CS243 Homework 3

Winter 2018

Due: February 7, 2018

Directions:

- Complete the following problems and hand them in at the beginning of class with your name and SUID at the top.
- **There are two Gradiance assignments this week (Register Allocation and Dominators)**. Remember to complete them by the start of class on the due date. There are no late days for Gradiance.
- If you need to use one or more late days, hand in your assignment by 4:30pm Thursday or Friday in Gates 407. Slide it under the door if it is locked.
- This is an individual assignment. You are allowed to discuss the homework with others, but you must write the solution individually. If you look up any material in the textbook or online, you should cite it appropriately.

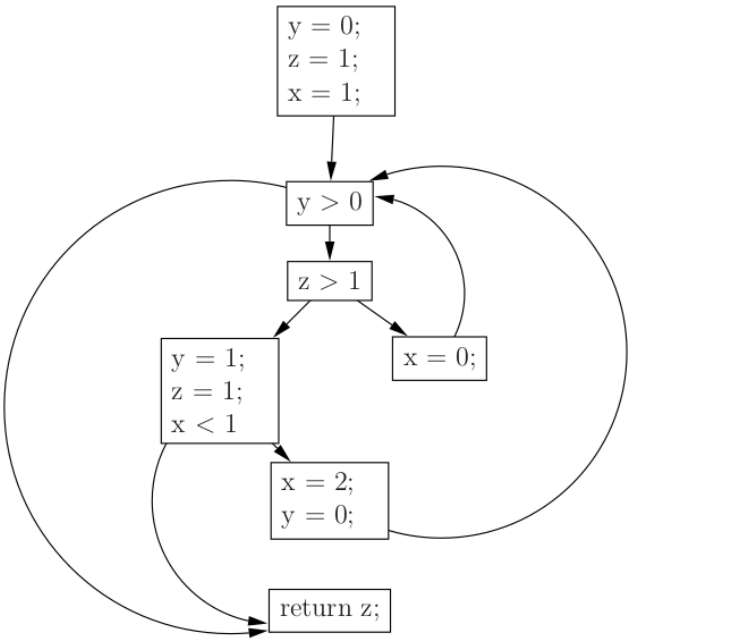
Problem 1. Initial Values in Dataflow Analysis

This question asks you to think about how changes to initial values in a data flow analysis can affect the result. Recall that an answer to a data flow problem is considered "safe" if it is no bigger than the ideal solution.

Suppose you have defined a forward dataflow algorithm that is distributive, has a monotonic transfer function and has finite descending chains. You accidentally initialized OUT[b] for all basic block to $\perp$.

1. Will your algorithm give a safe answer for all flow graphs?
2. If not, will it give a safe answer for some flow graphs? If it will, give an example.
3. Will your algorithm give the MOP solution for all flow graphs?
4. If not, will it give the MOP solution for some flow graphs? If it will, give an example.
5. What if instead you had initialized IN[b] to $\perp$. Would the algorithm produce the correct solution?

Problem 2. Consider the following flow graph:

