Polynomial-time Fence Insertion For Structured Programs

M. Taheri, A. Pourdamghani, M. Lesani
University of California, Riverside
Sharif University of Technology

Correctness Dependent on Program Order

```
bool p0()
    flag0 = 1
    if (flag1 == 1)
        return false;
    // critical
    flag0 = 0;
    return true;
}
```

```
bool p1()
    flag1 = 1
    if (flag0 == 1)
        return false;
    // critical
    flag1 = 0;
    return true;
}
```

Dekker Mutual Exclusion Algorithm

Weak (relaxed) memory models

```
1 bool p0()
3    read(flag1): 0


2    write(flag0, 1)

5    // critical
6    flag0 = 0;
7    return true;
8 }
```

```
1 bool p0()

2    write(flag0, 1)
3    read(flag1): 0


5    // critical
6    flag0 = 0;
7    return true;
8 }
```

Out of order execution
Both x86 and ARMv7

Fence Instructions

```
bool p0()
   flag0 = 1
   fence
   if (flag1 == 1)
       return false;
   // critical
   flag0 = 0;
   return true;
}
```

```
bool p1()
   flag1 = 1
   fence
   if (flag0 == 1)
       return false;
   // critical
   flag1 = 0;
   return true;
}
```

Declarative Fence Insertion

```
1 bool p0()
2    flag0 = 1
3    if (flag1 == 1)
4        return false;
5    // critical
6    flag0 = 0;
7    return true;
8 }
```

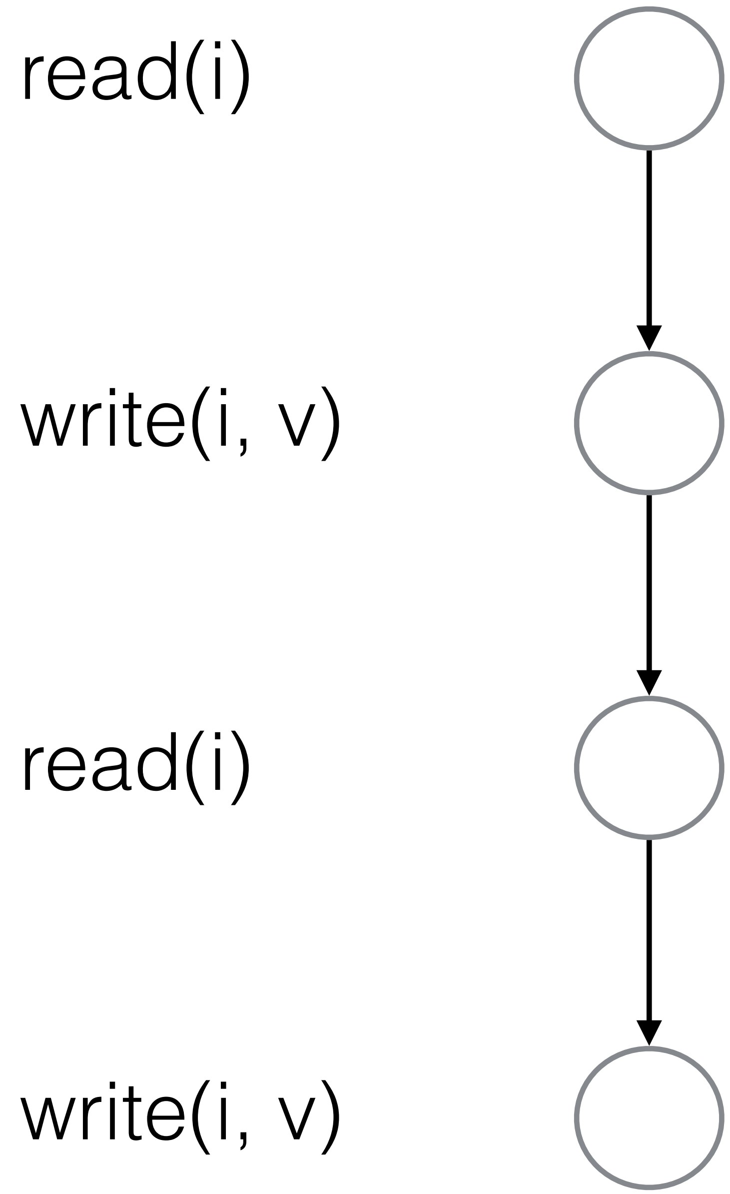
{2 -> 3}

```
1 bool p1()
2    flag1 = 1
3    if (flag0 == 1)
4        return false;
5    // critical
6    flag1 = 0;
7    return true;
8 }
```

