Number Theory

Prime Numbers

- Is x a prime?
  - if it is even number, ..
  - else keep dividing
- factorization
  - base of security
  - Given N = P*Q
    - P,Q are unique
- How many are there?

```
prime_factorization(long x)
{
        long i;
        long c;

        c = x;
        while ((c % 2) == 0) {
                printf("%ld\n",2);
                c = c / 2;
        }

        i = 3;
        while (i <= (sqrt(c)+1)) {
                if ((c % i) == 0) {
                        printf("%ld\n",i);
                        c = c / i;
                }
                else
                        i = i + 2;
        }

        if (c > 1) printf("%ld\n",c);
}
```

GCD & LCM

If $a = bt + r$ for integers t and r, then $gcd(a,b) = gcd(b,r)$

$$
\begin{aligned}
gcd(34398, 2132) &= gcd(34398 \bmod 2132, 2132) = gcd(2132, 286) \\
gcd(2132, 286) &= gcd(2132 \bmod 286, 286) = gcd(286, 130) \\
gcd(286, 130) &= gcd(286 \bmod 130, 130) = gcd(130, 26) \\
gcd(130, 26) &= gcd(130 \bmod 26, 26) = gcd(26, 0)
\end{aligned}
$$

$$lcm(x,y) = xy/gcd(x,y)$$

Modular Arithmetic (Congruences)

$$(x + y) \text{ mo } ((x \bmod n) + (y \bmod n)) \bmod n$$

$$(12 \bmod 100) - (53 \bmod 100) = -41 \bmod 100 = 59 \bmod 100$$

$$xy \bmod n = (x \bmod n)(y \bmod n) \bmod n$$

$$x^y \bmod n = (x \bmod n)^y \bmod n$$

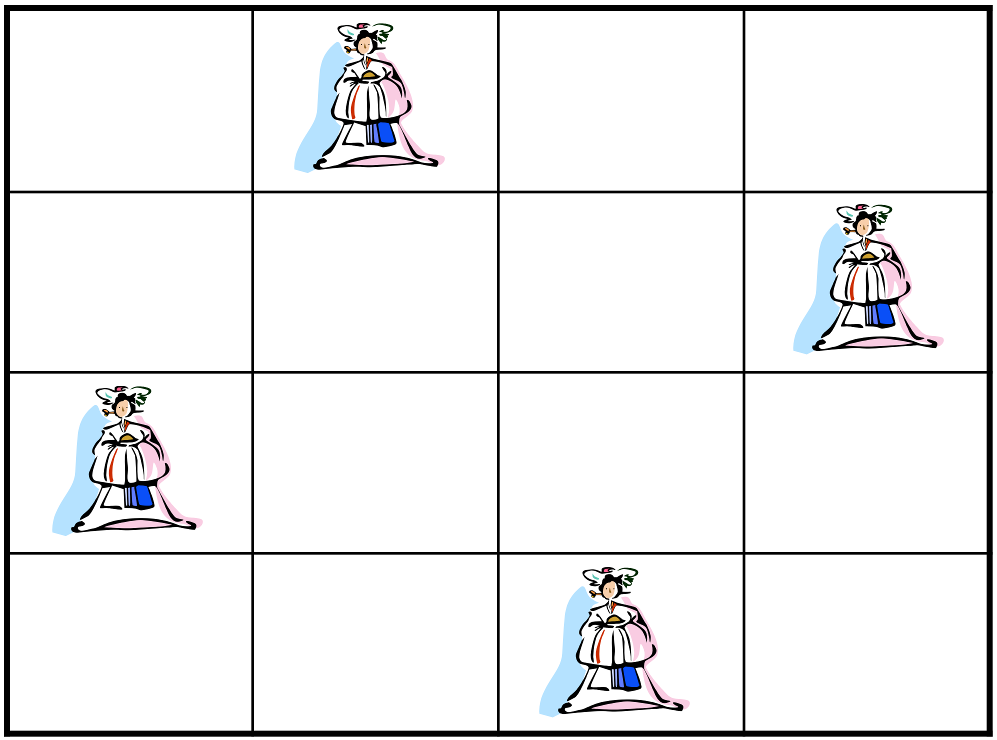
division?

