



南京理工大学

# 编译原理

项欣光

计算机科学与工程学院





## 第10章 代码优化

- 优化技术
- 局部优化
- 控制流分析和循环优化
- 数据流分析和全局优化（自学）



## 10.1 优化技术简介

- 代码优化
- 常用的优化技术
- 举例



# 一、代码优化

- 对中间代码或目标代码进行**等价变换**，使得变换后的代码与变换前的代码运行结果相同，时空效率提高（运行速度快，占用内存少）。
- 分类：
  - **局部优化**：在只有一个入口、一个出口的基本程序块上进行的优化
  - **循环优化**：对循环中的代码进行的优化
  - **全局优化**：在整个程序范围内进行的优化



## 二、常用的优化技术

- 1) 删除多余运算（删除公共子表达式）
- 2) 循环不变代码外提：减少循环中的代码总数
- 3) 强度削弱：把强度大的运算换成强度小的
- 4) 变换循环控制条件
- 5) 合并已知量
- 6) 复写传播
- 7) 删除无用赋值



### 三、举例

(1)  $P := 0$

(2)  $I := 1$

(3)  $T1 := 4 * I$

(4)  $T2 := \text{addr}(A) - 4$

(5)  $T3 := T2[T1]$

(6)  $T4 := 4 * I$

(7)  $T5 := \text{addr}(B) - 4$

(8)  $T6 := T5[T4]$

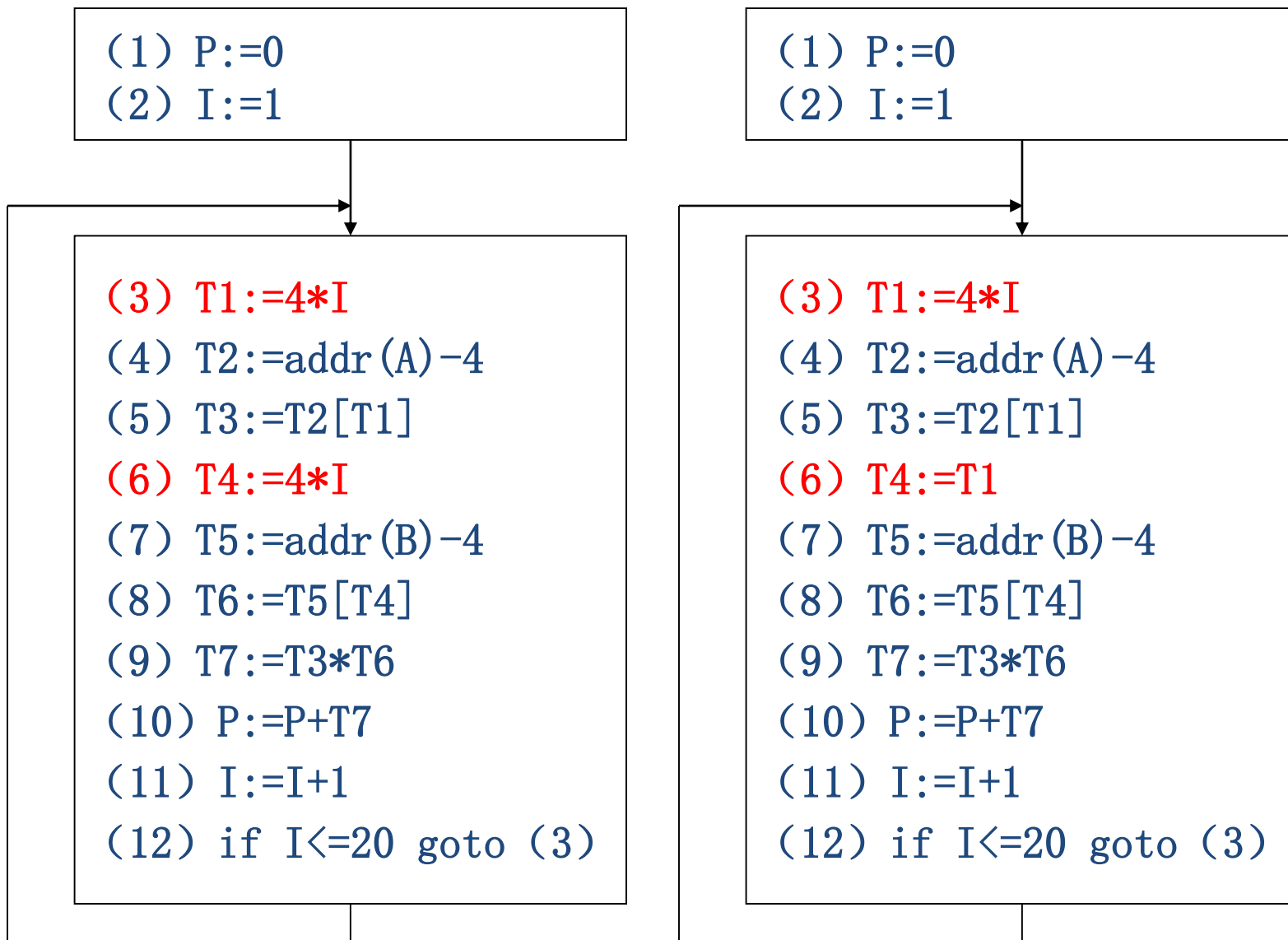
(9)  $T7 := T3 * T6$

(10)  $P := P + T7$

(11)  $I := I + 1$

(12) if  $I \leq 20$  goto (3)

