Partial Redundancy Elimination

CS243 Review Session
Full Redundancy

\[ x = b + c \]
\[ y = b + c \]
\[ z = b + c \]
Partial Redundancy

\[ x = b + c \]

\[ z = b + c \]
Partial Redundancy

\[ t = b + c \]
\[ x = t \]
\[ z = t \]
u = a + b
v = a + b
w = a + b

b = read()

Original graph
Add blocks on edges whose destinations have multiple predecessors
\[ u = a + b \]
\[ v = a + b \]
\[ w = a + b \]
\[ b = \text{read()} \]
$u = a + b$
$v = a + b$
$w = a + b$

$B_1$

$B_2$
$B_3$

$B_4$

$B_5$
$B_6$

$B_7$

$B_8$

$B_9$
$B_{10}$

$B_{11}$

**Anticipated expressions:** places where it is **safe** to place $a + b$
\[ u = a + b \]
\[ v = a + b \]
\[ w = a + b \]

Can delete added blocks where \[ a + b \] is not anticipated.
\[ u = a + b \]
\[ v = a + b \]
\[ w = a + b \]

Available expressions: points where \( a + b \) could be made available
\[ u = a + b \]

\[ v = a + b \]

\[ w = a + b \]

\[ b = \text{read()} \]

\textbf{Earliest:} when can we earliest compute \( a + b \)
\[ u = a + b \]
\[ v = a + b \]
\[ w = a + b \]

*Earliest: when can we earliest compute \( a + b \)*
How much can we postpone evaluating $a + b$?
Latest: need to compute $a + b$ here
Latest: need to compute \(a + b\) here
Remove added blocks where we are not going to compute anything
\[ u = t \]

\[ t = a + b \]

\[ v = t \]

\[ w = t \]

\[ b = \text{read()} \]

Use a temporary variable to store the result

\[ t = a + b \]
\[ u = t \]
\[ t = a + b \]
\[ v = t \]
\[ t = a + b \]
\[ w = t \]
\[ b = \text{read()} \]
\[ t = a + b \]
\[ u = t \]

\[ t = a + b \]

\[ v = t \]

\[ t = a + b \]

\[ w = t \]

\[ b = \text{read()} \]

Result not used beyond the block in which the variable is defined.
$u = t$

$v = a + b$

$t = a + b$

$w = t$

$b = \text{read()}$

Clean up unrequired temporaries
\[ u = t \]
\[ v = a + b \]
\[ t = a + b \]
\[ w = t \]
\[ b = \text{read()} \]