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 small (pre-computation)
Speed-Up Techniques

- use a heuristic (for instance: greedy choice first)
- branch and bound
- avoid recalculations
- precalculate part (or all) of the solution and store it in a table
Speed-Up Techniques

- use a heuristic (for instance: greedy choice first)
- branch and bound
- avoid recalculations
- precalculate part (or all) of the solution and store it in a table

Examples

- A*
- MiniMax
- Alpha-Beta-Search
- Prime-Number-Sieves
void prime_sieve (int n = 100000) {
    // ... stuff missing
    int s = (int)sqrt((double)n) + 1;
    for (int i=2; i < n; i+=2 ) { // what would be better?
        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 < in; j+=(2*i)) prime_mark[j] = false;
        }
}