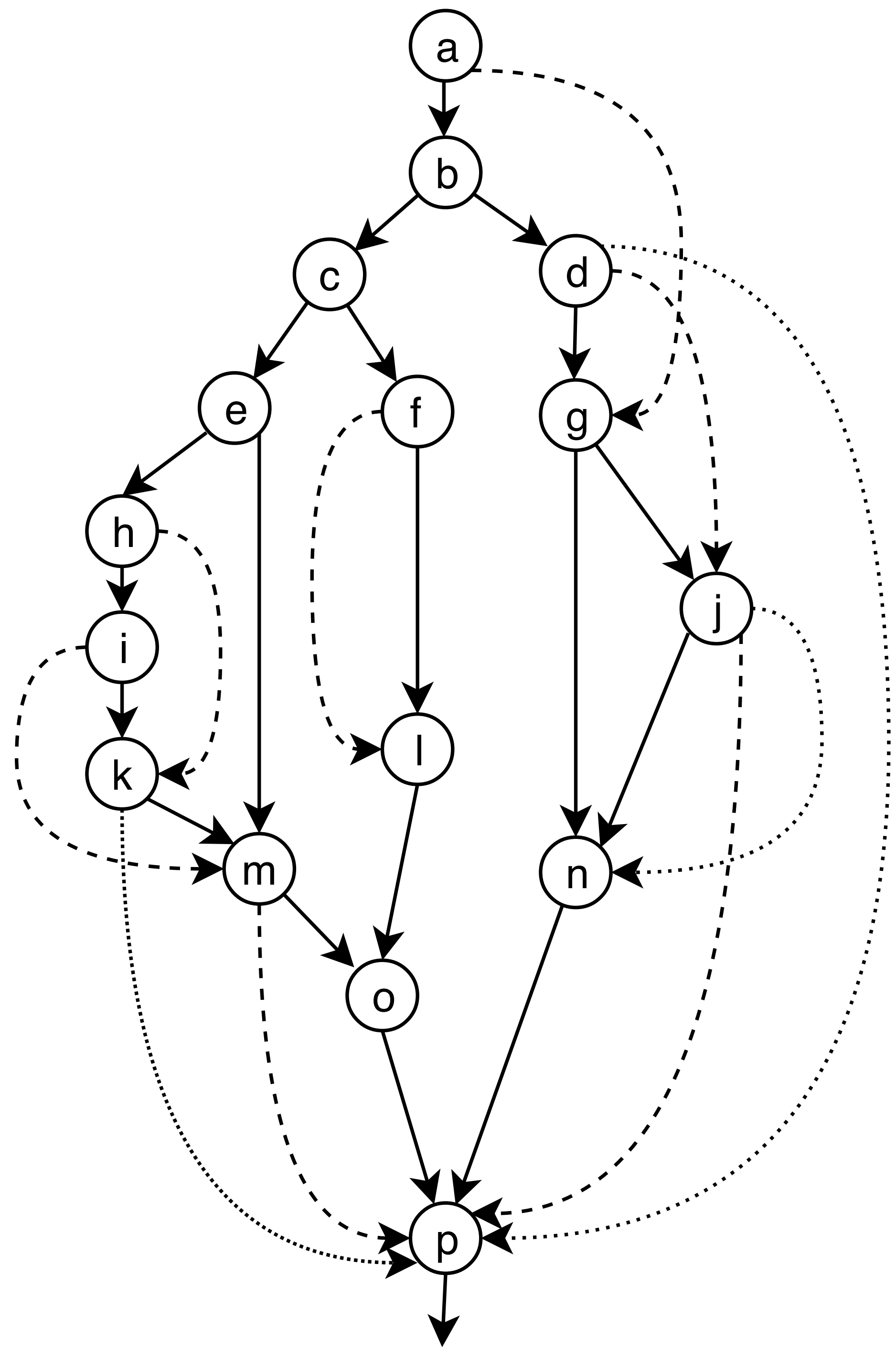
{2 -> 3}

Fence Insertion for Straight-line Programs

- Straight-line Programs
- Polynomial greedy algorithm



Fence Insertion for structured programs



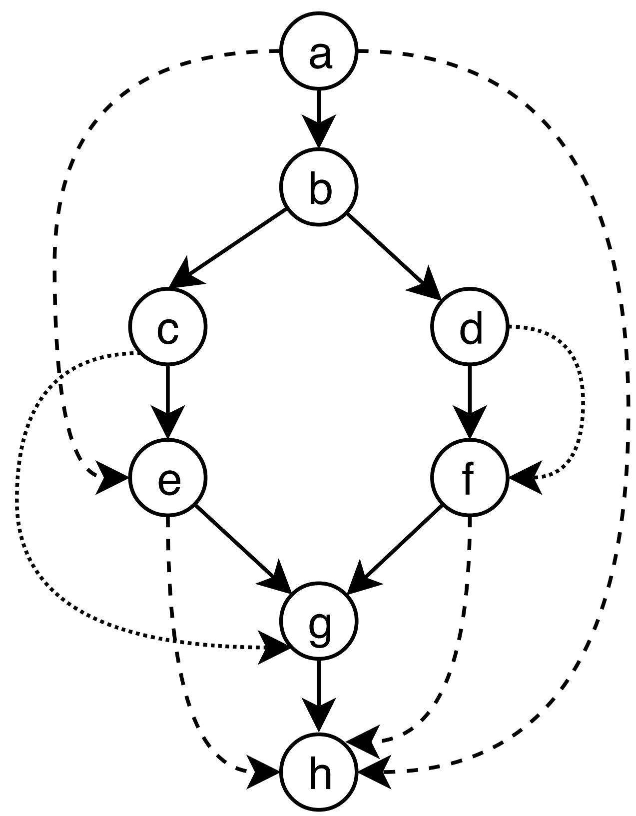
