Alias Analysis

15-411/15-611 Compiler Design

Seth Copen Goldstein

November 24, 2020

Constant Propagation on Memory

$$l_1: M[q] \leftarrow 4$$

 $l_2: M[p] \leftarrow 8128$
 $l_3: x \leftarrow M[q]$

• Can we replace l_3 with $\mathbf{x} \leftarrow \mathbf{4}$?

Constant Propagation on Memory

$$l_1: M[q] \leftarrow 4$$

 $l_2: M[p] \leftarrow 8128$
 $l_3: x \leftarrow M[q]$

- Can we replace l_3 with $x \leftarrow 4$?
- Only if $p \neq q$
- That is, only if p and q do not alias

CSE

```
foo() {
      int a,k;
      extern int *q;
      ... // maybe &a or &k
      k = a+6;
      f(a,&k);
      *q = 13;
      k = a+6
```

Source of Aliases

- Pass by reference parameters
- Address of operator
- Dereferencing pointers
- Array subscripting
- Non-local variables
- Assignment

CSE on Memory

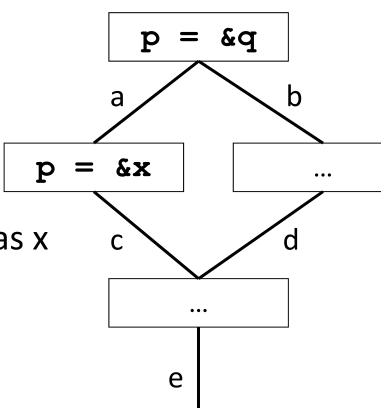
```
struct point {
   int x;
   int y;
};
typedef struct point pt;

void mult(int[] A, pt* p, pt* q) {
   q->x = A[0] * p->x + A[1] * p->y;
   q->y = A[2] * p->x + A[3] * p->y;
   return;
}
```

```
\mathsf{mult}(A, p, q):
       t_0 \leftarrow M[A+0]
       t_1 \leftarrow M[p+0]
       t_2 \leftarrow t_0 * t_1
       t_3 \leftarrow M[A+4]
       t_4 \leftarrow M[p+4]
     t_5 \leftarrow t_3 * t_4
      t_6 \leftarrow t_2 + t_5
       M[q+0] \leftarrow t_6
       t_8 \leftarrow M[A+8]
       t_9 \leftarrow M[p+0]
       t_{10} \leftarrow t_8 * t_9
       t_{11} \leftarrow M[A+12]
       t_{12} \leftarrow M[p+4]
       t_{13} \leftarrow t_{11} * t_{12}
       t_{14} \leftarrow t_{10} + t_{13}
       M[q+4] \leftarrow t_{14}
       return
```

Alias Information

- must-alias and may-alias
 - a: p must-alias q
 - b: p must-alias q
 - c: p must-alias q
 - d: p must-alias q
 - e: p may-alias q and p may-alias x
- Analysis:
 - flow-insensitive
 - flow-sensitive



What regions?

- Stack based
 - include pointers to stack allocated objects
 - Can use name of locals
- Heap based
 - pointers to heap addresses, e.g., returned from alloc
 - Need way to "name" the objects
 - Will use line number

Flow-based May-Alias

At each program point a set of tuples:
 (t, d, k)

