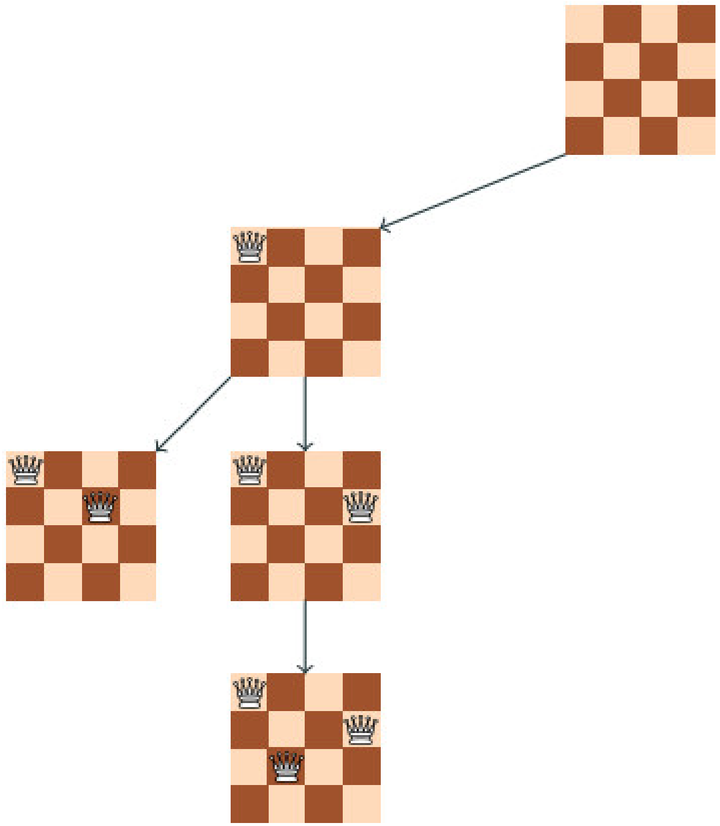
Computer Search – Backtracking



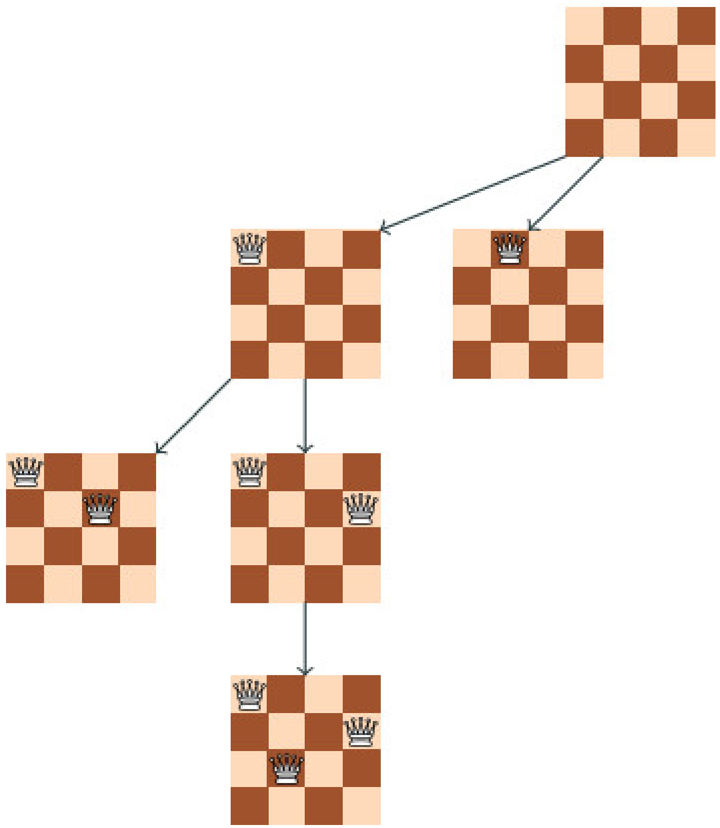
Computer Search – Backtracking



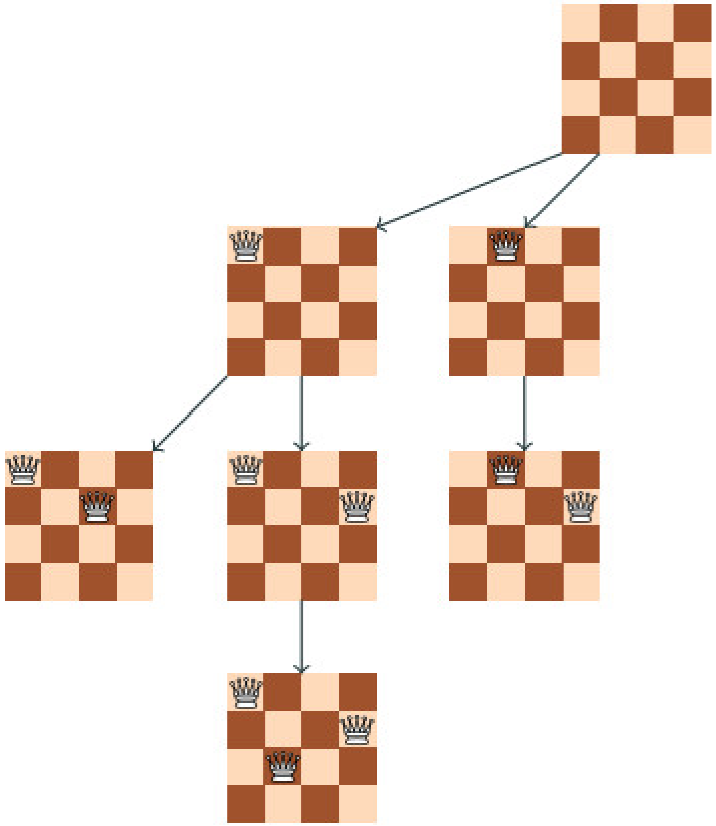
Computer Search – Backtracking



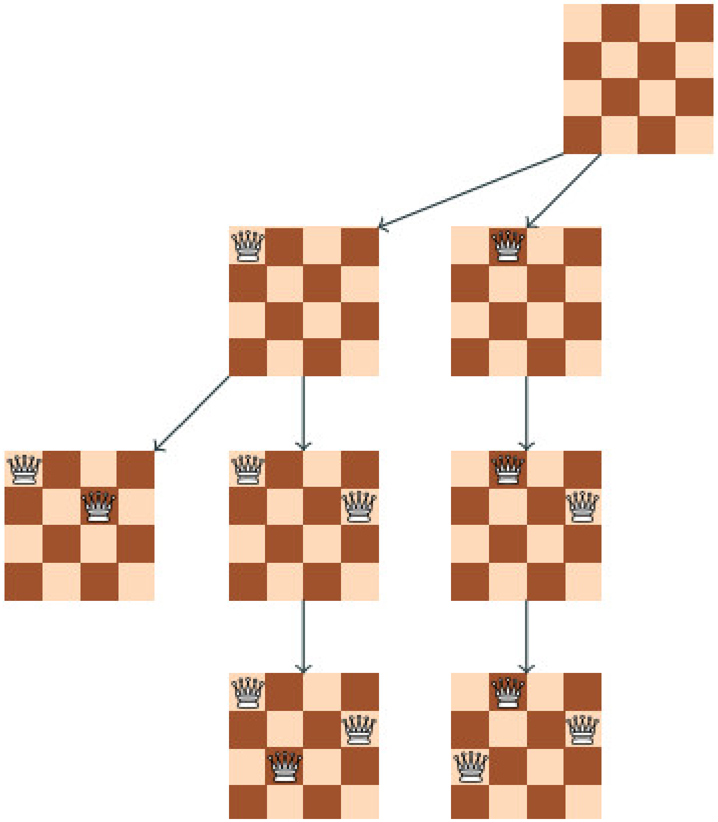
Computer Search – Backtracking



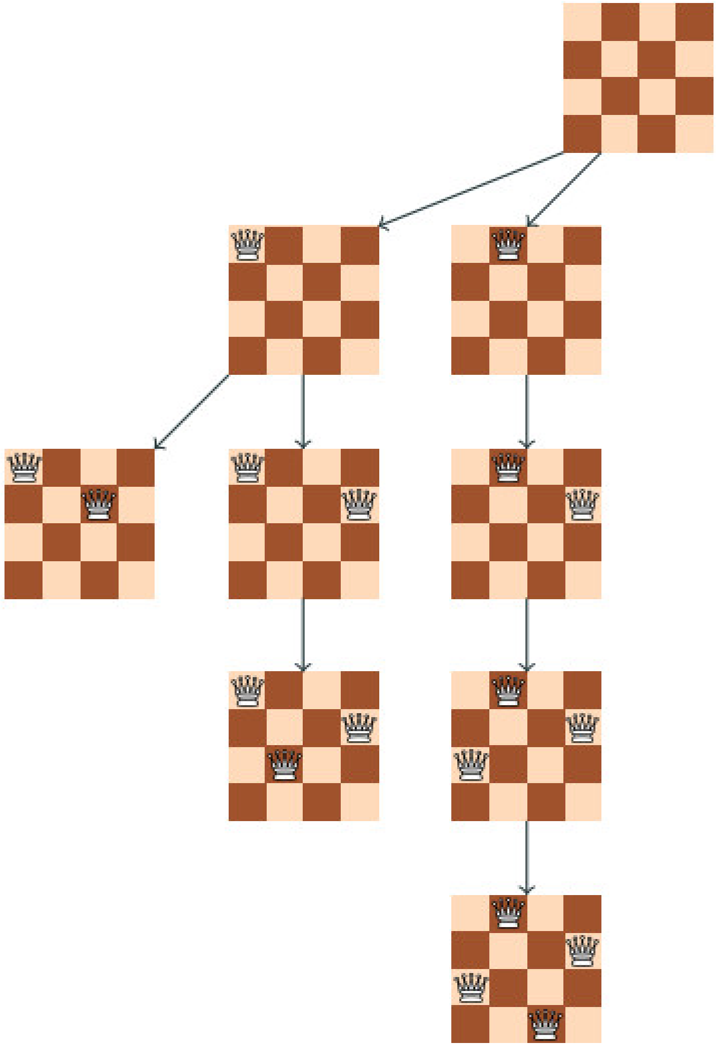
Computer Search – Backtracking