Fence Insertion for structured programs

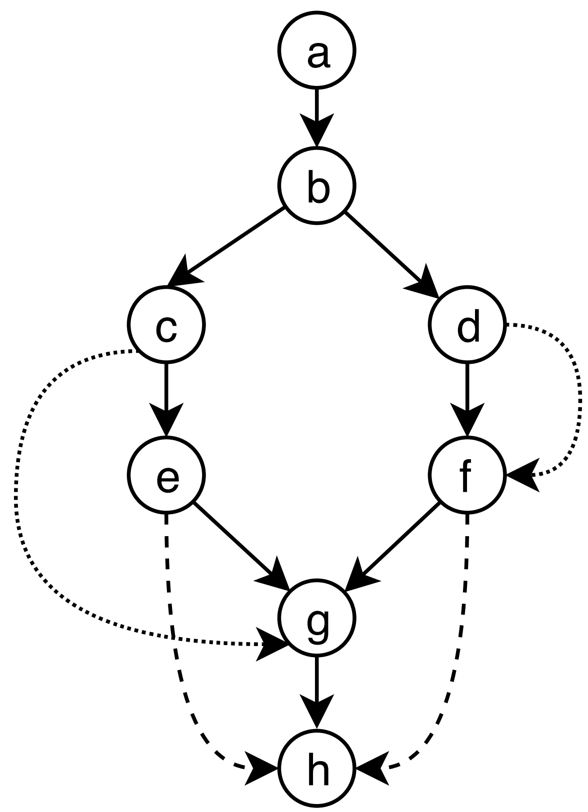
1. A greedy and polynomial-time optimum fence insertion algorithm for Structured programs.
2. The minimum fence insertion problem with multiple types of fence instructions is NP-hard.