- t: variable

- d: alias class

k: offset into alias class

- (t,d,k) means variable t may point to alias class d, or
- t-k points to d

```
struct intlist {
       int val;
       struct intlist* next;
};
struct int foo() {
      struct intlist* p;
      struct intlist* q;
      int a = 1;
      q = alloc(struct intlist);
      q->val = 0;
      q->next NULL;
      p = alloc(struct intlist);
      p->val = 0;
      p->next = q;
      q->val = 5;
      a = p->val;
      return a;
```

```
MOVE t2 <-0
1:
                                       p
2: MOVE t3
              <- 0
                                       q
3:
   MOVE t4
                <- 1
                                       a
  CALL t5 \leftarrow malloc(8) (t5,4,0) q \leftarrow alloc
4:
5: PLUS u1 <- t5,
                        0
6: STORE MEM[u1] <- 0
                                       q->val = 0
7: PLUS u2 <- t5, 4
8: STORE MEM[u2] <- 0
                                       q->next = 0
9: MOVE t3 <- t5
                                       q
10: CALL t6 <- malloc(8)
                                       p <- alloc
11: PLUS u3 <- t6, 0
12: STORE MEM[u3] <- 0
                                       p->val = 0
13: PLUS u4 <- t6,
                         4
14: STORE MEM[u4] <- t3
                                       p->next = q
15: MOVE t2 <- t6
                                       p
16: PLUS u5 <- t3,
                                       q->val = 5
17: STORE MEM[u5] <- 5
18: PLUS u6 <- t2, 0
                                       p
19: LOAD t4 <- MEM[u6]
                                       a
20: RET
                t4
```

```
MOVE t2 <- 0
1:
                                        p
   MOVE t3
               <- 0
2:
                                        q
                <- 1
3:
   MOVE t4
                                        a
  CALL t5 \leftarrow malloc(8) (t5,4,0)
4:
                                       q <- alloc
  PLUS u1 \leftarrow t5,
                             (u1,4,0)
5:
                         0
6: STORE MEM[u1] <- 0
                                       q->val = 0
7: PLUS u2 <- t5,
                         4
8: STORE MEM[u2] <- 0
                                       q->next = 0
9: MOVE t3 <- t5
                                       q
10: CALL t6 <- malloc(8)
                                       p <- alloc
11: PLUS u3 <- t6, 0
12: STORE MEM[u3] <- 0
                                       p->val = 0
13: PLUS u4 <- t6,
                         4
14: STORE MEM[u4] <- t3
                                       p->next = q
15: MOVE t2 <- t6
                                       p
16: PLUS u5 <- t3,
                         0
17: STORE MEM[u5] <- 5
                                       q->val = 5
18: PLUS u6 <- t2,
                         0
                                       p
19: LOAD t4
              <- MEM[u6]
                                       a
20: RET
                 t4
```

```
1:
   MOVE t2 <- 0
                                        p
   MOVE t3
               <- 0
2:
                                        q
   MOVE t4
3:
                <- 1
                                        a
  CALL t5 \leftarrow malloc(8) (t5,4,0)
4:
                                        q <- alloc
  PLUS u1 <- t5,
5:
                         0
                             (u1,4,0)
6: STORE MEM[u1] \leftarrow 0
                                        q->val = 0
  PLUS u2 <- t5, 4
7:
                             (u2,4,4)
8: STORE MEM[u2] <- 0
                                        q->next = 0
9:
  MOVE t3 <- t5
                                        q
10: CALL t6 <- malloc(8)
                                        p <- alloc
11: PLUS u3 <- t6, 0
12: STORE MEM[u3] <- 0
                                        p->val = 0
13: PLUS u4 <- t6,
                         4
14: STORE MEM[u4] <- t3
                                        p->next = q
15: MOVE t2 <- t6
                                        p
16: PLUS u5 <- t3,
                         0
17: STORE MEM[u5] <- 5
                                        q->val = 5
18: PLUS u6 <- t2,
                         0
                                        p
19: LOAD t4
              <- MEM[u6]
                                        a
20: RET
                 t4
                                                14
```

```
1:
   MOVE t2 <- 0
                                       p
   MOVE t3
               <- 0
2:
                                       q
   MOVE t4
3:
                <- 1
                                       a
  CALL t5 \leftarrow malloc(8) (t5,4,0)
4:
                                       q <- alloc
  PLUS u1 <- t5,
5:
                        0
                             (u1,4,0)
6: STORE MEM[u1] <- 0
                                       q->val = 0
7: PLUS u2 <- t5, 4
                             (u2,4,4)
8: STORE MEM[u2] <- 0
                                       q->next = 0
                             (t3,4,0)
9: MOVE t3 <- t5
                                       q
10: CALL t6 <- malloc(8)
                                       p <- alloc
11: PLUS u3 <- t6, 0
12: STORE MEM[u3] <- 0
                                       p->val = 0
13: PLUS u4 <- t6,
                         4
14: STORE MEM[u4] <- t3
                                       p->next = q
15: MOVE t2 <- t6
                                       p
16: PLUS u5 <- t3,
                         0
17: STORE MEM[u5] <- 5
                                       q->val = 5
18: PLUS u6 <- t2,
                         0
                                       p
19: LOAD t4