# 删除多余运算（删除公共子表达式）



# 循环不变代码外提

(1)  $P:=0$   
(2)  $I:=1$

(3)  $T1:=4*I$   
(4)  $T2:=\text{addr}(A)-4$   
(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(7)  $T5:=\text{addr}(B)-4$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(12) if  $I \leq 20$  goto (3)

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$

(3)  $T1:=4*I$   
(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(12) if  $I \leq 20$  goto (3)



# 强度削弱

```
(1) P:=0  
(2) I:=1  
(4) T2:=addr(A)-4  
(7) T5:=addr(B)-4
```

```
(3) T1:=4*I  
(5) T3:=T2[T1]  
(6) T4:=T1  
(8) T6:=T5[T4]  
(9) T7:=T3*T6  
(10) P:=P+T7  
(11) I:=I+1  
(12) if I<=20 goto (3)
```

```
(1) P:=0  
(2) I:=1  
(4) T2:=addr(A)-4  
(7) T5:=addr(B)-4
```

```
(3) T1:=4*I
```

```
(5) T3:=T2[T1]  
(6) T4:=T1  
(8) T6:=T5[T4]  
(9) T7:=T3*T6  
(10) P:=P+T7  
(11) I:=I+1  
(3') T1:=T1+4  
(12) if I<=20 goto (5)
```

# 变换循环控制条件

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4*I$

(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(3')  $T1:=T1+4$   
(12) if  $I \leq 20$  goto (5)

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4*I$

(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(3')  $T1:=T1+4$   
(12) if  $T1 \leq 80$  goto (5)

# 合并已知量

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4*I$

(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(3')  $T1:=T1+4$   
(12) if  $T1 \leq 80$  goto (5)

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4$

(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T4]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(3')  $T1:=T1+4$   
(12) if  $T1 \leq 80$  goto (5)

# 复写传播

(1)  $P := 0$   
(2)  $I := 1$   
(4)  $T2 := \text{addr}(A) - 4$   
(7)  $T5 := \text{addr}(B) - 4$   
(3)  $T1 := 4$

(5)  $T3 := T2[T1]$   
(6)  $T4 := T1$   
(8)  $T6 := T5[T4]$   
(9)  $T7 := T3 * T6$   
(10)  $P := P + T7$   
(11)  $I := I + 1$   
(3')  $T1 := T1 + 4$   
(12) if  $T1 \leq 80$  goto (5)

(1)  $P := 0$   
(2)  $I := 1$   
(4)  $T2 := \text{addr}(A) - 4$   
(7)  $T5 := \text{addr}(B) - 4$   
(3)  $T1 := 4$

(5)  $T3 := T2[T1]$   
(6)  $T4 := T1$   
(8)  $T6 := T5[T1]$   
(9)  $T7 := T3 * T6$   
(10)  $P := P + T7$   
(11)  $I := I + 1$   
(3')  $T1 := T1 + 4$   
(12) if  $T1 \leq 80$  goto (5)

# 删除无用赋值

(1)  $P:=0$   
(2)  $I:=1$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4$

(5)  $T3:=T2[T1]$   
(6)  $T4:=T1$   
(8)  $T6:=T5[T1]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(11)  $I:=I+1$   
(3')  $T1:=T1+4$   
(12) if  $T1 \leq 80$  goto (5)

(1)  $P:=0$   
(4)  $T2:=\text{addr}(A)-4$   
(7)  $T5:=\text{addr}(B)-4$   
(3)  $T1:=4$

(5)  $T3:=T2[T1]$   
(8)  $T6:=T5[T1]$   
(9)  $T7:=T3*T6$   
(10)  $P:=P+T7$   
(3')  $T1:=T1+4$   
(12) if  $T1 \leq 80$  goto (5)



## 10.2 局部优化

局部优化：基本块内的优化。

基本块：程序中一顺序执行的语句序列，其中只有一个入口语句和一个出口语句。

对一给定的程序，我们可以将它划分为一系列的基本块，在各基本块的范围内分别进行优化。

本节主要内容：

- 一、基本块的划分
- 二、基本块的变换
- 三、基本块的DAG表示
- 四、DAG的应用



# 一、基本块的划分

- 基本块的入口语句：
  - 1) 程序的第一个语句
  - 2) 条件转移语句或无条件转移语句的转移目标语句
  - 3) 紧跟在条件转移语句后面的语句
- 基本块的划分算法：
  - 1) 求出四元式序列的各基本块入口语句；
  - 2) 对每个入口语句，构造其所属的基本块：由本入口语句开始到其下一入口语句（不含下一入口语句）为止，或到一转移或者停止语句（含该转移或者停止语句）为止，所有的语句序列；
  - 3) 凡未被纳入到某一基本块的语句，都是控制流程无法到达的语句，将其删除。