Fence Insertion for loop-free programs

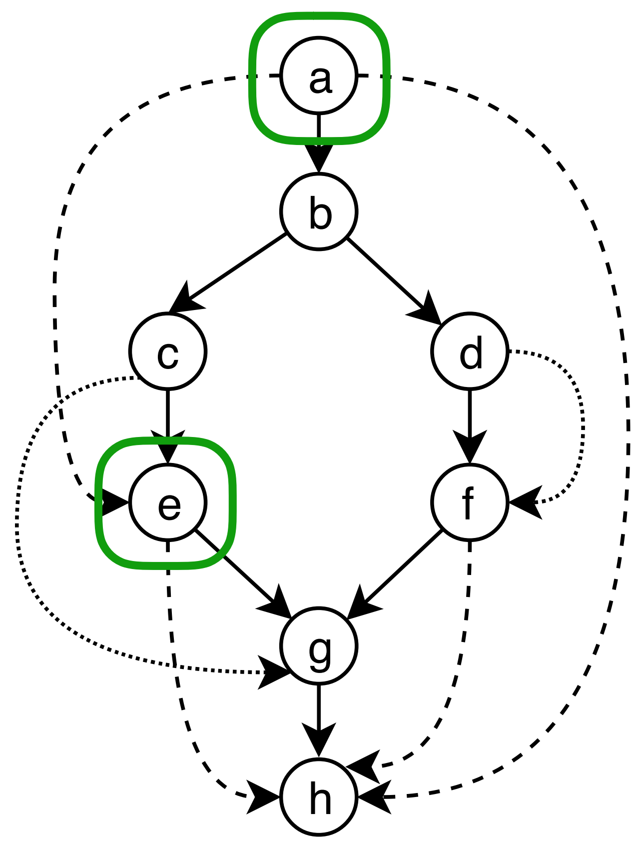
1. Constraint Elimination
2. Finding Diamonds
3. Diamond Decomposition
4. Fence Insertion for Simple Paths