             <- MEM[u6]
                                       a
20: RET
                t4
                                               15
```

```
1:
   MOVE t2 <- 0
                                       p
   MOVE t3
               <- 0
2:
                                       q
   MOVE t4
               <- 1
3:
                                       a
  CALL t5 <- malloc(8)
4:
                            (t5,4,0)
                                      q <- alloc
  PLUS u1 <- t5,
5:
                        0
                            (u1,4,0)
6: STORE MEM[u1] <- 0
                                      q->val = 0
        u2 <- t5, 4
7:
  PLUS
                            (u2,4,4)
8:
  STORE MEM[u2] <- 0
                                      q->next = 0
9:
  MOVE t3 <- t5
                            (t3,4,0)
                                      q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
12: STORE MEM[u3] <- 0
                                      p->val = 0
13: PLUS u4 <- t6,
                        4
14: STORE MEM[u4] <- t3
                                      p->next = q
15: MOVE t2 <- t6
                                      p
16: PLUS u5 <- t3,
                        0
17: STORE MEM[u5] <- 5
                                      q->val = 5
18: PLUS u6 <- t2,
                        0
                                      p
19: LOAD t4
             <- MEM[u6]
                                      a
20: RET
                t4
                                              16
```

```
1:
   MOVE t2 <- 0
                                        p
   MOVE t3
               <- 0
2:
                                        q
   MOVE t4
3:
                <- 1
                                        a
  CALL t5 <- malloc(8) (t5,4,0)
4:
                                       q <- alloc
  PLUS u1 <- t5,
5:
                         0
                             (u1,4,0)
6: STORE MEM[u1] \leftarrow 0
                                       q->val = 0
        u2 <- t5, 4
7:
  PLUS
                             (u2,4,4)
8: STORE MEM[u2] <- 0
                                       q->next = 0
9:
  MOVE t3 <- t5
                             (t3,4,0)
                                       q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
                             (u3,10,0)
12: STORE MEM[u3] <- 0
                                       p->val = 0
        u4 <- t6,
                         4
13: PLUS
14: STORE MEM[u4] <- t3
                                       p->next = q
15: MOVE t2 <- t6
                                       p
16: PLUS u5 <- t3,
                         0
17: STORE MEM[u5] <- 5
                                       q->val = 5
18: PLUS u6 <- t2,
                         0
                                       p
19: LOAD t4
              \leftarrow MEM[u6]
                                       a
20: RET
                 t4
                                               17
```

```
1:
   MOVE
        t2 <- 0
                                       p
   MOVE t3
                <- 0
2:
                                        q
3:
   MOVE t4
                <- 1
                                       a
   CALL t5 <- malloc(8)
4:
                             (t5,4,0)
                                       q <- alloc
  PLUS u1 <- t5,
                             (u1,4,0)
5:
                         0
6: STORE MEM[u1] \leftarrow 0
                                       q->val = 0
        u2 <- t5, 4
7:
  PLUS
                             (u2,4,4)
  STORE MEM[u2] <- 0
8:
                                       q->next = 0
9:
  MOVE
        t3 <- t5
                             (t3,4,0)
                                       q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
                             (u3,10,0)
12: STORE MEM[u3] <- 0
                                       p->val = 0
                             (u4,10,4)
        u4 <- t6,
                         4
13: PLUS
14: STORE MEM[u4] <- t3
                                       p->next = q
15: MOVE t2 <- t6
                                       p
16: PLUS u5 <- t3,
                         0
17: STORE
         MEM[u5] < - 5
                                       q->val = 5
18: PLUS
        u6 <- t2,
                         0
                                       p
19: LOAD t4
              <- MEM[u6]
                                       a
20: RET
                t4
```

```
1:
   MOVE
        t2 <- 0
                                        p
   MOVE t3
                <- 0
2:
                                        q
3:
   MOVE t4
                <- 1
                                        a
   CALL t5 <- malloc(8)
4:
                             (t5,4,0)
                                        q <- alloc
             <- t5,
                             (u1,4,0)
5:
  PLUS u1
                         0
6: STORE MEM[u1] \leftarrow 0
                                        q->val = 0
         u2 <- t5, 4
7:
  PLUS
                             (u2,4,4)
  STORE MEM[u2] <- 0
8:
                                        q->next = 0
9:
  MOVE
        t3 <- t5
                             (t3,4,0)
                                        q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
                             (u3,10,0)
12: STORE MEM[u3] <- 0
                                        p->val = 0
        u4 <- t6,
                         4
                              (u4,10,4)
13: PLUS
14: STORE MEM[u4] <- t3
                                        p->next = q
15: MOVE t2 <- t6
                              (t2,10,0)