# 一、基本块的划分

- 举例：

(1) read (C)

(2) A:= 0

(3) B:= 1

(4) L1:A:=A + B

(5) if B>= C goto L2

(6) B:=B+1

(7) goto L1

(8) L2: write (A)

(9) halt

**B1**

**B2**

**B3**

**B4**





## 二、基本块的变换

基本块的两类等价变换：

- 保结构的变换
  - 1) 删除公共子表达式
  - 2) 删除无用赋值
  - 3) 重新命名临时变量
    - $T := A + B$ （其中 $T$ 为临时变量）改为：
    - $U := A + B$ （ $U$ 是新的临时变量，并把对 $T$ 的所有引用改为 $U$ ）
  - 4) 交换语句次序
    - 两相邻语句： $T1 := A + B$       （其中， $A, B$ 不是 $T2$ ）  
    $T2 := C + D$       （其中， $C, D$ 不是 $T1$ ）
    - 那么，可以交换这两个相邻语句。



## 二、基本块的变换

基本块的两类等价变换：

- 代数变换

- 1) 化简表达式

例如：可以删除形如  $x := x + 0$  或  $x := x * 1$  的表达式

- 2) 强度削弱

用较快的运算代替较慢的运算

例如：可以将  $x := y ** 2$  的指数运算（通常由函数实现），变换成  $x := y * y$



### 三、基本块的DAG表示

DAG（无环有向图），可以利用DAG图来优化基本块。

相关概念：

- **前驱**：有向图中任一有向边 $n_i \rightarrow n_j$ ， $n_i$ 是 $n_j$ 的前驱
- **后继**：有向图中任一有向边 $n_i \rightarrow n_j$ ， $n_j$ 是 $n_i$ 的后继
- **通路**：有向图中，若存在任一有向边序列  $n_1 \rightarrow n_2$ ,  $n_2 \rightarrow n_3, \dots, n_i \rightarrow n_j$ ，则从结点 $n_1$ 到 $n_j$ 存在一条通路
- **环路**：如果 $n_1$ 到 $n_j$ 的通路中 $n_1 = n_j$ ，则该通路为环路
- **DAG（无环路有向图）**：有向图中的任意一条通路都不是环路。
- **祖先、后代**：在DAG中，如果  $(n_1, n_2, \dots, n_k)$  是其中的一条通路，则称结点 $n_1$ 是 $n_k$ 的祖先， $n_k$ 是 $n_1$ 的后代。



## DAG结点带有的标记（附加信息）

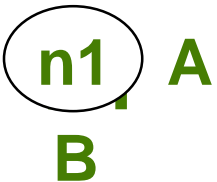
- **叶结点**（无后继）：附加独特的标识符、常数、变量地址标记。
- **内部结点**（有后继）：运算符标记。代表应用该运算符对其后继结点的值进行运算的结果。
- **各个结点**：可能附加多个标识符标记。表示这些变量具有该结点所代表的值。

