Brute Force Techniques
Why a Brute Force Approach?

- if problem is known to be NP-Hard and you need an optimal solution (and no pseudo polynomial algorithm exists)
- if (approximation) algorithms are too slow in practice or too complex to implement
- if search space of the input instances is known to be sufficiently small
Brute Force Approach

<table>
<thead>
<tr>
<th>Problem Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Finding encryption keys</td>
</tr>
<tr>
<td>- MiniMax (optimal strategy finding, e.g., chess)</td>
</tr>
<tr>
<td>- Prime Number Sieves (Eratosthenes, Atkin)</td>
</tr>
</tbody>
</table>
Brute Force Approach

**Speed-Up Techniques**

- Use a heuristic (for instance: greedy choice first)
- Backtracking (non-optimization problems)
- Branch and Bound (optimization problems)
- Avoid recalculation
- Precalculate parts of a solution and store it in a table
Backtracking

The Four-Queens Problem

Brute Force Techniques
void prime_sieve (int n = 100000) {
    ...
    int s = (int)sqrt((double)n) + 1;
    for (int i=2; i < n; i+=2 ) {
        prime_mark[i] = false;
        prime_mark[i+1] = true;
    }
    prime_mark[0] = prime_mark[1] = false;
    prime_mark[2] = true;
    for( int i=3; i<s; i+=2 )
        if (prime_mark[i]) {
            num_primes++;
            for (int j=3*i; j < n; j+=(2*i))
                prime_mark[j] = false;
        }
}