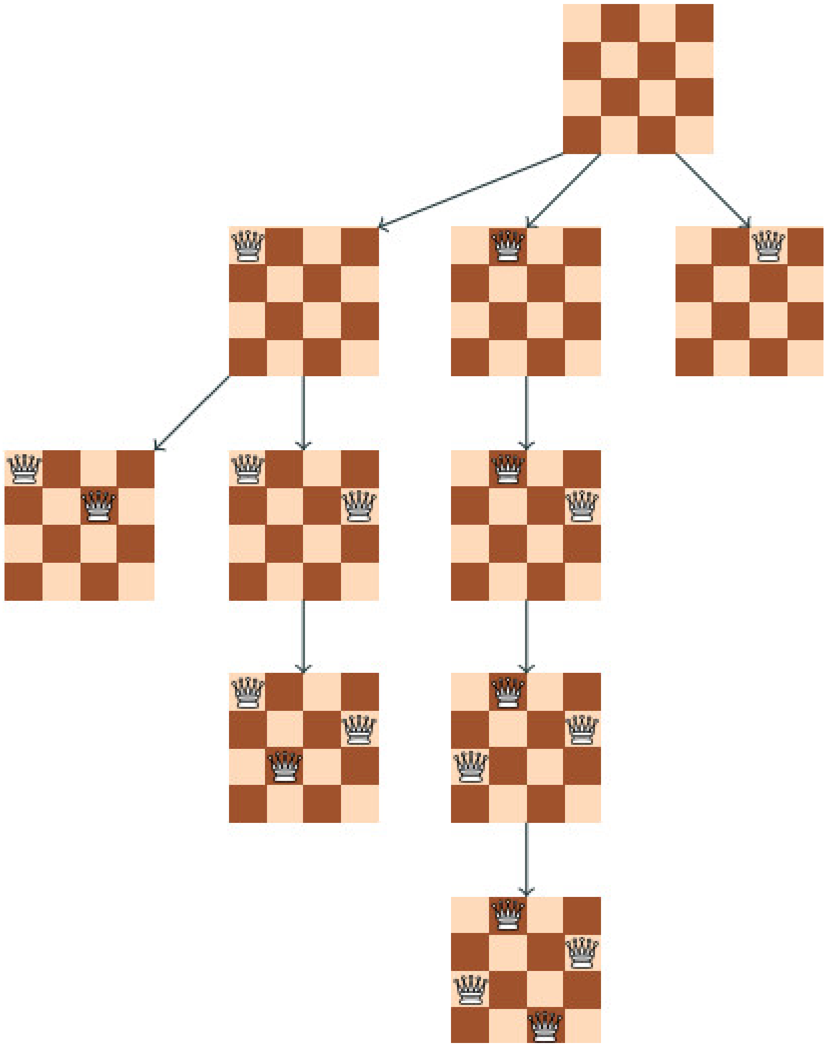
Computer Search – Backtracking



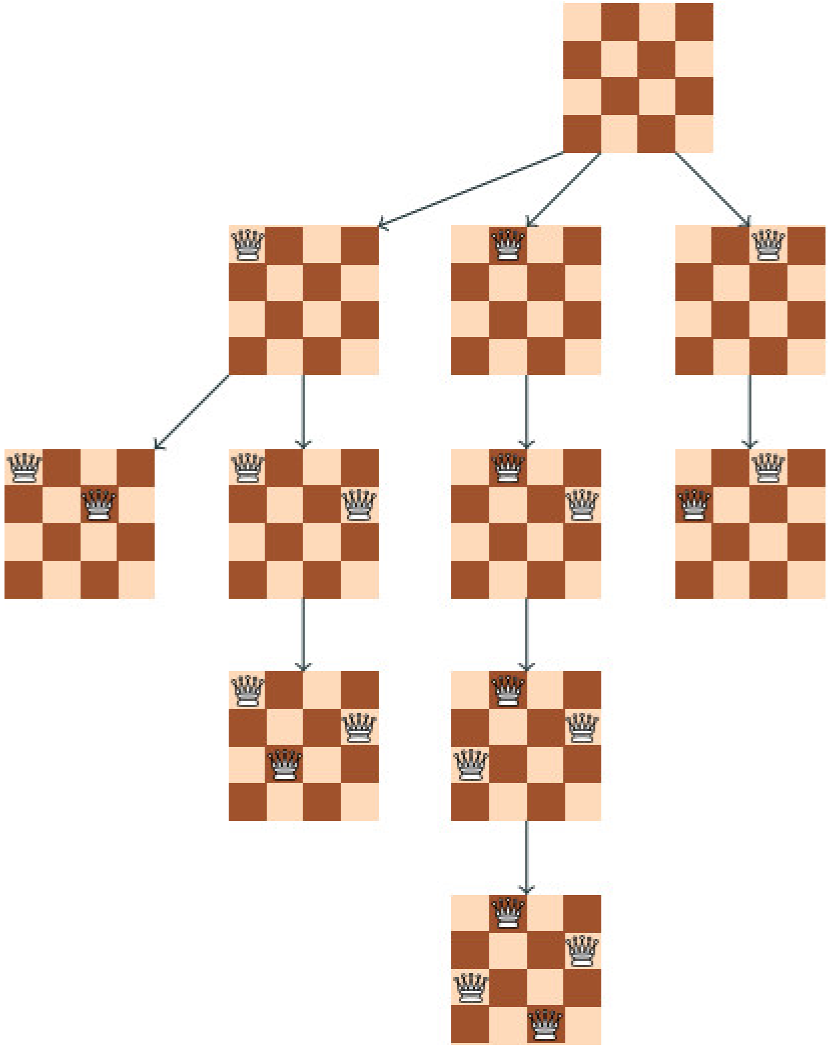
Computer Search – Backtracking



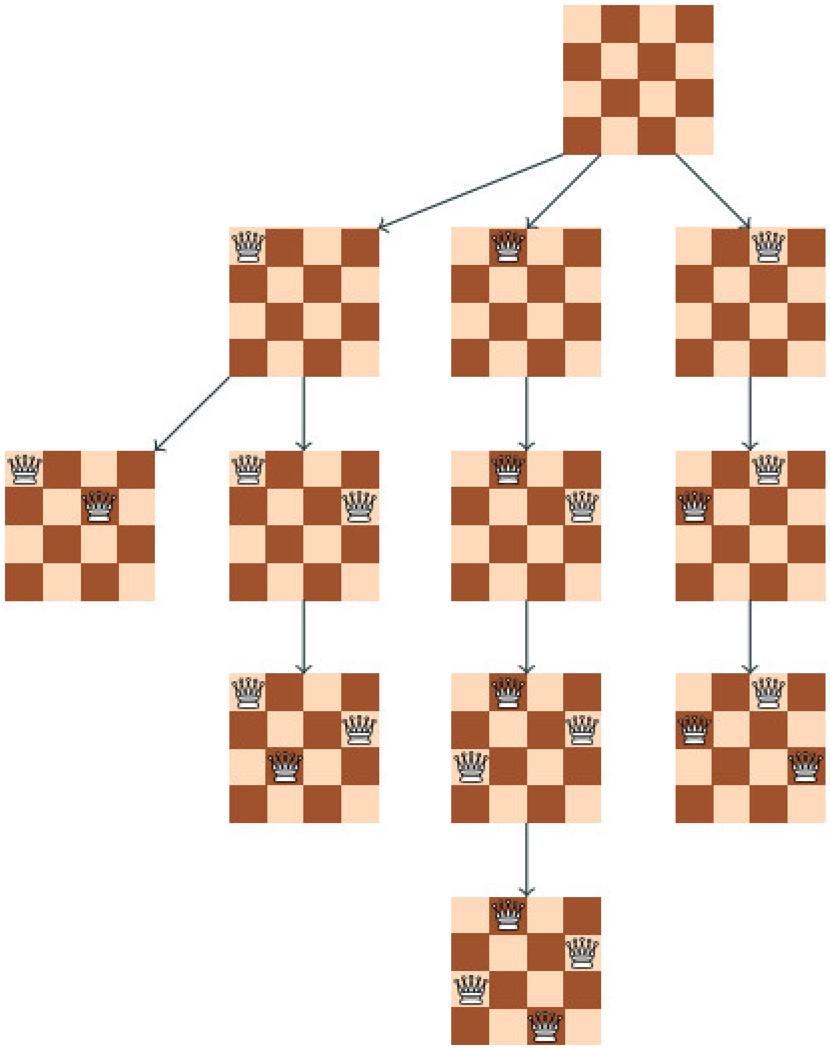
Computer Search – Backtracking



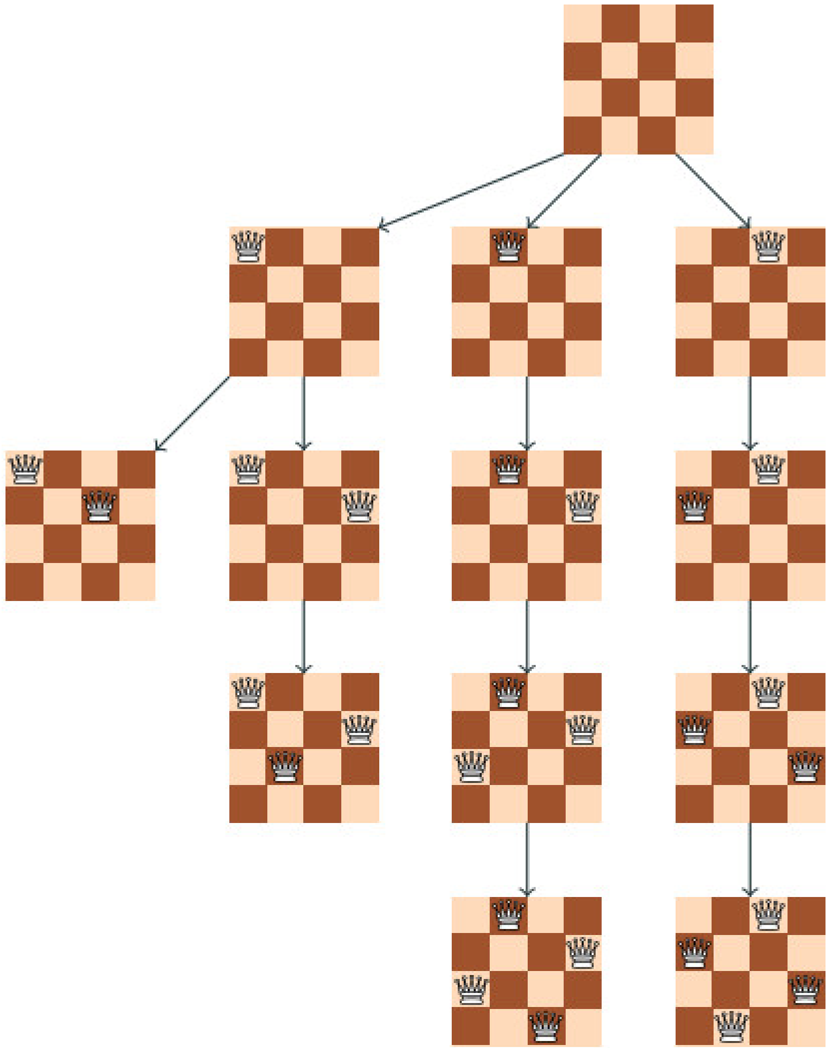
Computer Search – Backtracking



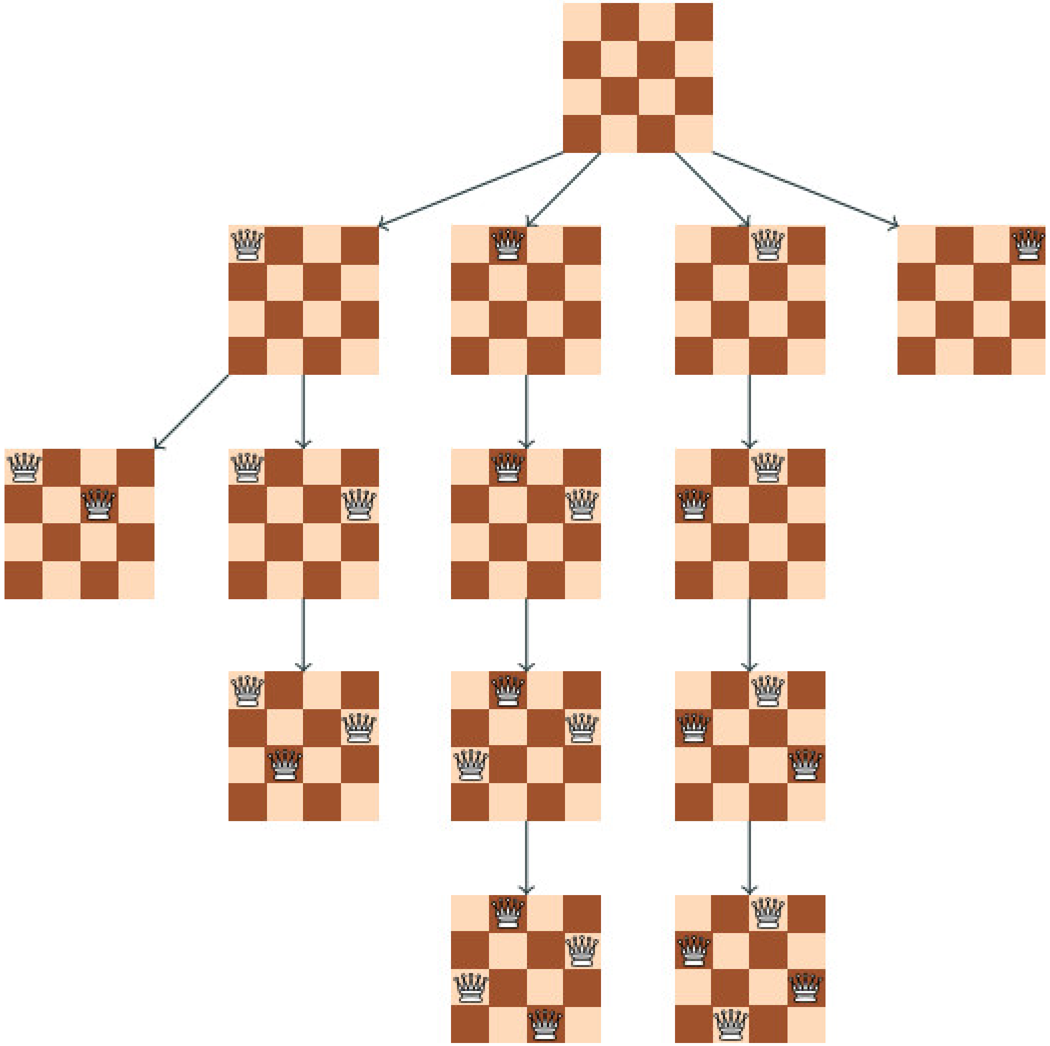