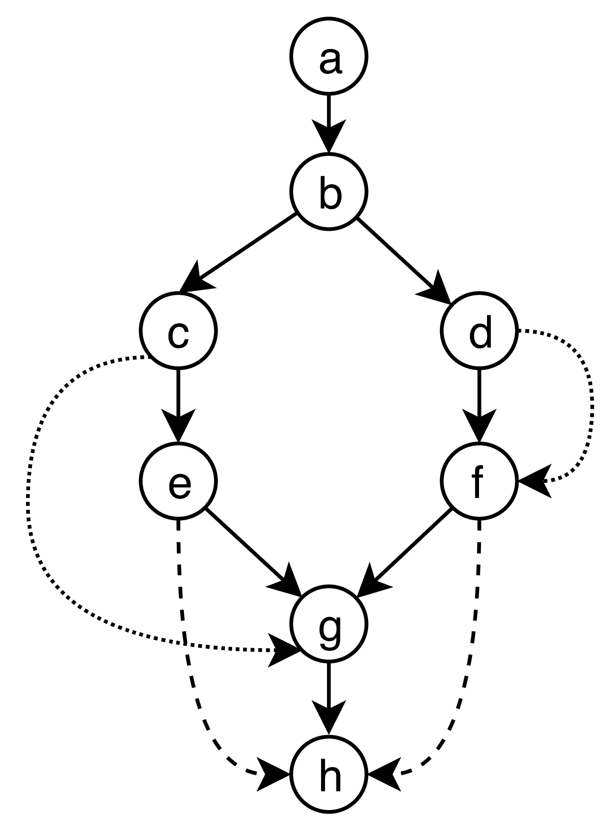
Constraint Elimination



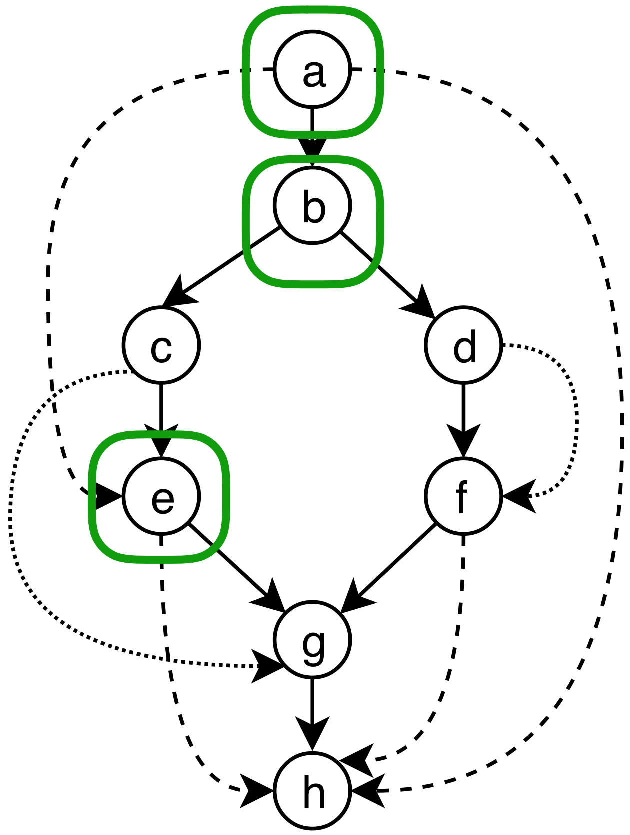


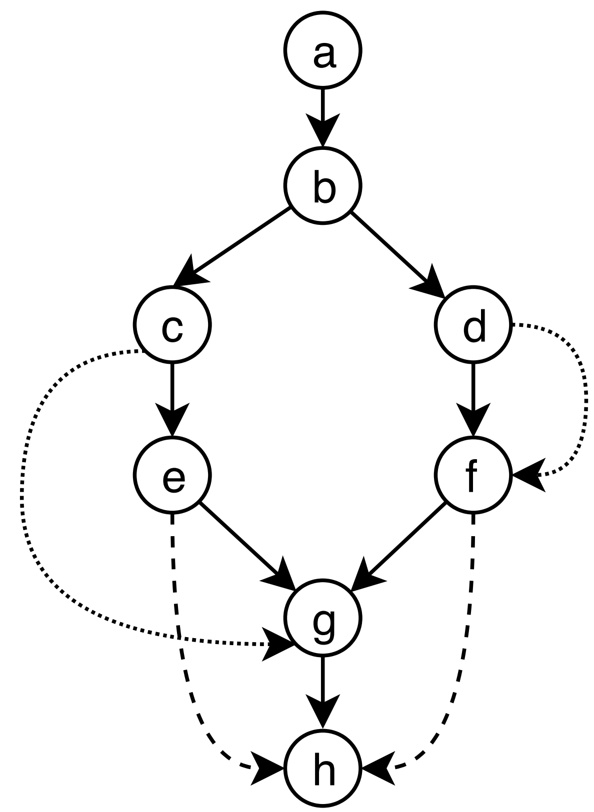