                                        p
16: PLUS u5 <- t3,
                         0
17: STORE
         MEM[u5] < - 5
                                        q->val = 5
18: PLUS
        u6 <- t2,
                         0
                                        p
19: LOAD t4
              <- MEM[u6]
                                        a
20: RET
                 t4
```

```
1:
   MOVE
        t2 <- 0
                                         p
   MOVE t3
                <- 0
2:
                                         q
3:
   MOVE t4
                <- 1
                                         a
   CALL t5 <- malloc(8)
4:
                              (t5,4,0)
                                        q <- alloc
              <- t5,
                              (u1,4,0)
5:
  PLUS u1
                         0
6: STORE MEM[u1] \leftarrow 0
                                        q->val = 0
         u2 <- t5,
7:
  PLUS
                         4
                              (u2,4,4)
8:
  STORE MEM[u2] <- 0
                                        q->next = 0
9:
   MOVE
        t3
              <- t5
                              (t3,4,0)
                                        q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
                              (u3,10,0)
                                        p->val = 0
12: STORE MEM[u3] <- 0
         u4 <- t6,
                         4
13: PLUS
                              (u4,10,4)
14: STORE MEM[u4] <- t3
                                        p->next = q
             <- t6
        t2
15: MOVE
                              (t2,10,0)
                                        p
16: PLUS u5 <- t3,
                         0
                              (u5,4,0)
17: STORE
         MEM[u5] < - 5
                                        q->val = 5
18: PLUS
         u6 <- t2,
                         0
                                        p
19: LOAD t4
              <- MEM[u6]
                                        a
20: RET
                 t4
```

```
1:
   MOVE
        t2 <- 0
                                        p
   MOVE t3
                <- 0
2:
                                        q
3:
   MOVE t4
                <- 1
                                        a
   CALL t5 <- malloc(8)
4:
                             (t5,4,0)
                                        q <- alloc
             <- t5,
                             (u1,4,0)
5:
  PLUS u1
                         0
6: STORE MEM[u1] <- 0
                                        q->val = 0
         u2 <- t5, 4
7:
  PLUS
                             (u2,4,4)
  STORE MEM[u2] <- 0
8:
                                        q->next = 0
9:
   MOVE
        t3
              <- t5
                             (t3,4,0)
                                        q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
11: PLUS u3 <- t6, 0
                             (u3,10,0)
12: STORE MEM[u3] <- 0
                                        p->val = 0
        u4 <- t6,
                         4
13: PLUS
                              (u4,10,4)
14: STORE MEM[u4] <- t3
                                        p->next = q
        t2 <- t6
15: MOVE
                              (t2,10,0)
                                        p
16: PLUS u5 <- t3,
                         0
                             (u5, 4, 0)
17: STORE
         MEM[u5] < - 5
                                        q->val = 5
18: PLUS
        u6 <- t2,
                         0
                             (u6,4,0)
                                        p
19: LOAD t4
              <- MEM[u6]
                                        a
20: RET
                 t4
```

```
1:
   MOVE
         t2
                  <- 0
                                            p
    MOVE t3
                  <- 0
2:
                                            q
3:
    MOVE t4
                  <- 1
                                            a
   CALL t5 <- malloc(8)
4:
                                (t5,4,0)
                                           q <- alloc
               <- t5,
5:
   PLUS
         u1
                           0
                                (u1,4,0)
6: STORE MEM[u1] <- 0
                                           q->val = 0
               <- t5,
          u2
7:
   PLUS
                           4
                                (u2,4,4)
8:
   STORE MEM[u2] <- 0
                                           q->next = 0
9:
    MOVE
         t3
                 <- t5
                                (t3,4,0)
                                           q
10: CALL t6 <- malloc(8) (t6,10,0) p <- alloc
               <- t6,
                                (u3, 10, 0)
11: PLUS u3
                           0
12: STORE
         MEM[u3] < - 0
                                           p->val = 0
          \mathbf{u4}
                <- t6,
                           4
13: PLUS
                                (u4,10,4)
14: STORE
         MEM[u4] < - t3
                                           p->next = q
         t2
                <- t6
15: MOVE
                                (t2,10,0)
                                           p
16: PLUS u5
               <- t3,
                           0
                                (u5,4,0)
17: STORE
          MEM[u5]
                 <- 5
                                           q->val = 5
                                (u6,10,0)
18: PLUS
                <- t2,
                           0
         u6
                                           p
19: LOAD
                  <- MEM[u6]
