Code Optimization

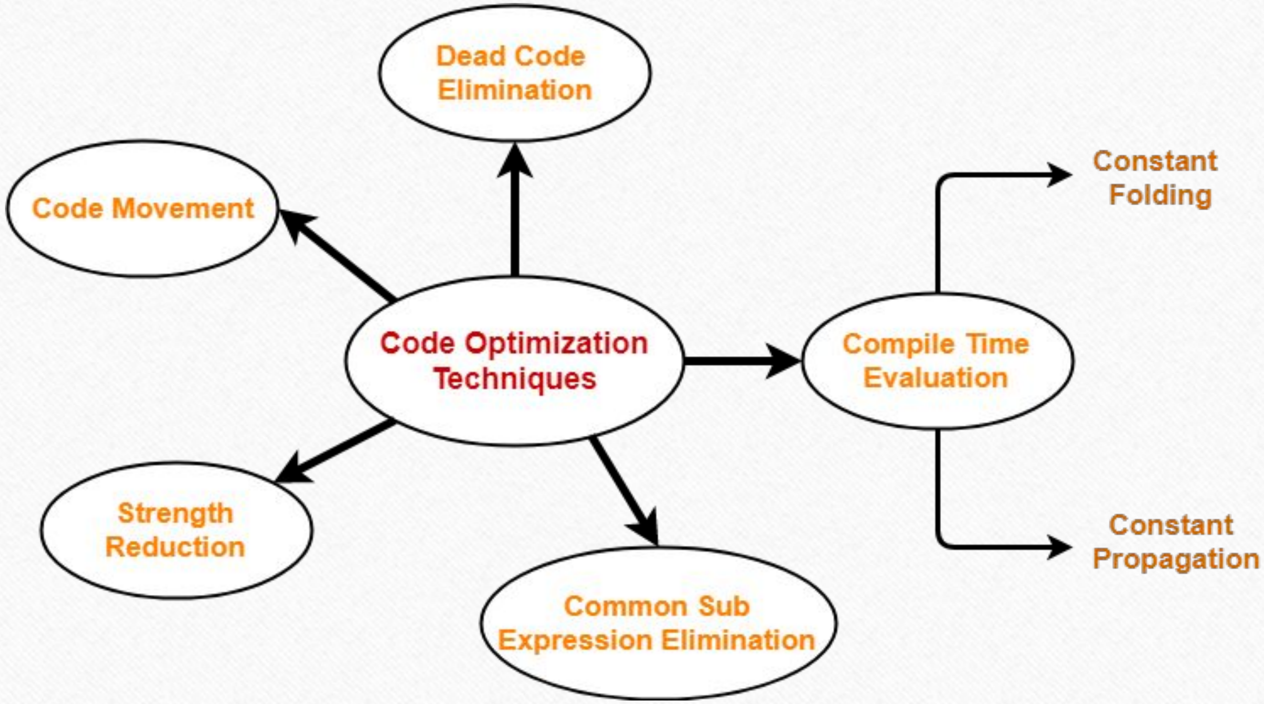
Code Optimization

- Eliminating the unwanted code lines
- Rearranging the statements of the code

Advantages

- Optimized code has faster execution speed.
- Optimized code utilizes the memory efficiently.
- Optimized code gives better performance.

Techniques


Dead Code Elimination
Code Movement
Code Optimization Techniques
Compile Time Evaluation
Constant Folding
Constant Propagation
Strength Reduction
Common Sub Expression Elimination

Compile Time Evaluation

1. **Constant Folding**

- As the name suggests, it involves folding the constants.
- The expressions that contain the operands having constant values at compile time are evaluated.
- Those expressions are then replaced with their respective results

Compile Time Evaluation (Cont.)

Constant Folding example:

Circumference of Circle = (22/7) x Diameter

- Here,
- This technique evaluates the expression 22/7 at compile time.
- The expression then is replaced with its result 3.14
- This saves the time at run time

Compile Time Evaluation (Cont.)

2. **Constant Propagation**

- If some variable has been assigned some constant value, then it replaces that variable with its constant value in the further program during compilation.
- The condition is that the value of variable must not get alter in between.

Compile Time Evaluation (Cont.)

Constant Propagation Example

pi = 3.14

radius = 10

Area of circle = pi x radius x radius

- This technique substitutes the value of variables ‘pi’ and ‘radius’ at compile time.
- It then evaluates the expression 3.14 x 10 x 10.
- The expression is then replaced with its result 314.
- This saves the time at run time.

Common Sub-Expression Elimination

The expression that has been already computed before and appears again in the code for computation is called as Common Sub-Expression.

Common Sub-Expression Elimination (Cont.)

In this technique,

- As the name suggests, it involves eliminating the common sub expressions.
- The redundant expressions are eliminated to avoid their re-computation.
- The already computed result is used in the further program when required.

Common Sub-Expression Elimination (Cont.)

- Example

Before Optimization	After Optimization
S1 = 4 x i	S1 = 4 x i
S2 = a[S1]	S2 = a[S1]
S3 = 4 x j	S3 = 4 x j
S4 = 4 x i	S4 = S1
S5 = n	S5 = n
S6 = b[S4] + S5	S6 = b[S4] + S5

Common Sub-Expression Elimination (Cont.)

- Remember, if some expression is used more than once, you can save that expression in a temporary variable and then use it.

Before Optimization	After Optimization
A=B+C+D	X=B+C
E=F+G	A=X+D
H=B+C+I	E=F+G
	H=X+I

Code Movement

In this technique,

- As the name suggests, it involves movement of the code.
- The code present inside the loop is moved out if it does not matter whether it is present inside or outside.
- Such a code unnecessarily gets execute again and again with each iteration of the loop.
- This leads to the wastage of time at run time.

Code Movement (Cont.)

- Example

Before Optimization	After Optimization

for (int j = 0 ; j < n ; j ++)
{
x = y + z ;
a[j] = 6 x j;
}
	x = y + z ;
	for (int j = 0 ; j < n ; j ++)
	{
	a[j] = 6 x j;
	}

Dead Code Elimination

In this technique,

- As the name suggests, it involves eliminating the dead code.
- The statements of the code which either never executes or are unreachable or their output is never used are eliminated.

Dead Code Elimination (Cont.)

- Example

Before Optimization	After Optimization

i = 0 ;
if (i == 1)
{
a = x + 5 ;
}
	i = 0 ;

Strength Reduction

In this technique,

- As the name suggests, it involves reducing the strength of expressions.
- This technique replaces the expensive and costly operators with the simple and cheaper ones.

Strength Reduction (Cont.)

- **Example**

Before Optimization	After Optimization
B = A x 2	**B=A+A**

Here,

- The expression “A x 2” is replaced with the expression “A + A”.
- This is because the cost of multiplication operator is higher than that of addition operator.

Problem 1

Before Optimization	After Optimization

i = 1;
y = x + 5;
if (i == 0)
{
a = x + 5 ;
}
	i = 1;
	y = x + 5;
	
	Explanation: Because the integer "I" will never be 0, so that loop will never be executed. This is an example of dead code elimination

Problem 2

Before Optimization	After Optimization

i = 1;
y = x + 5;
if (i == 1)
{
a = x + 5 ;
}
	i = 1;
	y = x + 5;
	if (i == 1)
	{
	a = y;
	}
	
	Explanation: since the subexpression x+5 is used more than once, we replaced x+5 with y in the code.