注：上述的这个DAG可以用来描述计算过程，因此又称为描述计算过程的DAG。

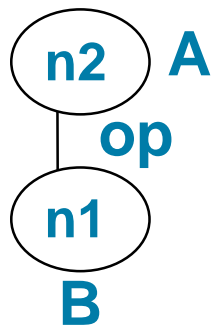
一个基本块可以用一个DAG表示



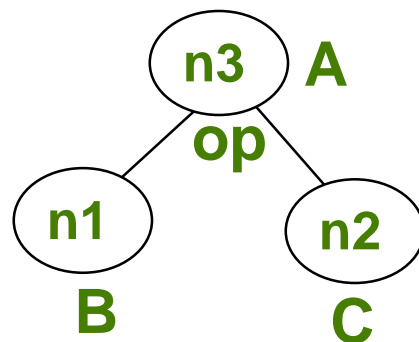
## 基本块的DAG表示与构造 P254

0型  $A := B$  

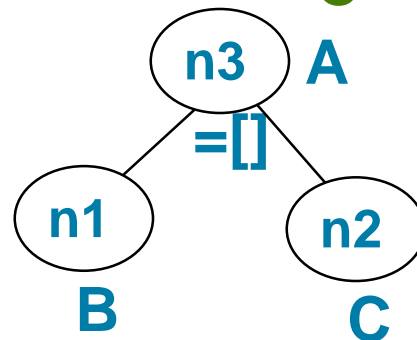
1型  $A := \text{op } B$



2型  $A := B \text{ op } C$



2型  $A := B[C]$





## 基本块DAG的构造算法 P254, 255

只讨论0、1、2型四元式的基本块的DAG构造算法：

首先DAG为空，再对基本块的每个四元式依次执行：（**A** 运算结果，**B** 运算对象1，**C** 运算对象2）

1、(1) 如果NODE(**B**) 无定义，则构造一个标记为B的叶结点，并定义NODE(**B**) 为这个结点；

(NODE(A) 是描述标识符与DAG结点对应关系的一个函数，它的值要么是一个结点的编号n（A是其上的标记或附加标记符），要么无定义。）

(2) 如果当前四元式是0型，则记NODE(**B**) 的值为n，转4

(3) 如果当前四元式是1型，则转2(1)

(4) 如果当前四元式是2型，[如果NODE(**C**) 无定义，则构造一标记为C的叶结点，并定义NODE(**C**) 为这个结点]，转2(2)



## 基本块DAG的构造算法 P254, 255

- 2、
  - (1) 如果NODE(B)是常数叶结点，则转2(3)；否则转3(1)
  - (2) 如果NODE(B)和NODE(C)都是常数叶结点，则转2(4)；否则转3(2)
  - (3) 执行op B（合并已知量），令得到的新常数为P。  
若NODE(B)是当前四元式新构造的结点，则删除它；  
若NODE(P)无定义，则构造一叶结点n，P做标记。置NODE(P)=n, 转4
  - (4) 执行 B op C（合并已知量），令新常数为P  
若NODE(B)或NODE(C)是处理当前四元式时新构造出的结点，则删除；  
若NODE(P)无定义，则构造一叶结点n，P做标记。置NODE(P)=n, 转4





## 基本块DAG的构造算法 P254, 255

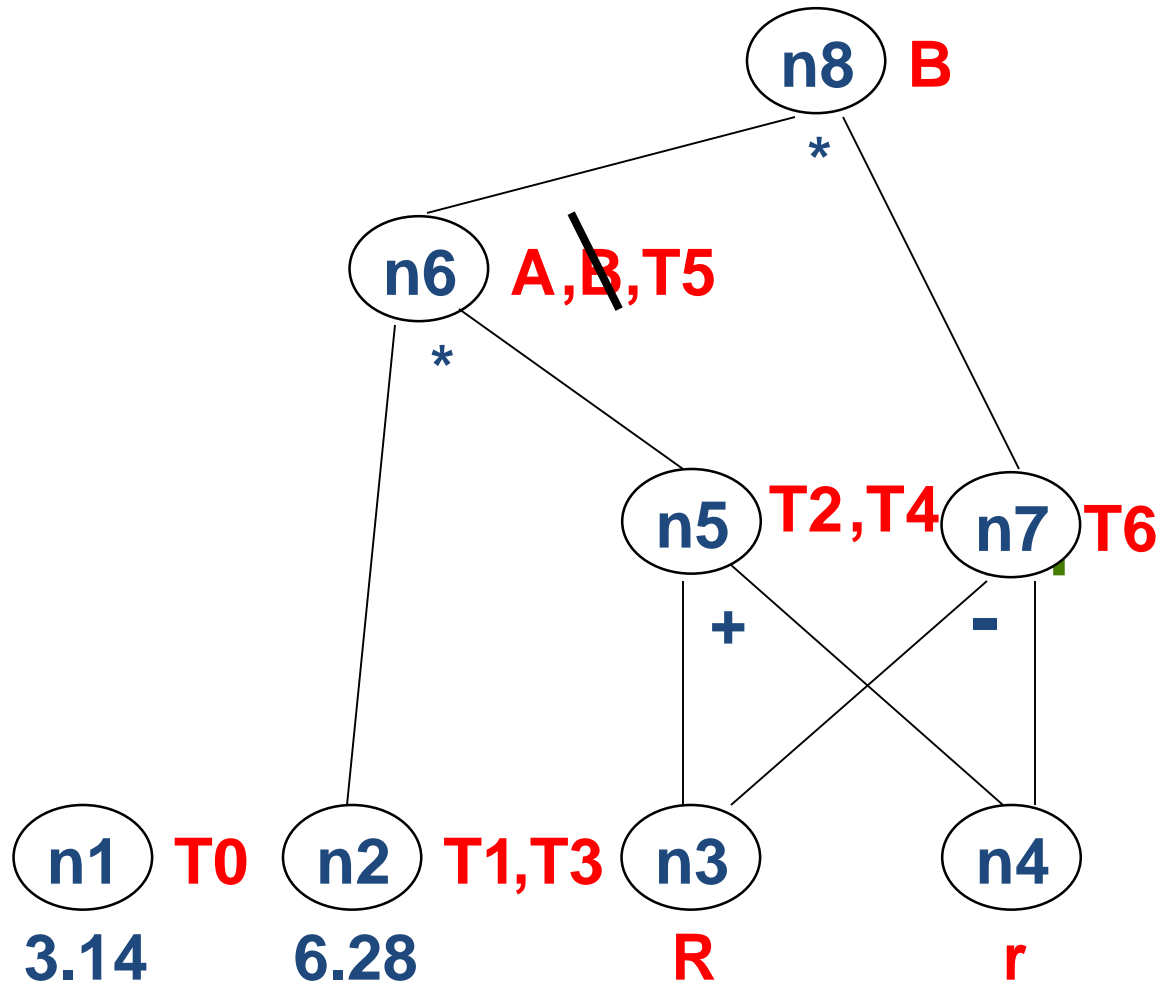
- 3、(1) 检查DAG中是否已有结点，其唯一后继为NODE(B)，且标记为op(找公共子表达式)。若没有，则构造该结点n，否则把已有的结点作为它的结点，并设该结点为n，转4。  
(2) 检查DAG中是否已有结点，其左后继为NODE(B)，右后继为NODE(C)，且标记为op(找公共子表达式)。若没有，则构造该结点n，否则把已有的结点作为它的结点，并设该结点为n。转4。
- 4、若NODE(A)无定义，则把A附加在结点n上，并令NODE(A)=n；  
否则，先把A从NODE(A)结点上的附加标识符集中删除，（若NODE(A)为叶结点，就不删除），把A附加到新结点n上，并令NODE(A)=n。  
转向处理下一四元式。