         t4
                                           a
20: RET
                  t4
```

```
1:
    MOVE
          t2
                   <- 0
                                               p
    MOVE t3
                   <- 0
2:
                                               q
    MOVE t4
3:
                   <- 1
                                               a
                   <- malloc(8)
4:
   CALL t5
                                   (t5,4,0)
                                               q <- alloc
                 <- t5,
5:
   PLUS
          u1
                              0
                                   (u1,4,0)
6: STORE
          MEM[u1] <- 0
                                               q->val = 0
                   <- t5,
           u2
7:
   PLUS
                              4
                                   (u2,4,4)
8:
   STORE MEM[u2] <- 0
                                               q->next = 0
9:
    MOVE
          t3
                    <- t5
                                   (t3,4,0)
                                               q
                                   (t6,10,0) p <- alloc
10: CALL t6
                 <- malloc(8)
                  <- t6,
                                   (u3, 10, 0)
11: PLUS u3
                             0
12: STORE
          MEM[u3] < - 0
                                               p->val = 0
                  <- t6,
                              4
13: PLUS
           \mathbf{u4}
                                   (u4,10,4)
14: STORE
          MEM[u4] < -t3
                                               p->next = q
          t2
                  <- t6
15: MOVE
                                   (t2,10,0)
                                               p
16: PLUS
                 <- t3,
                              0
                                   (u5, 4, 0)
          u5
           MEM[u5] < - 5
17: STORE
                                               q->val = 5
18: PLUS
           <del>u6</del>
                    < t2,
19: MOVE
           t4
                    <- 0
                                               a
201: RET
                    t4
```

```
1:
    MOVE
           t2
                    <- 0
                                                  p
          t3
                    <- 0
2:
    MOVE
                                                  q
3:
    MOVE
          t4
                     <- 1
                                                  a
                    <- malloc(8)
4:
    CALL t5
                                     (t5,4,0)
                                                 q <- alloc
                   <- t5,
5:
    PLUS
          u1
                               0
                                     (u1,4,0)
6:
    STORE
           MEM[u1]
                   <- 0
                                                 q->val = 0
           u2
7:
   PLUS
                    <- t5,
                               4
                                     (u2,4,4)
8:
   STORE
           MEM[u2] <- 0
                                                 q->next = 0
9:
    MOVE
           t3
                     <- t5
                                     (t3,4,0)
                                                 q
                                     (t6,10,0)
                                                 p <- alloc
10: CALL t6
                    <- malloc(8)
                    <- t6,
11: PLUS
          u3
                                     (u3,10,0)
                               0
12: STORE
           MEM[u3] < - 0
                                                 p->val = 0
                   <- t6,
13: PLUS
           \mathbf{u4}
                               4
                                     (u4,10,4)
14: STORE
           MEM[u4] < -t3
                                                 p->next = q
           t2
15: MOVE
                    <- t6
                                     (t2,10,0)
                                                 p
                    <- t3,
16: PLUS
                               0
           u5
                                     (u5,4,0)
17: STORE
           MEM[u5]
                       5
                   <-
                                                 q->val = 5
18: PLUS
                     <del>< t2,</del>
                                                 a
201: RET
                     0
```

```
struct intlist* foo() {
      struct intlist* p;
      struct intlist* q;
      int a = 1;
      p = alloc(struct intlist);
      p->val = 0;
      p->next = 0;
      q = alloc(struct intlist);
      q->val = 6;
      q->next = p;
      if (a == 0) p = q;
      p->val = 4;
      return p;
```

```
1: CALL t2 <- malloc(8)
                                      p
2: PLUS u1 <- t2,0
3: STR MEM[u1] < -0
                                      p->val = 0
4: PLUS u2 <- t2, 4
5: STR MEM[u2] \leftarrow 0
                                      p->next = 0
6: MOVE t3 <- t2
                                      q=x
7: CALL t4 <- malloc(8)
                                      q
8: PLUS u3 <- t4,0
9: STR MEM[u3] <- 6
                                      q->val = 6
10: PLUS u4 <- t4,0
11: STR MEM[u4] <- t3
                                      q-next = p
12: MOVE t5 <- t4
                                      q
13: MOVE t6 <- 1
                                      a
14: CJUMP t6 == 0, ifT0, ifF1
                                      a == 0 ?
ifT0:
16: MOVE t3' <- t5
                                      x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
```

