◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 - 釣�(♡

# Course Introduction

Mattox Beckman

University of Illinois at Urbana-Champaign Department of Computer Science

Fall, 2023

▲ロト ▲ □ ト ▲ 三 ト ▲ 三 ト つ Q ()

## Objectives

- Describe four patterns of brute force;
- Describe the times when a brute force solution is necessary.
- Describe some techniques to optimize brute force algorithms.

▲ロ▶▲掃▶▲ヨ▶▲ヨ▶ ヨーのQ@

#### What is it?

- You must traverse the entire problem space to get the answer.
- Sometimes you can prune the problem space.

- max=a[0]; // why not just put 0 here?
  for(int i=1; i<7; i++)</pre>
- if (a[i]>max) max=a[i];

▲ロ▶▲掃▶▲ヨ▶▲ヨ▶ ヨーのQ@

#### When to Use It

- Tradeoffs
  - Bad: It's slow!
  - Good: It's simple! More likely to give correct solution.

#### Three situations:

- When you have no choice.
- When the problem set is small.
- To verify your real solution!Introduction

◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 - 釣�(♡

## Categories

- Code Pattern
   Iterative
   Recursive
   Traversal Pattern
   Filtering
  - GeneratingIntroduction

▲ロ▶▲掃▶▲ヨ▶▲ヨ▶ ヨーのQ@

## Speed

- Use bits instead of boolean arrays
- Use primitive types when appropriate:
  - int32 instead of int64
  - arrays instead of vector
  - character arrays instead of string
- Prefer iteration to recursion
- The STL algorithm include has next\_permutation, which is very fast
- Declare large data structures in the global scope

Discussion

▲ロ▶▲掃▶▲ヨ▶▲ヨ▶ ヨーのQ@

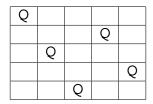
#### The *n*-queens problem

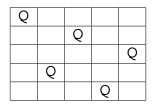
	х		х	
х	х	х		
х	Q	х	х	х
х	х	х		
	х		х	

- ▶ If you don't know chess, you might want to learn the basic rules.
- Classic problem: place n queens on a  $n \times n$  chessboard.
  - How many ways can you do it?

Discussion

#### Two examples





### How to write this?

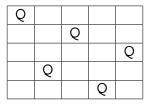
```
Attempt 1 : Massive nested for loops
   vvi board(8,vi(8)); // get
1
   count = 0;
2
   for(i=0; i<7; ++i) {</pre>
3
     board[0][i] = 1; // place queen
4
     for(j=0; j<7; ++j)</pre>
5
        if (! collides(board,1,j)) {
6
          board[1][j] = 1;
7
          for(k=0; ...) ; // rest of program
8
          board[1][j] = 0;
9
        }
10
     board[0][i] = 0; // remove queen
11
Final position; if no collisions increment count.
```

▶ What do you think of this code? 8<sup>8</sup> attempts....

Discussion

◆□▶ ◆□▶ ◆ □▶ ◆ □▶ ○ □ ○ ○ ○ ○

#### Improvements



- We don't need  $8^8$  checks.
- Instead of modeling the chess board, model where the queens are placed.

This example is {0,3,1,4,2}.

Introduction and Objectives	Discussion 000	Example: <i>n</i> Queens
Backtracking		
Example cod	e from Competive Programming 4	
void backtrac	k(int c) {	
2 if ((c == 8	) && (row[b] == a)) { // a	candidate solution
₃ printf("%	2d %d", ++lineCounter, row[	[0]+1);
4 for (int	j = 1; j < 8; ++j)	
5 printf(	" %d", row[j]+1);	
6 printf("\	n");	
7 return; /	/ optional statement	
8 }		
9 for (int r	= 0; r < 8; ++r) { // try a	ill possible row
10 if ((c ==	b) && (r != a)) continue;	<pre>// early pruning</pre>
11 if (canPl	ace(r, c)) // can place a G	lueen here?
12 row[c]	= r, backtrack(c+1); // put	t here and recurse
13 }		
14 }		
	< □ >	4日、4日、4日、4日、日、9へで

\_\_\_\_

▲ロト ▲ □ ト ▲ 三 ト ▲ 三 ト つ Q ()

## **Checking Placement**

```
Don't walk the diagonals; use math!
bool canPlace(int r, int c) {
  for (int prev = 0; prev < c; ++prev) // check previou
    if ((row[prev] == r) ||
        (abs(row[prev]-r) == abs(prev-c)))
    return false; // infeasible
  return true;
  }
</pre>
```

▲ロト ▲ □ ト ▲ 三 ト ▲ 三 ト つ Q ()

#### Using bitmasks

- Keep three bit-vectors.
- Shifting the bits handles the diagonals.

```
// Place a queen in row $r$
rows |= (1 << r);
up |= (1 << r);
down |= (1 << r);</pre>
```

// Moving to the next column...

up <<= 1; down >>= 1;