COP 3503 1/16/24

Backtracking

- Brute Force but cutting out of searches doomed to fail.

↳ list all possible items search space
evaluate each one, take the best

location visit 1,2,3,4 L/R move LLLL
1,2,4,3 LLLR
LLRL
⋮ ⋮
4,3,2,1 RRRR

Standard BF code

```
go(int* sol, int k, int n) {
    if(k==n) {
        evaluate
        return
    }
    for each x: next move {
        ⇒ sol[k]=x   // Can we skip some since they are doomed to fail?
            recursively call go(sol, k+1, n)
            // logic
    }
    return ___;
}
```

if(x is doomed to fail)
continue;

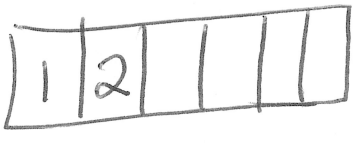
k divisibility

an integer of n digits a k divisible integer if each of its prefixes are divisible by their length.

5 digit k divisible #

12365

1 div 1
12 div 2
123 div 3
1236 div 4
12365 div 5

go( , 2, 9, 12)

int curval

```
void go(int* num, int k, int n, ) {
    if (k==n) {
        print(num, n);
        return;
    } // i is next digit to place
    for (int i=0; i<10; i++) {
        if ((10*curval + i) %(k+1) != 0) continue
        num[k] = i;
        go(num, k+1, n, 10*curval + i);
    }
}
```

BACKTRACKING

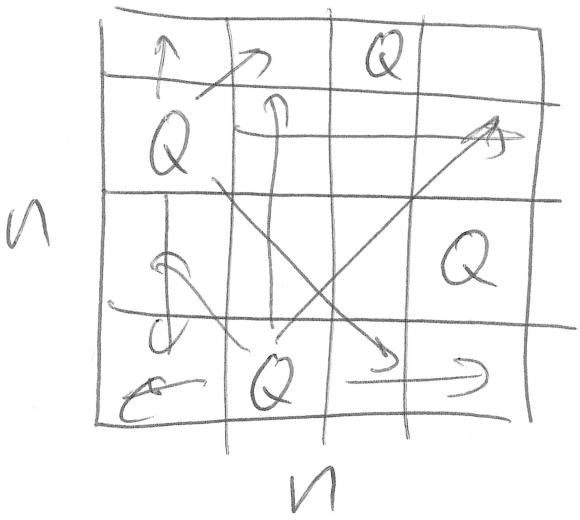
Runtime was O(10^digits) with

O(digits!) actually permutations

10×5×4×3

weird way better!

n Queens

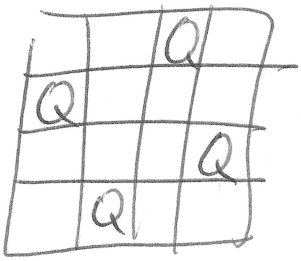


bad brute force

$\binom{n^2}{n}$ - Choose any n slots out of n^2 board eval

Q Q Q Q really bad

1Q 1Q 1Q 1Q



any valid solution is a permutation of rows

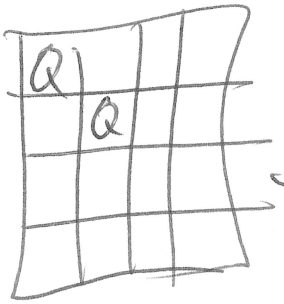
2, 4, 1, 3 → row 3 col 4

↓ row2 col1 | ↓ row4 col2 | row1 col3

gen $O(n!)$

run-time $O(n! \times n^2)$

↑ perm ↑ for double loop through queens



→ perm code run $(n-2)!$ times] These are doomed to fail!!!

Idea

Skip placing any queen that conflicts w/a previously placed queen!

```
                                    int[] used
int go(int[] perm, int k) {
    if (k == perm.length) {
        print(perm);
        return 1;
    }
    int res = 0;
    for (int i=0; i<perm.length; i++) {
        if (used[i]) continue;
        if (conflict(perm, k, i)) continue;
        perm[k] = i
        used[i] = true;
        res += go(perm, used, k+1);
        used[i] = false;
    }
    return res;
}
                                        col
boolean conflict(int[] perm, int k, int row) {
    for (int i = 0; i<col; i++)
        if (Math.abs(col-i) == Math.abs(perm[i]-row))
            return true;
    return false;
}
```

Magic Square

4 2	9	2
3	5	7
8	1	6

each # appears once

sum = 45

3 row disjoint

magic sum = $\frac{45}{3} = 15$

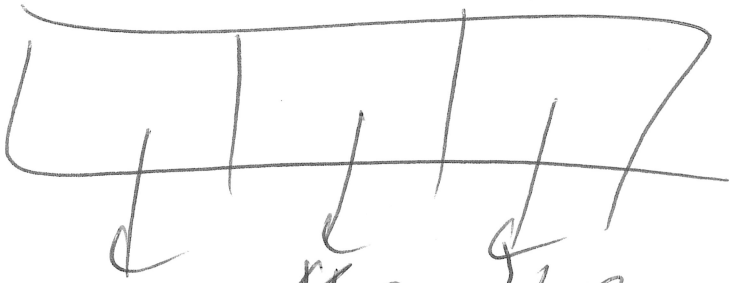
1, 2, X impossible backtrack

1, 3, X

1, 4, X

1, 5, 9

paintress



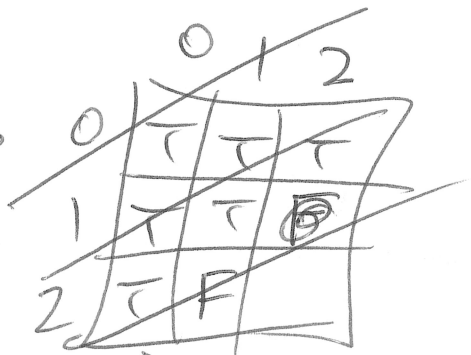
Map

green → 0

red → 1

blue → 2

red / blue conflict



conflict

F	F	F
F	F	T
F	T	F