21: RET t2

```
1: CALL t2 <- malloc(8) (t2,1,0) p
2: PLUS u1 <- t2,0
3: STR MEM[u1] < -0
                                      p->val = 0
4: PLUS u2 <- t2, 4
5: STR MEM[u2] \leftarrow 0
                                      p->next = 0
6: MOVE t3 <- t2
                                      q=x
7: CALL t4 <- malloc(8)
                                      q
8: PLUS u3 <- t4,0
9: STR MEM[u3] <- 6
                                      q->val = 6
10: PLUS u4 <- t4,0
11: STR MEM[u4] <- t3
                                      q-next = p
12: MOVE t5 <- t4
                                      q
13: MOVE t6 <- 1
                                      a
14: CJUMP t6 == 0, ifT0, ifF1
                                     a == 0 ?
ifT0:
16: MOVE t3' <- t5
                                      x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
21: RET t2
```

```
1: CALL t2 <- malloc(8) (t2,1,0)
                                      p
2: PLUS u1 \leftarrow t2,0 (u1,1,0)
3: STR MEM[u1] < -0
                                      p->val = 0
4: PLUS u2 \leftarrow t2, 4 (u2,1,4)
5: STR MEM[u2] < -0
                                      p->next = 0
6: MOVE t3 <- t2
                                      q=x
7: CALL t4 <- malloc(8)
                                      q
8: PLUS u3 <- t4,0
9: STR MEM[u3] <- 6
                                      q->val = 6
10: PLUS u4 <- t4,0
11: STR MEM[u4] <- t3
                                      q-next = p
12: MOVE t5 <- t4
                                      q
13: MOVE t6 <- 1
                                      a
14: CJUMP t6 == 0, ifT0, ifF1
                                      a == 0 ?
ifT0:
16: MOVE t3' <- t5
                                      x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
21: RET t2
```

```
1: CALL t2 <- malloc(8) (t2,1,0)
                                       p
2: PLUS u1 \leftarrow t2,0 (u1,1,0)
3: STR MEM[u1] \leftarrow 0
                                       p->val = 0
4: PLUS u2 \leftarrow t2, 4 (u2,1,4)
5: STR MEM[u2] \leftarrow 0
                                       p->next = 0
6: MOVE t3 \leftarrow t2 (t3,1,0)
                                       q=x
7: CALL t4 <- malloc(8) (t4,7,0)
                                       q
8: PLUS u3 \leftarrow t4,0 (u3,7,0)
9: STR MEM[u3] <- 6
                                       q->val = 6
10: PLUS u4 < -t4,0 (u4,7,0)
11: STR MEM[u4] <- t3
                                       q-next = p
12: MOVE t5 < -t4 (t5,7,0)
                                       q
13: MOVE t6 <- 1
                                       a
14: CJUMP t6 == 0, ifT0, ifF1
                                       a == 0 ?
ifT0:
16: MOVE t3' <- t5
                                       x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
21: RET t2
```

```
1: CALL t2 <- malloc(8) (t2,1,0)
                                      p
2: PLUS u1 \leftarrow t2,0 (u1,1,0)
3: STR MEM[u1] < - 0
                                      p->val = 0