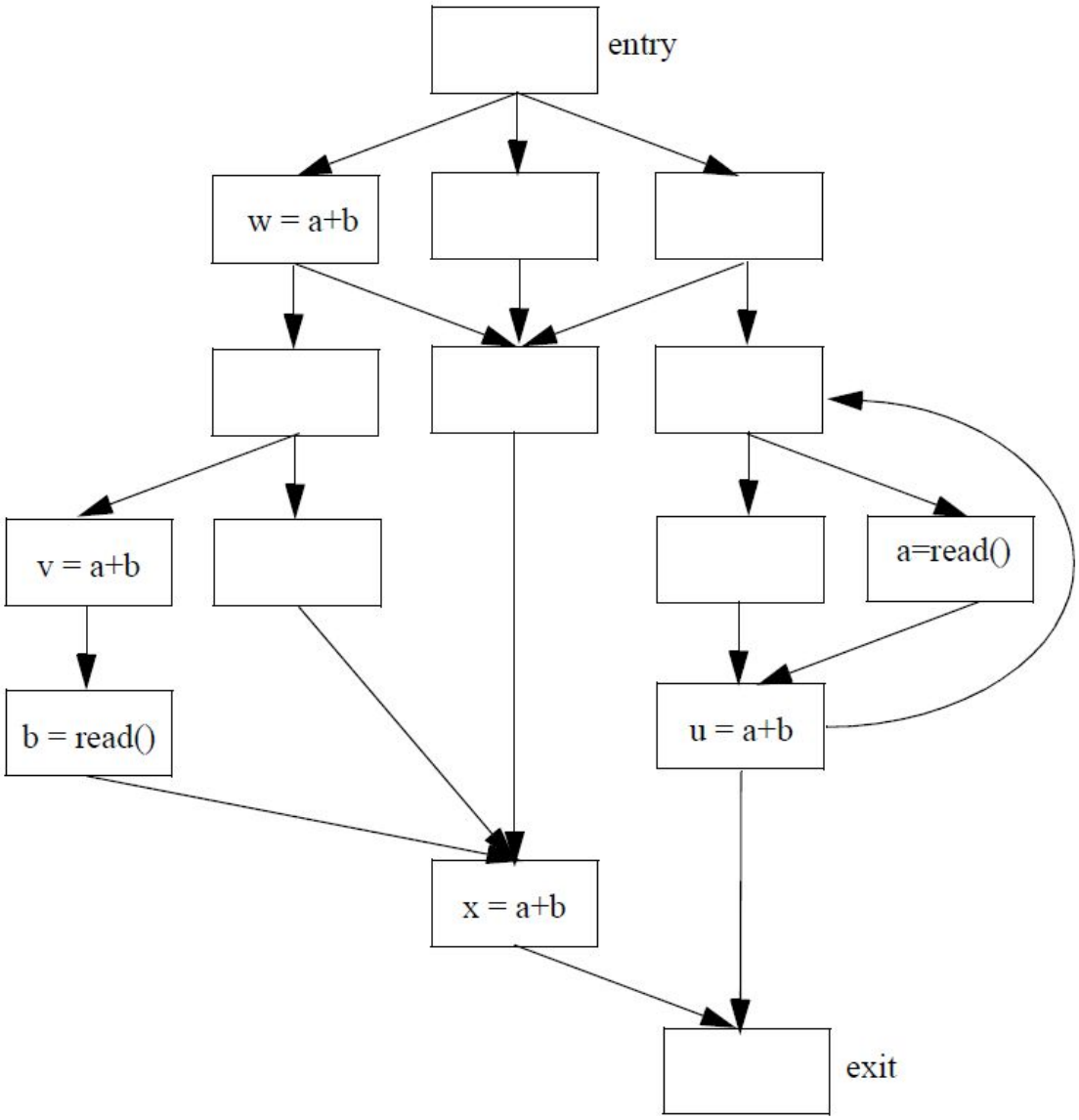


1. As seen in class, a flow graph is *reducible* if all retreating edges are back edges. Is this flow graph reducible? Justify your answer.

2. Assuming a language with **goto**, **if**, **while**, **break** and **continue** statements, what code could generate that flow graph? Use the minimum number of gotos.

3. Given the previous language, are there flow graphs that cannot be expressed using those control statements? What if we remove **goto**?

Problem 3. Apply Partial Redundancy Elimination to the following program. Assume that w, v, x, and u are used in other portions of the code (not shown), possibly in the same basic block. You do not need to show the intermediate steps, just show the optimized code. You may add basic blocks to the flow graph, but only show those that are not empty in your solution (existing basic blocks are not empty, even if they appear to be).

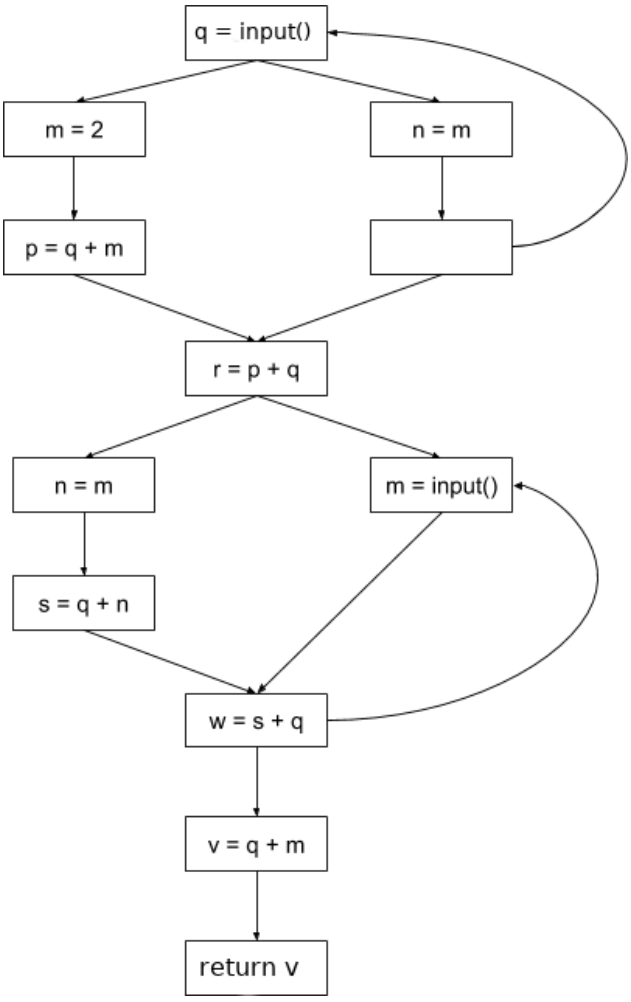

entry
w = a+b
v = a+b
b = read()
a=read()
u = a+b
x = a+b
exit

Problem 4. You are given the task of optimizing the code given below. You are only allowed to run the following four optimization techniques:

- PRE (as discussed in class)
- Constant Propagation (as discussed in class)
- Copy Propagation (as discussed in Section 9.1.5 of the textbook)
- Dead Code Elimination (liveness analysis, as discussed in class and in Homework 2)

in any order and multiple time if necessary. You can move and duplicate input(), but it must be executed exactly the same number of times as the original graph. You cannot modify the control flow graph or eliminate empty basic blocks, except to preprocess it for PRE. As in joeq, assume that expression can take both registers and constants.

1. What is the order in which you should execute them to produce the best optimized code by running a minimum number of analyses?
2. What is the final optimized program?



Problem 5. For the following control flow graph, perform register allocation. Assign each definition and use of a variable to a live range. Draw the interference graph and the assignment of registers (i.e. colors) to live ranges.

1. What is the number of registers used by the heuristic algorithm?

2. What is the minimum number of registers needed by this program? Justify your answer.