Computer Search – Backtracking



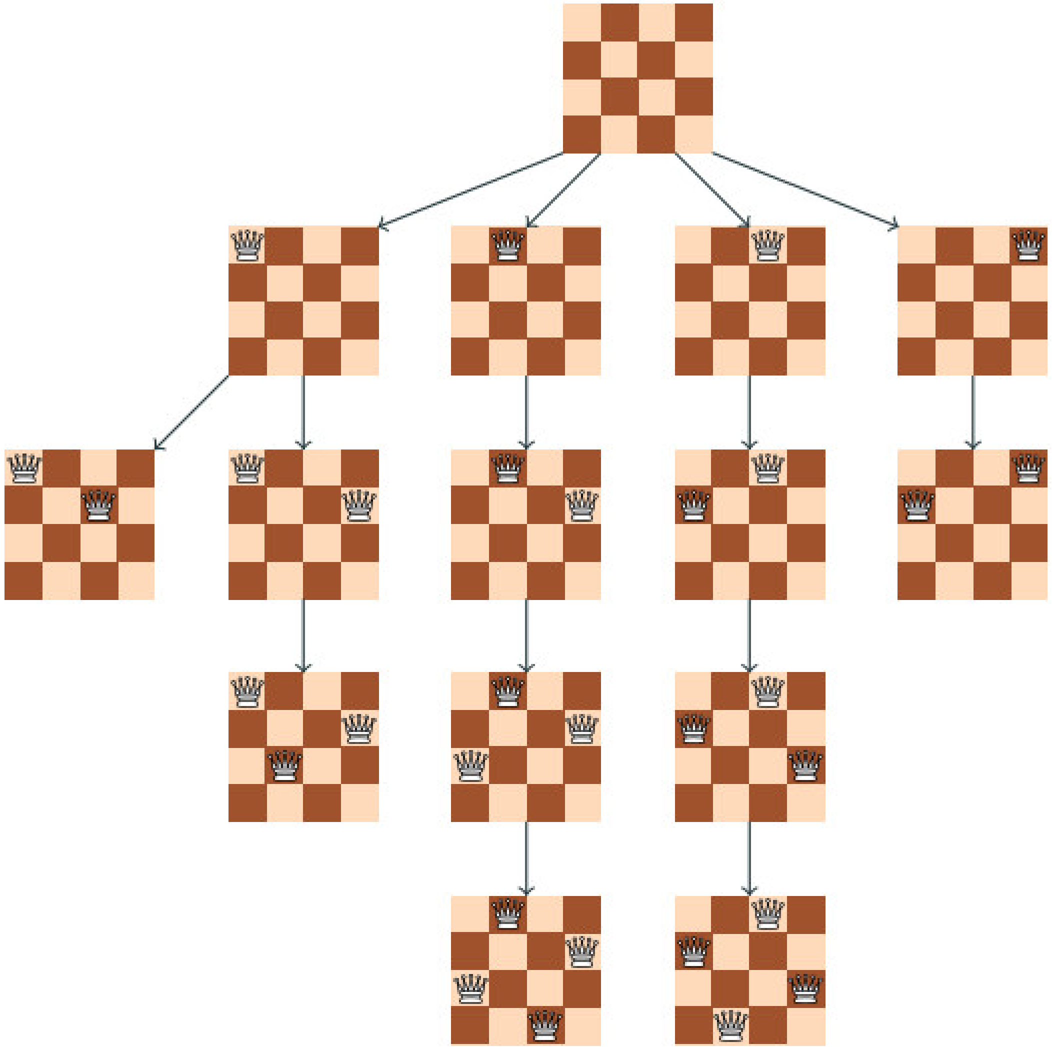
Computer Search – Backtracking



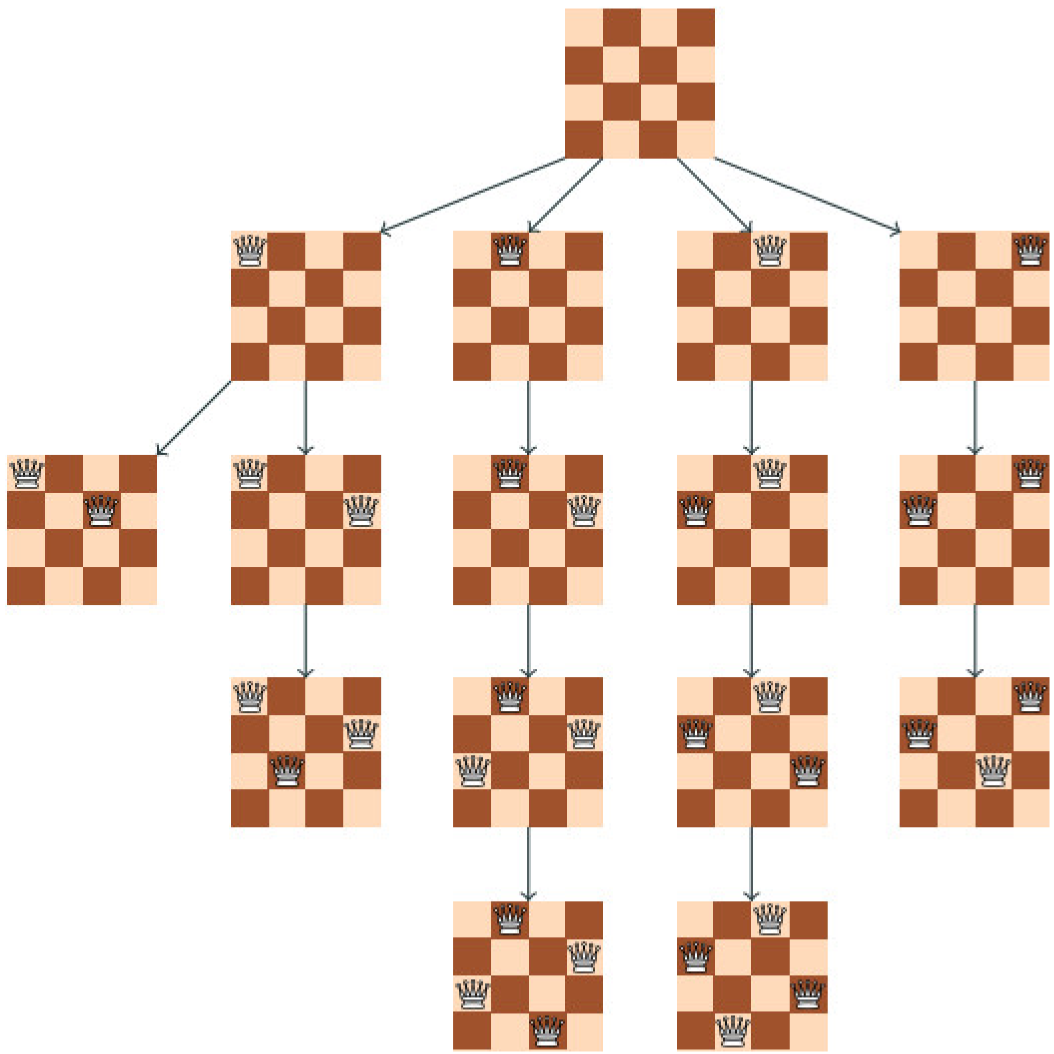
Computer Search – Backtracking



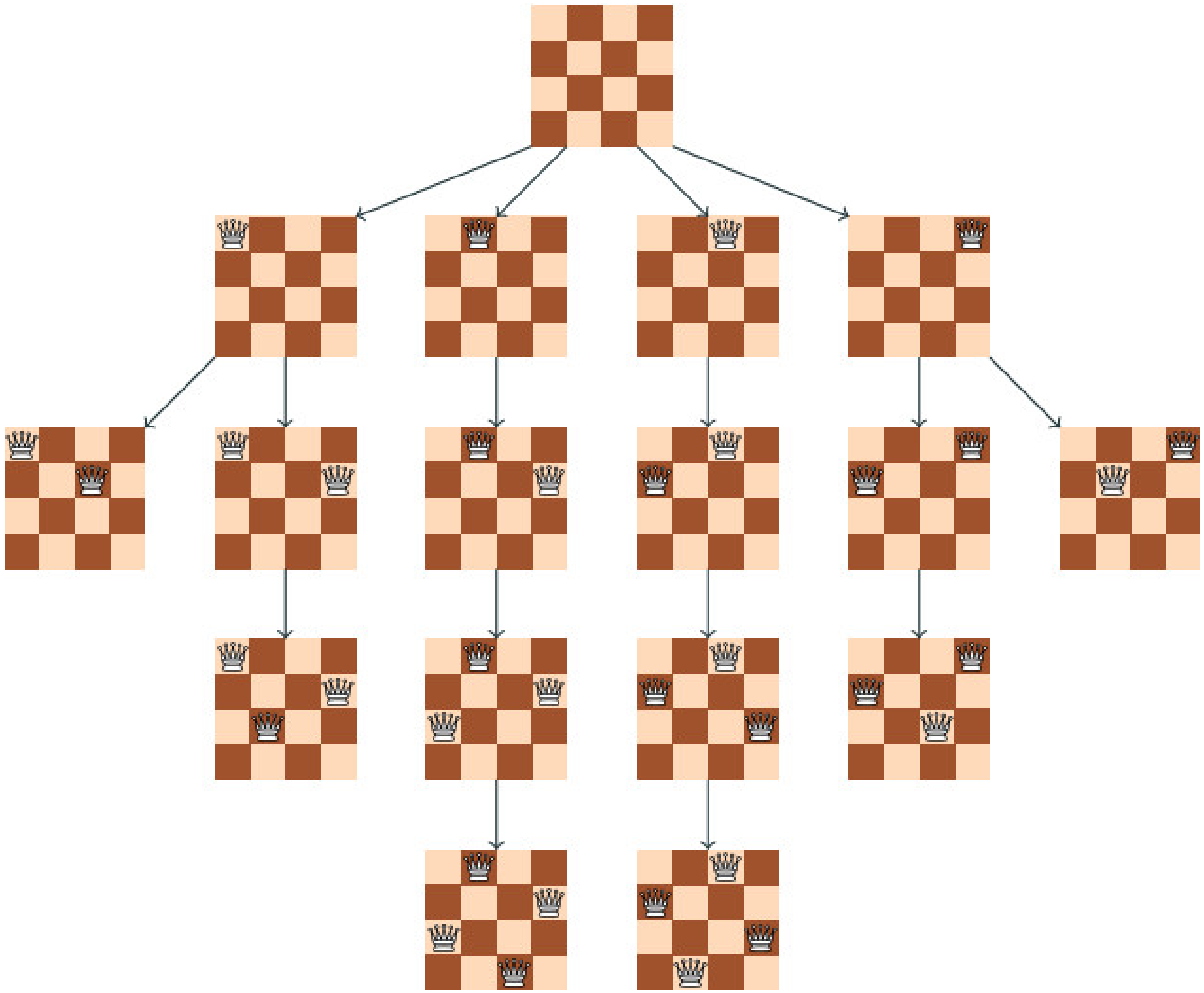
Computer Search – Backtracking



Computer Search – Backtracking



Computer Search – Backtracking



Generating All Permutations