# 基本块DAG的构造 举例

- (1)  $T0 := 3.14$
- (2)  $T1 := 2 * T0$
- (3)  $T2 := R + r$
- (4)  $A := T1 * T2$
- (5)  $B := A$
- (6)  $T3 := 2 * T0$
- (7)  $T4 := R + r$
- (8)  $T5 := T3 * T4$
- (9)  $T6 := R - r$
- (10)  $B := T5 * T6$





## 四、DAG应用

利用DAG进行优化：

- 在构造DAG时，已经作了一些优化工作。

### 1) 合并已知量

步骤2：运算对象是编译时的已知量，并不生成计算该节点值的内部结点，而是执行该运算，将计算结果生成一个叶节点。

### 2) 删除多余运算

步骤3：检查公共子表达式，对所有含有公共子表达式的四元式，只产生一个计算该表达式值得内部结点，那些被赋值的变量标识符附加到该结点上。

### 3) 删除无用赋值

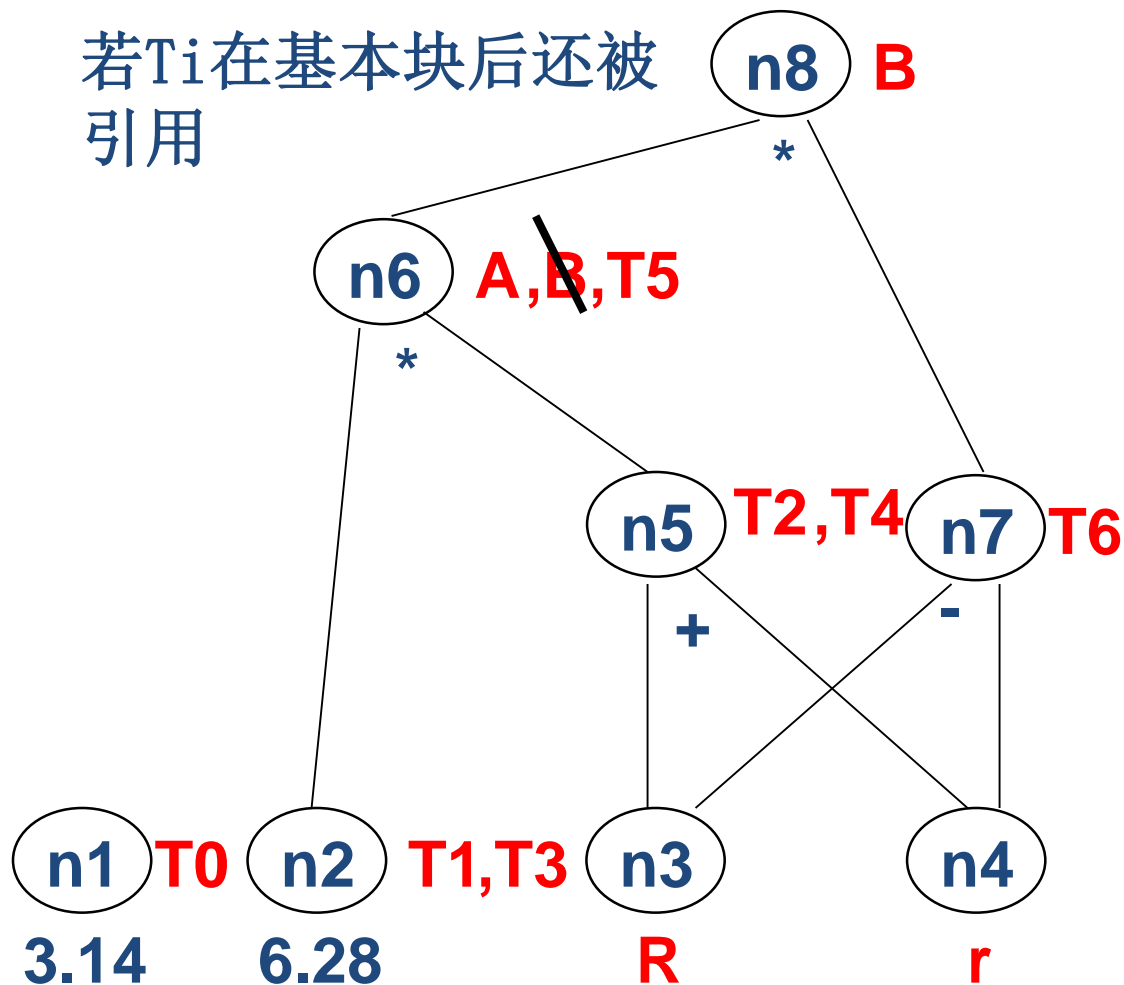
步骤 4：变量赋值后在被引用之前重新赋值，从前一值结点上删除

- 通过DAG重构优化后的代码序列



## 利用DAG进行优化

若 $T_i$ 在基本块后还被引用



重构四元式序列

- $T_0 := 3.14$
- $T_1 := 6.28$
- $T_3 := T_1$
- $T_2 := R + r$
- $T_4 := T_2$
- $A := 6.28 * T_2$
- $T_5 := A$
- $T_6 := R - r$
- $B := A * T_6$



