什么是local optimization,什么是loop optimization,什么又是global optimization这三者之间有什么异同?

在编译器优化中，**Local Optimization（局部优化）**、**Loop Optimization（循环优化）**和**Global Optimization（全局优化）**是三种不同范围的代码优化技术，它们的异同如下：

---

## 1. Local Optimization（局部优化）

**定义**：针对代码中的**基本块（Basic Block）**进行的优化。基本块是连续的语句序列，没有分支进入或退出（除入口和出口外）。

**范围**：仅限于单个基本块内部。

**特点**：

无需分析控制流或跨块数据流。

执行速度快，适合在编译早期阶段应用。

**常见优化**：

常量传播（Constant Propagation）

公共子表达式消除（Common Subexpression Elimination, CSE）

死代码删除（Dead Code Elimination）

代数简化（如 `x = y + 0` → `x = y`）。

---

## 2. Loop Optimization（循环优化）

**定义**：专门针对**循环结构**的优化，目标是减少循环开销或提高循环体内的执行效率。

**范围**：跨越循环的多个基本块，但仅关注循环相关的代码。

**特点**：

需要分析循环的控制流（如循环条件、迭代变量）。

对性能影响显著（循环通常是热点代码）。

**常见优化**：

循环不变代码外提（Loop Invariant Code Motion）

归纳变量优化（Induction Variable Optimization）

循环展开（Loop Unrolling）

循环融合（Loop Fusion）或分裂（Loop Fission）。

---

### 3. Global Optimization（全局优化）

**定义**：在**整个函数或程序**范围内进行的优化，跨越多个基本块和分支。

**范围**：分析整个控制流图（CFG）和数据流。

**特点**：

需要全局数据流分析（如活跃变量分析、到达定义分析）。

开销较大，但优化效果更显著。

**常见优化**：

全局常量传播（Global Constant Propagation）

跨基本块的公共子表达式消除

函数内联（Function Inlining）

全局寄存器分配。

---

### 三者的异同

| 特性 | Local Optimization | Loop Optimization | Global Optimization |
| --- | --- | --- | --- |
| **范围** | 单个基本块 | 循环结构内部 | 整个函数/程序 |
| **分析复杂度** | 低（无需控制流分析） | 中等（需分析循环结构） | 高（需全局数据流分析） |
| **优化目标** | 语句级效率 | 减少循环开销 | 跨块资源利用 |
| **依赖信息** | 局部数据流 | 循环变量、迭代次数 | 控制流图、全局数据流 |
| **示例** | 删除冗余赋值 | 循环展开 | 函数内联 |

---

## 关键区别

**作用域**：

局部优化局限于基本块，全局优化覆盖函数/程序，循环优化介于两者之间。

**分析成本**：

局部优化成本最低，全局优化最高（需跨分支分析）。

**适用阶段**：

局部优化常发生在编译早期（如语法树生成后），全局优化在中间代码生成后，循环优化可能单独处理。

---

## 协作关系

优化通常分层进行：先局部优化清理代码，再处理循环（热点代码），最后全局优化整合资源。

例如：循环内的公共子表达式可能先被局部优化识别，再通过循环优化外提不变表达式，最后由全局优化跨循环传播常量。

通过结合这三种优化，编译器能在不同层次上提升代码性能。