- Using python

```
def generate_permutations(perm, n):
    if len(perm) == n:
        print(perm)
        return

    for k in range(n):
        if k not in perm:
            perm.append(k)
            generate_permutations(perm, n)
            perm.pop()

generate_permutations(perm = [], n = 4)
```

Output

```
[0, 1, 2, 3]
[0, 1, 3, 3]
[0, 2, 1, 3]
[0, 2, 3, 1]
[0, 3, 1, 2]
[0, 3, 2, 1]
[1, 0, 2, 3]
[1, 0, 3, 2]
[1, 2, 0, 3]
[1, 2, 3, 0]
[1, 3, 0, 2]
[1, 3, 2, 0]
[2, 0, 1, 3]
[2, 0, 3, 1]
[2, 1, 0, 3]
[2, 1, 3, 0]
[2, 3, 0, 1]
[2, 3, 1, 0]
[3, 0, 1, 2]
[3, 0, 2, 1]
[3, 1, 0, 2]
[3, 1, 2, 0]
[3, 2, 0, 1]
[3, 2, 1, 0]
```

Idea

- If current (partial) permutation cannot be extended to an answer (i.e. there are two queens attacking each other), stop trying to extend it

```
def can_be_extended_to_answer(perm):
    i = len(perm) - 1
    for j in range(i):
        if i - j == abs(perm[i] - perm[j]):
            return False
    return True
```

Resulting Program

- Complete code using python

$n = 20$

n_1, n_2 ... n_{20}

```
def can_be_extended_to_answer(perm):
    i = len(perm) - 1
    for j in range(i):
        if i - j == abs(perm[i] - perm[j]):
            return False
    return True

def extend(perm, n):
    if len(perm) == n:
        print(perm)
        exit()

    for k in range(n):
        if k not in perm:
            perm.append(k)
            if can_be_extended_to_answer(perm):
                extend(perm, n)
            perm.pop()

extend(perm = [], n = 20)
```

- append(k)– adds item “k” into end of the list
- pop() – removes item at end of the list and gives the item to user

Summary

- main idea of backtracking
  - cut dead ends of the recursion tree
- since many ends are dead
  - it works faster than naïve enumeration of all permutations

Thank you.