Competitive Programming Training

Brute Force and Computational Complexity

Competitive Programming Training

Brute Force and ~~Computational Complexity~~

Brute Force

- Also called “**exhaustive search**”
- Go through **all possible cases** to look for solution
- Guaranteed to find solution if
  - Solution is in fact present
  - Algorithm / program is correct
- Disadvantage: slow!

A Classic Problem:

- You go to a farm with chickens for sale. The prices are listed as below:

Small	`$0.50`
Medium	`$1.00`
Large	`$5.00`

- List all the ways where you can buy exactly 100 chicken for exactly $100 dollars.

Solution

Try every combination of numbers of chicken to buy.
Let `i` be the number of **small chicken**, `j` be the number of **medium chicken**, and `k` be the number of **large chicken**.

$$i + j + k = 100$$

$$0.5i + j + 5k = 100$$

Solve for ordered triplets `(i, j, k)` where all 3 numbers are integers.

Solution

```
for (int i = 0; i <= 100; i++){
    for (int j = 0; j <= 100; j++){
        for (int k = 0; k <= 100; k++){
            if (i + j + k == 100 &&
                    i * 5 + 10 * j + 50 * k == 1000){

                System.out.println(i + " " + j + " " + k);
            }
        }
    }
}
```

Output:

```
0 100 0
8 91 1
16 82 2
24 73 3
32 64 4
40 55 5
48 46 6
56 37 7
64 28 8
72 19 9
80 10 10
88 1 11
```

```java
for (int i = 0; i <= 100; i++){
    for (int j = 0; j <= 100; j++){
        for (int k = 0; k <= 100; k++){
            if (i + j + k == 100 &&
                i * 5 + 10 * j + 50 * k == 1000){

                System.out.println(i + " " + j + " " + k);
            }
        }
    }
}
```

How many times is the highlighted part executed?

Computational Complexity

- ➔ Also called “time complexity”
- ➔ Refers to the relationship between input size and computational time.
- ➔ Used to estimate if the algorithm is going to finish within the time limit.
- ➔ The big-O notation is used to measure time complexity.

Common Time Complexity

Complexity	**Typical Algorithms**	**N for TL=1s**
O(1)	Adding 2 numbers	any N
O(log N)	Binary search in sorted array	a very big number
O(N)	Iterating through array	N < 10^8
O(N * log N)	`Arrays.sort()`	N < 3.9*10^7
O(N^2)	Bubble sort	N < 30000
O(2^N)	Binary state compression	N < 28
O(N!)	All permutations of array	N < 13

Practice Problems

List Minimum: (Iterating through array)

https://dmoj.ca/problem/bf1

Next Prime: (While-loop + prime checking)

https://dmoj.ca/problem/bf3

Roll the dice: (For-loop + simple arithmetics)

https://dmoj.ca/problem/ccc06j2