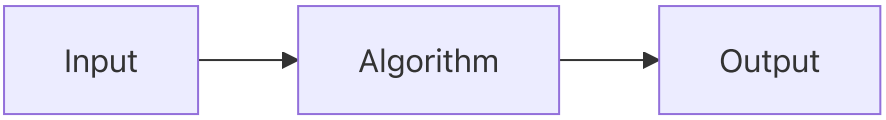
What is an Algorithm ?

A precise instruction based on computational operation, which takes some values as input and process them into output



Algorithm should be Correct and Efficient

- Correct
  - For any admissible instance, it must correctly produce desired output
- Efficient
  - Compute the output using reasonable resources

Difference Design aims to Difference Performance

An example of Algorithm:

```
1. Open `Obsidian`.
2. Write down lecture notes.
3. Enjoy writing / recalling knowledge.
```

Brute Force

A fundamental tools for solving several problems, however, brute force is usually slow.

Constrain Satisfaction Problem (CSP)

The problem must give the constrains that we have to satisfy. (Yes / No)

```
def fn(S, T)
    for each x in S
        if T(x)
            return x
```

- Let S be a set of candidate solutions.
- Let $T(x)$ be a function that test if a candidate solution x satisfies all constrains.

Constrain Optimization Problem (COP)

An extension of CSP by including an objective function in the problem.

The goal is not only to find a solution that satisfies all constrains, but the solution must give optimal value of the objective function.

```
def fn(S, T, O)
    best = INFINITY
    for each x in S
        if T(x) and O(x) < best
            best = O(x)
            best_answer = x
    return best_answer
```

- Let S be a set of candidate solutions.
- Let $T(x)$ be a function that test if a candidate solution x satisfies all constrains.
- Let $O(x)$ be an objective function.

Combination & Permutation

Permutation

A selection of members of the set.

```
void permutation(int index, int n, int k,
                 vector<bool> &check, vector<int> &v) {
    if (index == k) {
        for (int x: v) {
            cout << x << ' ';
        }
        cout << '\n';
    } else {
        for (int i=1; i<=n; i++) {
            if (check[i]) continue;
            check[i] = true;
            v[index] = i;
            permutation(index+1, n, k, check, v);
            check[i] = false;
        }
    }
}
```

Combination

An arrangement of sequence.

```
void combination(int index, int n, int k, int start, vector<int> &v) {
    if (index == k) {
        for (int x: v) {
            cout << x << ' ';
        }
        cout << '\n';
    } else {
        for (int i=start; i<=n; i++) {
            v[index] = i;
            combination(index+1, n, k, i+1, v);
```

```
        }
    }
}
```