## 利用DAG进行优化

- 从基本块的DAG中得到的其它优化信息：
  - 1) 在基本块外被定值，并在基本块内被引用的所有标识符。

叶子结点上标记的标识符

- 2) 在基本块内被定值，并该值能在基本块之后被引用的所有标识符。

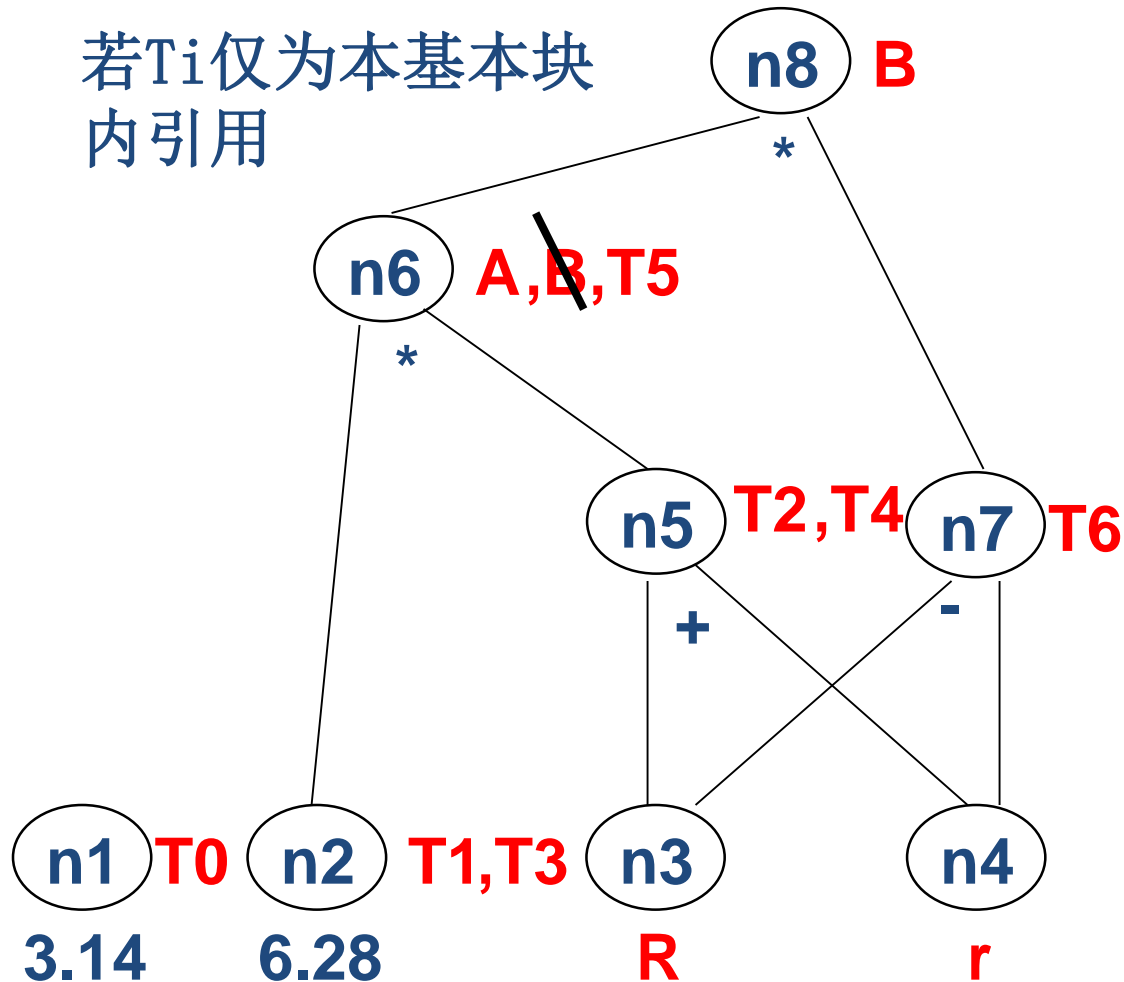
DAG各结点上的附加标识符。

利用这些信息，根据有关变量在基本块后被引用的情况，可以进一步删除基本块中的其它无用赋值。



## 利用DAG进行优化

若 $T_i$ 仅为本基本块内引用



重构四元式序列

- $T2 := R + r$
- $A := 6.28 * T2$
- $T6 := R - r$
- $B := A * T6$



## 10.3 控制流分析 和 循环优化

- 循环中的代码需要反复执行，提高循环的代码优化对整个目标代码的效率至关重要。
- 要进行循环优化，必须找出程序中的循环(一种控制结构)，即需要对程序的控制流进行分析，从程序的控制流图中找出程序的循环。
- 本节主要内容：
  - 一、程序流图
  - 二、循环的查找
  - 三、循环优化

# 一、程序流图

- 控制流程图

具有唯一首结点的有向图。简称为流图。

三元组  $G=(N, E, n_0)$

- 程序可以用一个流图来表示：

- 1) 流图中的**结点**：就是程序中的基本块，首结点就是包含程序第一个语句的基本块。