4: PLUS u2 \leftarrow t2, 4 (u2,1,4)
5: STR MEM[u2] \leftarrow 0
                                      p->next = 0
6: MOVE t3 <- t2 (t3,1,0)
                                      q=x
7: CALL t4 \leftarrow malloc(8) (t4,7,0)
                                      q
8: PLUS u3 \leftarrow t4,0 (u3,7,0)
9: STR MEM[u3] <- 6
                                      q->val = 6
10: PLUS u4 < -t4,0 (u4,7,0)
11: STR MEM[u4] <- t3
                                      q-next = p
12: MOVE t5 < -t4 (t5,7,0)
                                      q
13: MOVE t6 <- 1
                                      a
14: CJUMP t6 == 0, ifT0, ifF1
                                      a == 0 ?
ifT0:
16: MOVE t3' <- t5 (t3',7,0) x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
21: RET t2
```

```
1: CALL t2 <- malloc(8) (t2,1,0)
                                     p
2: PLUS u1 \leftarrow t2,0 (u1,1,0)
3: STR MEM[u1] < -0
                                     p->val = 0
4: PLUS u2 <- t2, 4 (u2,1,4)
5: STR MEM[u2] \leftarrow 0
                                     p->next = 0
6: MOVE t3 <- t2 (t3,1,0)
                                     q=x
7: CALL t4 <- malloc(8) (t4,7,0)
                                     q
8: PLUS u3 \leftarrow t4,0 (u3,7,0)
9: STR MEM[u3] <- 6
                                     q->val = 6
10: PLUS u4 < -t4,0 (u4,7,0)
11: STR MEM[u4] <- t3
                                     q-next = p
12: MOVE t5 <- t4 (t5,7,0)
                                     q
13: MOVE t6 <- 1
                                     a
14: CJUMP t6 == 0, ifT0, ifF1
                                     a == 0 ?
ifT0:
16: MOVE t3' <- t5 (t3',7,0) x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')(t3'',7,0),(t3'',1,0)
19: PLUS u5 <- t3'', 0
20: STR MEM[u5] < -4
21: RET t2
```

```
1: CALL t2 <- malloc(8) (t2,1,0)
                                      p
2: PLUS u1 \leftarrow t2,0 (u1,1,0)
3: STR MEM[u1] < -0
                                       p->val = 0
4: PLUS u2 \leftarrow t2, 4 (u2,1,4)
5: STR MEM[u2] \leftarrow 0
                                      p->next = 0
6: MOVE t3 \leftarrow t2 (t3,1,0)
                                      q=x
7: CALL t4 \leftarrow malloc(8) (t4,7,0)
                                       q
8: PLUS u3 \leftarrow t4,0 (u3,7,0)
9: STR MEM[u3] <- 6
                                       q->val = 6
10: PLUS u4 < -t4,0 (u4,7,0)
11: STR MEM[u4] <- t3
                                       q-next = p
12: MOVE t5 < -t4 (t5,7,0)
                                       q
13: MOVE t6 <- 1
                                       a
14: CJUMP t6 == 0, ifT0, ifF1
                                      a == 0 ?
ifT0:
16: MOVE t3' <- t5 (t3',7,0) x=q
ifF1:
18: PHI t3'' \leftarrow \phi(t3,t3')(t3'',7,0),(t3'',1,0)
19: PLUS u5 <-t3'', 0 (u5,7,0), (u5,1,0)
20: STR MEM[u5] < -4
21: RET t2
```

Dependence Information

```
foo(int[] a, int[] b, int n) {
    int i;
    for (i=1; i<n; i++) {
        a[i] = b[i-1]+a[i]
    }
}</pre>
```

What are the dependence distances?