Linear Congruence $ax \equiv b \pmod{n}$

Backtracking

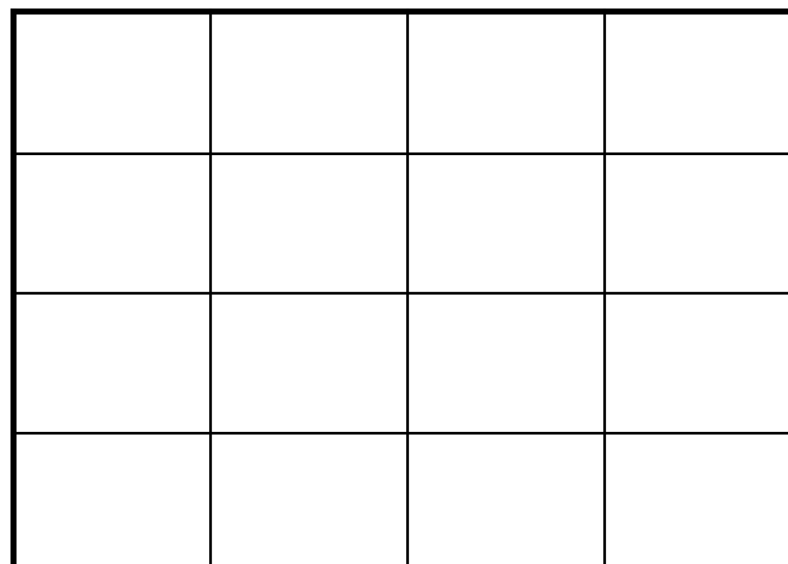
N-Queen Problem

4-Queens Problem



Solution for N-Queen

- no solution for $n < 4$
- $n=4$ case
  - list all case systematically
  - test each case if it is a solution
  - ${}_{16}C_4$ cases – ${}_{n^2}C_n$ cases
  - better way?



Backtracking

- Solution set
  - a vector a = $(a_1, a_2, \ldots, a_n)$

1. let a = $(a_1, \ldots, a_k)$
2. add a possible solution a_{k+1}
3. check validity
4. if it is a solution, do something
5. else check if a_{k+1} can generate more possible solutions
6. if yes, add them to the solution vector
7. else remove a_{k+1}

```
bool finished = FALSE;                  /* found all solutions yet? */

backtrack(int a[], int k, data input)
{
        int c[MAXCANDIDATES];           /* candidates for next position */
        int ncandidates;                /* next position candidate count */
        int i;                          /* counter */

        if (is_a_solution(a,k,input))
                process_solution(a,k,input);
        else {
                k = k+1;
                construct_candidates(a,k,input,c,&ncandidates);
                for (i=0; i<ncandidates; i++) {
                        a[k] = c[i];
                        backtrack(a,k,input);
                        if (finished) return;   /* terminate early */
                }
```