- 2) 流图中的**有向边**： [从i到j引一有向边]

- 基本块i的出口语句不是无条件转移语句 或 停止语句时，并且基本块j在程序的位置紧跟在基本块i之后。
- 基本块i的出口语句是if...goto(s)或goto(s)，并且(s)是基本块j的入口语句。



# 控制流程图

- 举例:

(1) read (C)

(2)  $A := 0$

(3)  $B := 1$

(4) L1:  $A := A + B$

(5) if  $B \geq C$  goto L2

(6)  $B := B + 1$

(7) goto L1

(8) L2: write (A)

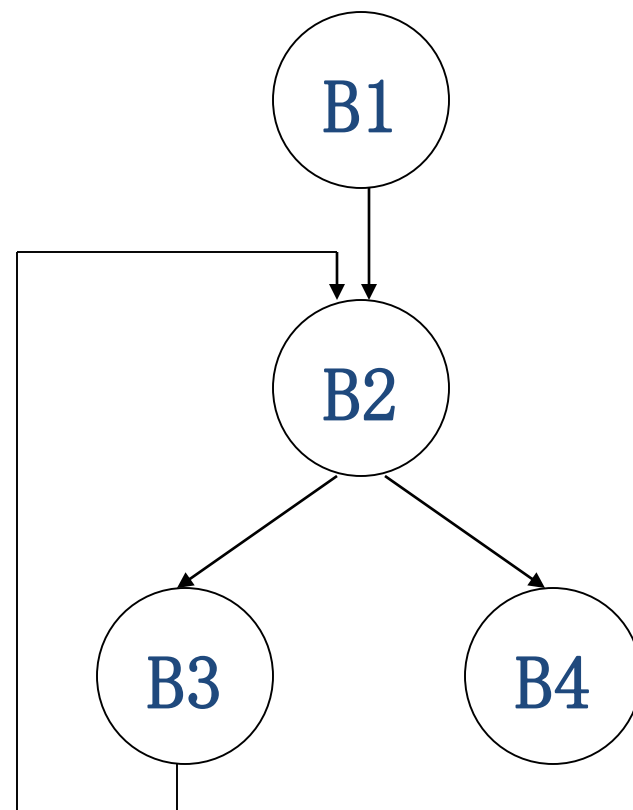
(9) halt

**B1**

**B2**

**B3**

**B4**







## 二、循环的查找

- 循环

流图中具有唯一入口结点的强连通子图。[强连通子图：任意两个结点之间必有一条通路，且该通路上各结点都属于该循环]

- 相关概念

- **必经结点**：在程序流图中，对任意两个结点 $m$ 、 $n$ ，如果从流图的首结点出发，到达 $n$ 的任一通路都要经过 $m$ ，则称  $m$  为  $n$  的必经结点。记为  $m \text{ DOM } n$ 。
- **必经结点集**：结点 $n$ 的所有必经结点的集合，称为  $n$  的必经结点集，记为  $D(n)$ 。

