Code Optimization
Definition

Code Optimization is a technique which tries to improve the code by eliminating unnecessary code lines and arranging the statements in such a sequence that speed up the program execution without wasting the resources.

Advantages:

- Executes faster
- Efficient memory usage
- Yields better performance
Techniques

- Dead Code Elimination
- Code Movement
- Strength Reduction
- Compile Time Evaluation
  - Constant Folding
  - Constant Propagation
  - Common subexpression elimination
( Compile Time Evaluation

1) Constant Folding:

It refers to a technique of evaluating the expressions whose operands are known to be constant at compile time itself.

Example: \( \text{length} = (2217) \times d \)

2) Constant Propagation:

In constant propagation, if a variable is assigned a constant value, then subsequent use of that variable can be replaced by a constant as long as no intervening assignment has changed the value of the variable.
Example:

\[ \beta l = 3.14 \]
\[ r = 5 \]

Area = \( \beta l \times r \times r \)

Here, the value of \( \beta l \) is replaced by 3.14 and \( r \) by 5, then computation of \( 3.14 \times r \times r \) is done during compilation.
Common sub-expression elimination:

The common sub-expression is an expression appearing repeatedly in the code which is computed previously. This technique replaces redundant expression each time it is encountered.

Example:

Before Optimization

\[
\begin{align*}
T_1 &= 4 \times i \\
T_2 &= a[T_1] \\
T_3 &= 4 \times j \\
T_4 &= 4 \times i \\
T_5 &= n \\
T_6 &= b[T_4] + T_5
\end{align*}
\]

After Optimization

\[
\begin{align*}
T_1 &= 4 \times i \\
T_2 &= a[T_1] \\
T_3 &= 4 \times j \\
T_5 &= n \\
T_6 &= b[T_4] + T_5
\end{align*}
\]
III Code Movement:

It is a technique of moving a block of code outside a loop if it won't have any difference if it is executed outside or inside the loop.

Example:

Before Optimization

```c
for (int i = 0; i < n; i++)
{
    x = y + z;
    a[i] = 6 * i;
}
```

After Optimization

```c
x = y + z;
for (int i = 0; i < n; i++)
{
    a[i] = 6 * i;
}
```
IV Dead Code Elimination:

Dead Code Elimination includes eliminating those code statements which are either never executed or unreachable or if executed their output is never used.

Example:

```plaintext
i = 0
if ( i == 1 )
    a = x + 5;

Before Optimization

After Optimization
```
**Strength Reduction:**

It is the replacement of expressions that are expensive with cheaper and simpler ones.

**Example:**

\[ B = A \times 2 \quad \text{Before Optimization} \]
\[ B = A + A \quad \text{After Optimization} \]