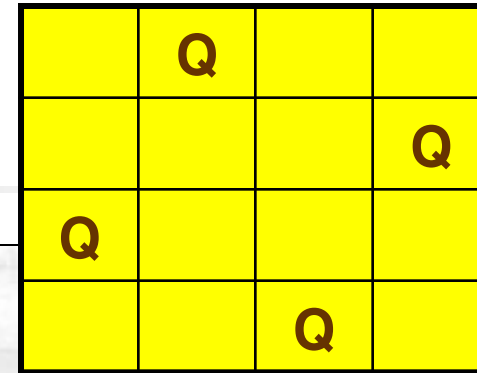
Backtracking Efficiency

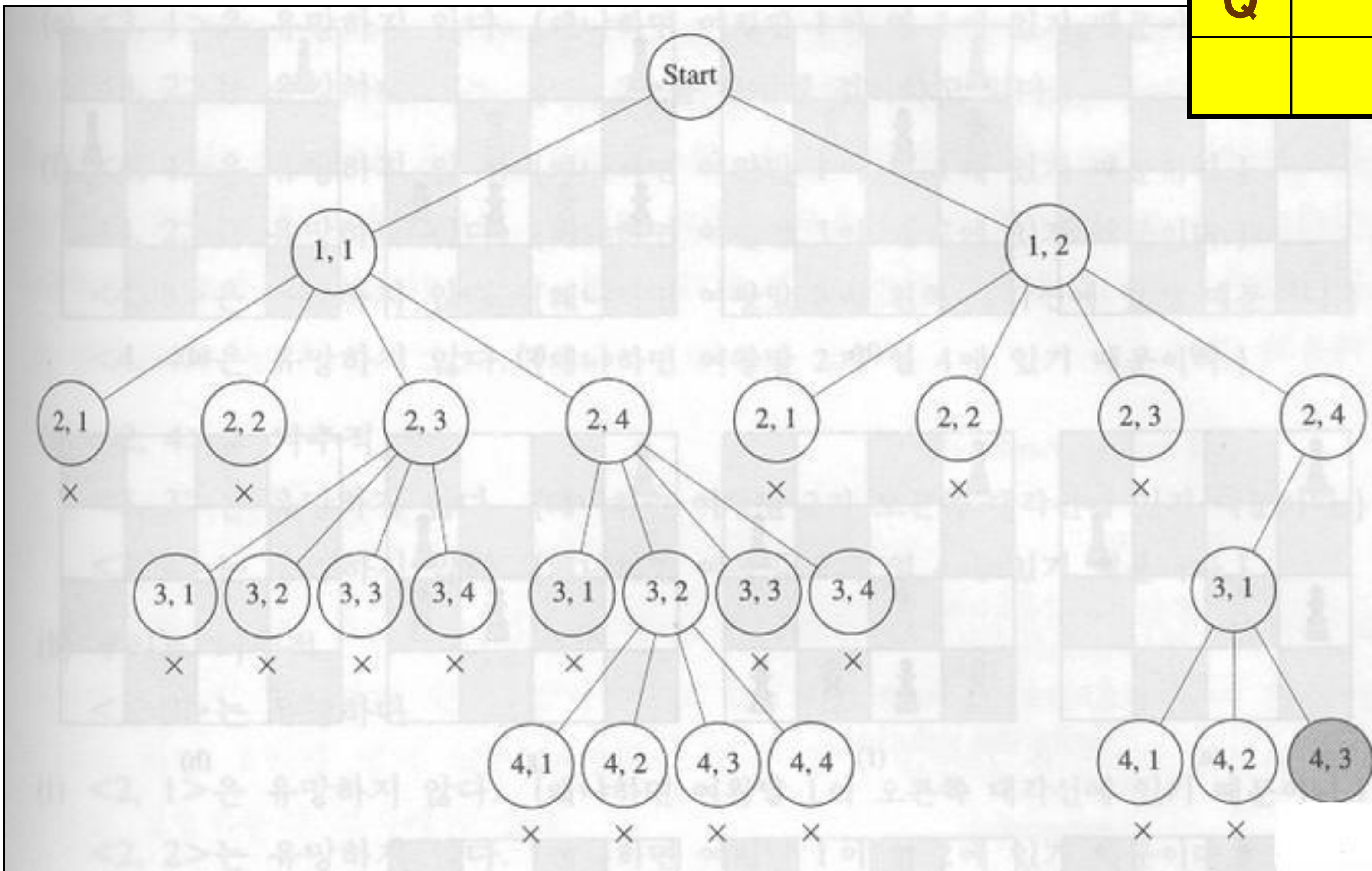
- It is usually correct
- The issue is efficiency
- solution space of the n-queens problem
  - there are n^2 cells
  - each cell either has a queen (TRUE) or not (FALSE)
  - total combinations 2^n
    - 8-queens problem ~ $1.84 * 10^{19}$
- another solution
  - the 1^{st} queen = 64 cases
  - the 2^{nd} queen = 64 cases
  - the 8^{th} queen = 64 cases
  - TOTAL $64^8 = 2.81 * 10^{14}$ cases

Prunning

- $6^{48} = 2.81 * 10^{14}$ is still a huge number
- remove invalid case as early as possible
  - no two queens sit on the same cell
  - once a queen is placed, the second will be at a higher numbered cell
    - ${}_{64}C_8 = 4.426 * 10^9$
  - can you do it more?
    - 1st queen at the 1st row
    - 2nd queen at the 2nd row
    - 8th queen at the 8th row
    - TOTAL $8^8 = 1.677 * 10^7$ cases
  - how about the column regulation?

Prunning for 4-Queens Problem





15-Puzzle


```
2 3 4 0
1 5 7 8
9 6 10 12
13 14 11 15
```

```
LLLDRDRDR
This puzzle is not solvable.
```

Tug of War

A tug of war is being arranged for the office picnic. The picnickers must be fairly divided into two teams. Every person must be on one team or the other, the number of people on the two teams must not differ by more than one, and the total weight of the people on each team should be as nearly equal as possible.

Sample Input

```
1

3
100
90
200
```

Sample Output

```
190 200
```