Constraint Elimination



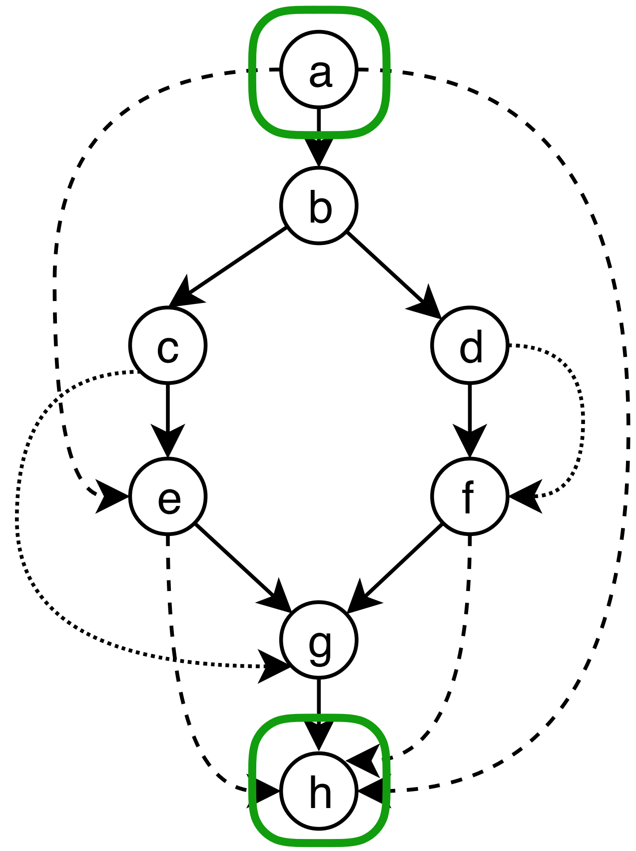


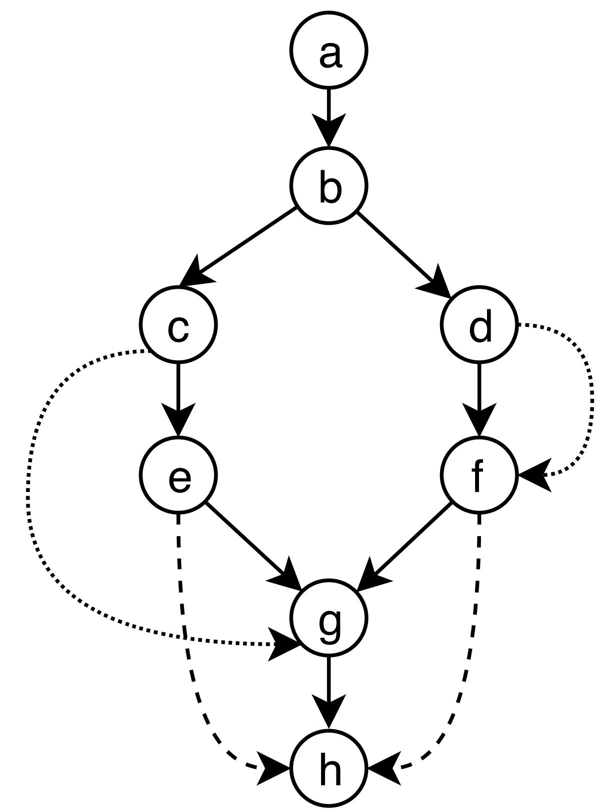
Constraint Elimination