## 二、循环的查找

- 步骤

- (1) 求必经结点集

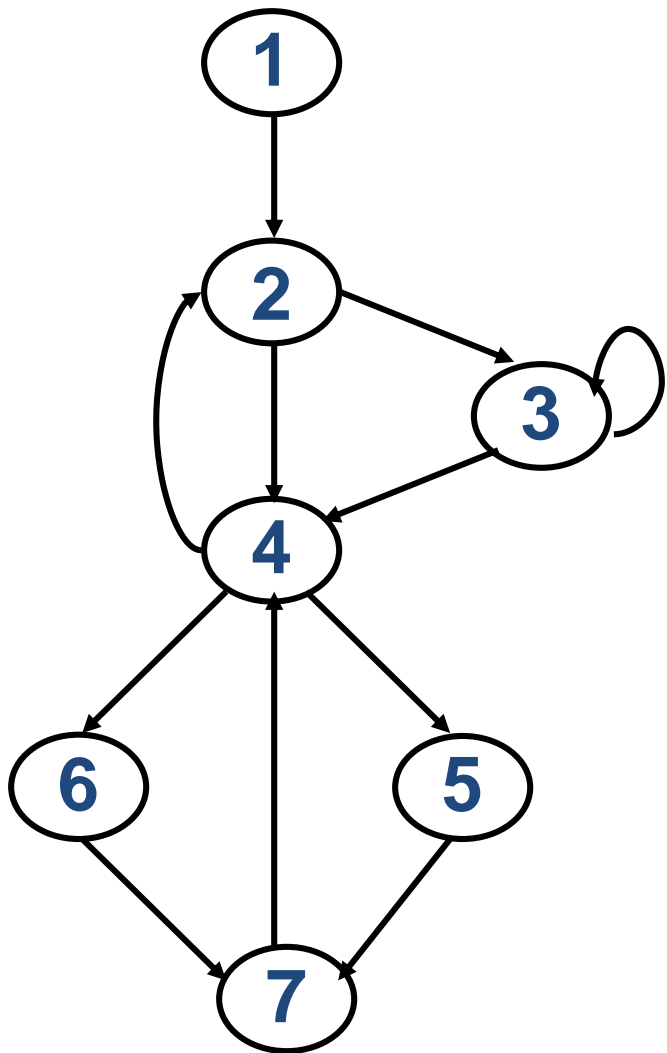
- (2) 求回边

假设 $a \rightarrow b$ 是流图中的一条有向边，如果 $b$ 是 $a$ 的必经结点，则称 $a \rightarrow b$ 是流图中的一条回边。

- (3) 根据回边确定循环

回边 $a \rightarrow b$ ， $b$ 是循环的唯一入口，循环包括 $b$ ， $a$ ，以及所有有通路到达 $a$ 并且不经过 $b$ 的结点。

## 循环查找 举例



### (1) 求必经结点集

$$D(1) = \{1\}$$

$$D(2) = \{1, 2\}$$

$$D(3) = \{1, 2, 3\}$$

$$D(4) = \{1, 2, 4\}$$

$$D(5) = \{1, 2, 4, 5\}$$

$$D(6) = \{1, 2, 4, 6\}$$

$$D(7) = \{1, 2, 4, 7\}$$

### (2) 求回边

$$3 \rightarrow 3 \quad \because D(3) = \{1, 2, 3\}$$

$$4 \rightarrow 2 \quad \because D(2) = \{1, 2, 4\}$$

$$7 \rightarrow 4 \quad \because D(4) = \{1, 2, 4, 7\}$$

### (3) 根据回边查找循环

{3}

{2, 3, 4, 5, 6, 7},

{4, 5, 6, 7}

## 三、循环优化

循环优化的重要技术：

- 代码外提：减少循环中的代码数目
- 强度削弱
- 删除归纳变量



## 代码外提

1. 建立前置结点（外提的代码全部放入前置结点中）
2. 将循环不变运算外提到前置结点中：
  - 外提的运算是循环所有出口结点的必经结点；
  - 外提的变量只在循环的一个基本块中是定值点，在循环的其他基本块中不再有定值点。



## 强度削弱

- 把程序中执行时间较长的运算替换为执行时间较短的运算。
- 强度削弱后可能出现一些新的无用赋值，如果它们在循环的出口之后不是活跃变量，将其删除。



## 删除归纳变量

- 基本归纳变量：循环中变量I只有唯一的形如  $I := I + C$  的赋值，其中的C为循环不变量，则称 I 为循环中的基本归纳变量。
- 归纳变量：若I为循环中的基本归纳变量， $J = C1 * I + C2$ ，C1和C2都是循环不变量，则称J为归纳变量，并称 J与I同族。
- 基本归纳变量往往用于计算其它归纳变量和 控制循环，可以用其同族的某一归纳变量来替换，之后就可删除它的无用赋值。





南京理工大学

谢谢各位同学！

