一. `calculateBT` 函数实现

1. 函数作用

- `calculateBT` 用于遍历函数的基本块和四元式指令，分析每个块是否可达（可执行），并推断每个变量的常量传播信息（单值/多值/无值）
- 该过程是常量传播和可达性分析的核心，便于后续优化（如删除无用代码、替换常量等）

---

2. 主要流程

- 外层循环遍历所有基本块，跳过不可达块
- 内层循环遍历块内所有四元式指令，根据不同指令类型进行处理
- 若分析过程中有新信息产生（如某块变为可达、某变量值发生变化），则跳转回头重新分析，直到收敛

---

3. 关键实现细节

3.1 标签指令（LABEL）

- 若当前块可达且只有一个后继块，则将后继块标记为可达

3.2 条件跳转（CJUMP）

- 若左右操作数均为常量或单值变量，则直接计算条件结果，分别将真/假分支标记为可达
- 若任一操作数为多值，则两个分支都标记为可达

3.3 赋值（MOVE）

- 若源操作数为常量，则目标变量获得单值
- 若源操作数为变量，则目标变量获得与源变量相同的值类型

3.4 二元操作赋值（MOVE_BINOP）

- 若两个操作数均为常量或单值变量，则目标变量获得计算结果的单值
- 若任一操作数为多值，则目标变量为多值

3.5 Phi函数（PHI）

- 若有任一参数为多值且对应前驱块可达，则目标变量为多值
- 若所有可达前驱参数均为单值且值相同，则目标变量为该单值；若值不同，则为多值

3.6 读内存/函数调用

- 读内存、类方法调用、外部函数调用等均将目标变量标记为多值

4. 迭代与收敛

- 只要有新的可达块或变量值发生变化，立即跳回重新分析，保证信息充分传播，直到所有信息收敛

二. `modifyFunc` 函数实现

1. 函数作用

- `modifyFunc` 用于根据常量传播和可达性分析的结果，对函数的四元式中间代码进行优化和重写
- 主要目标是：**删除无用代码、替换常量、消除不可达块、简化控制流**

---

2. 主要流程

2.1 初始化

- 创建 `Temp_map`，用于新建临时变量和标签
- 遍历所有基本块和块内所有语句

2.2 删除不可达块

- 如果某个基本块不可达，直接从块列表中移除

2.3 删除无用赋值语句

- 对于赋值（`MOVE`）、二元操作赋值（`MOVE_BINOP`）、Phi函数（`PHI`）：
  - 如果目标变量是单值（常量传播结果），则删除该语句

2.4 优化Phi函数参数

- 对于 `PHI` 语句的每个参数：
  - 如果参数是单值变量，则在前驱块插入一条常量赋值语句，并将参数重命名为新变量

2.5 替换单值变量为常量

- 对每条语句的所有使用变量：
  - 如果该变量是单值，则将其在语句中替换为常量

2.6 条件跳转优化

- 对于条件跳转（`CJUMP`）：
  - 如果左右操作数均为常量，则直接计算条件结果，将 `CJUMP` 替换为无条件跳转（`JUMP`），并更新出口标签

2.7 更新计数

- 最后，更新函数的最大临时变量编号和最大标签编号

---

3. 关键实现细节

- **循环与goto**：每次有代码结构发生变化（如删除语句、块），立即跳回重新遍历，保证所有优化都能被及时应用
- **常量传播与替换**：利用 `getRtValue` 获取变量的传播结果，进行常量替换和死代码删除
- **Phi参数特殊处理**：单值参数需要在前驱块插入赋值，保证SSA形式正确
- **控制流简化**：将恒定条件的 `CJUMP` 转换为 `JUMP`，简化控制流

三. Git Graph