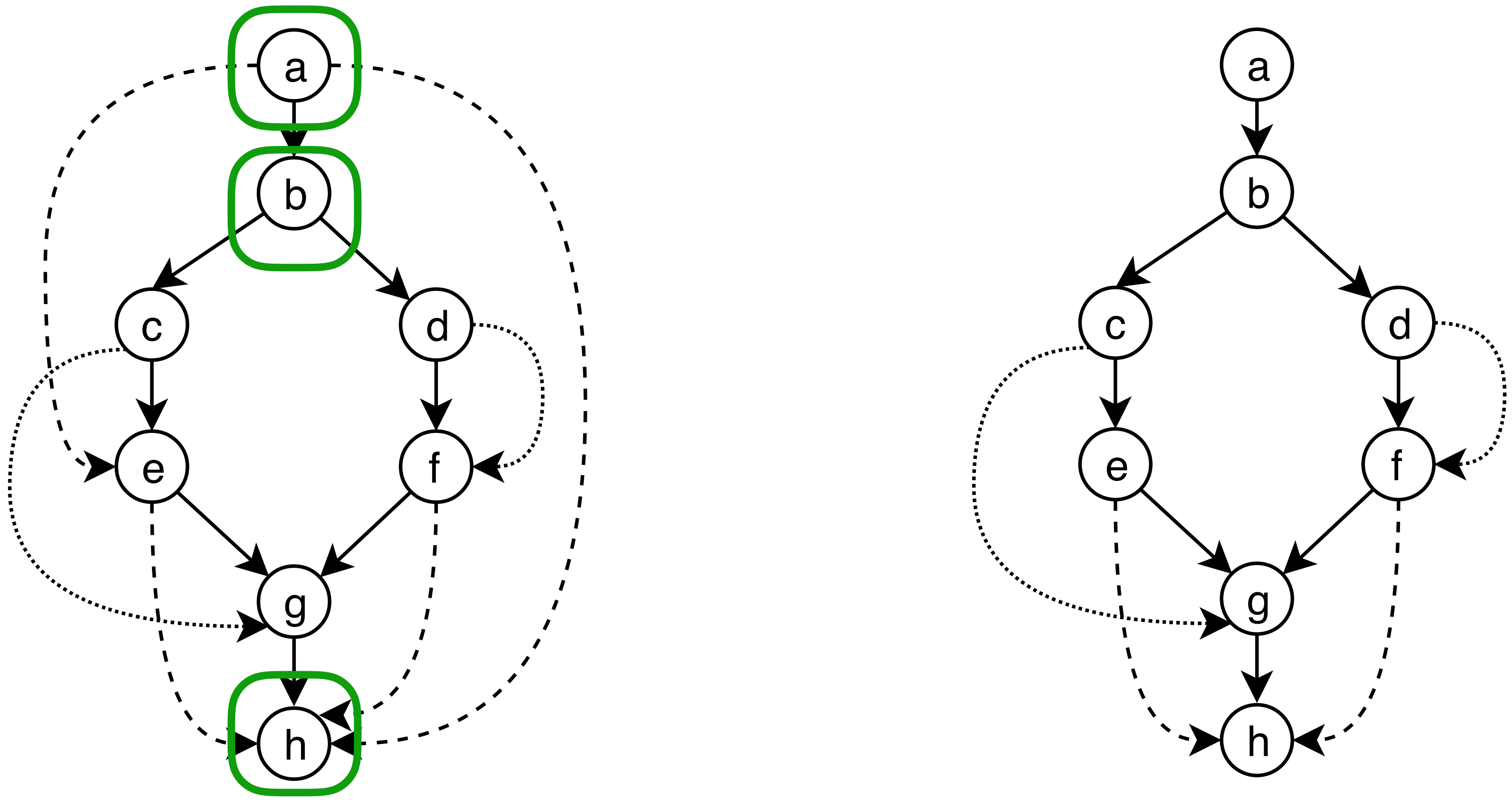


Constraint Elimination

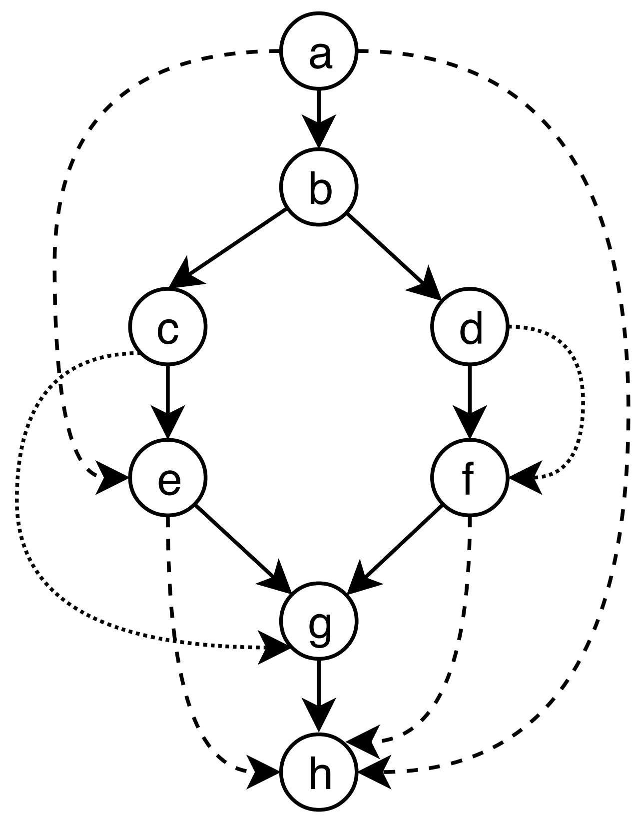


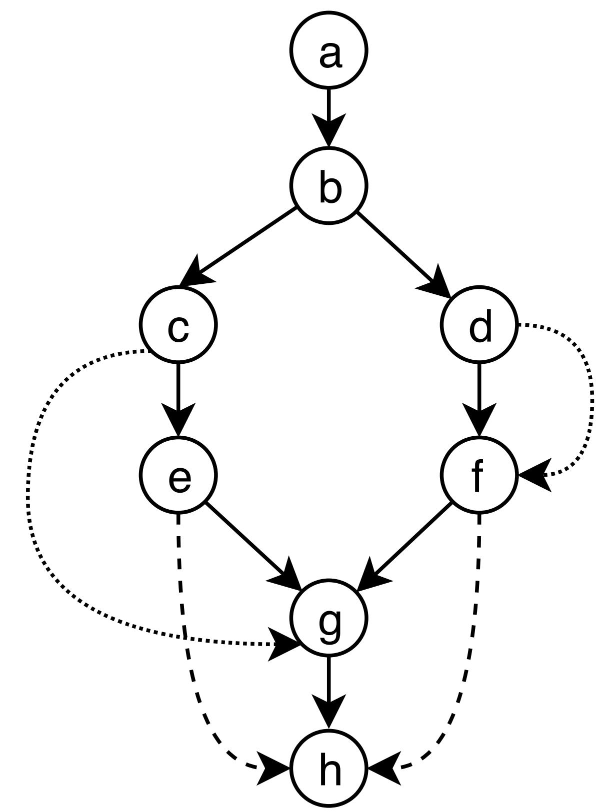


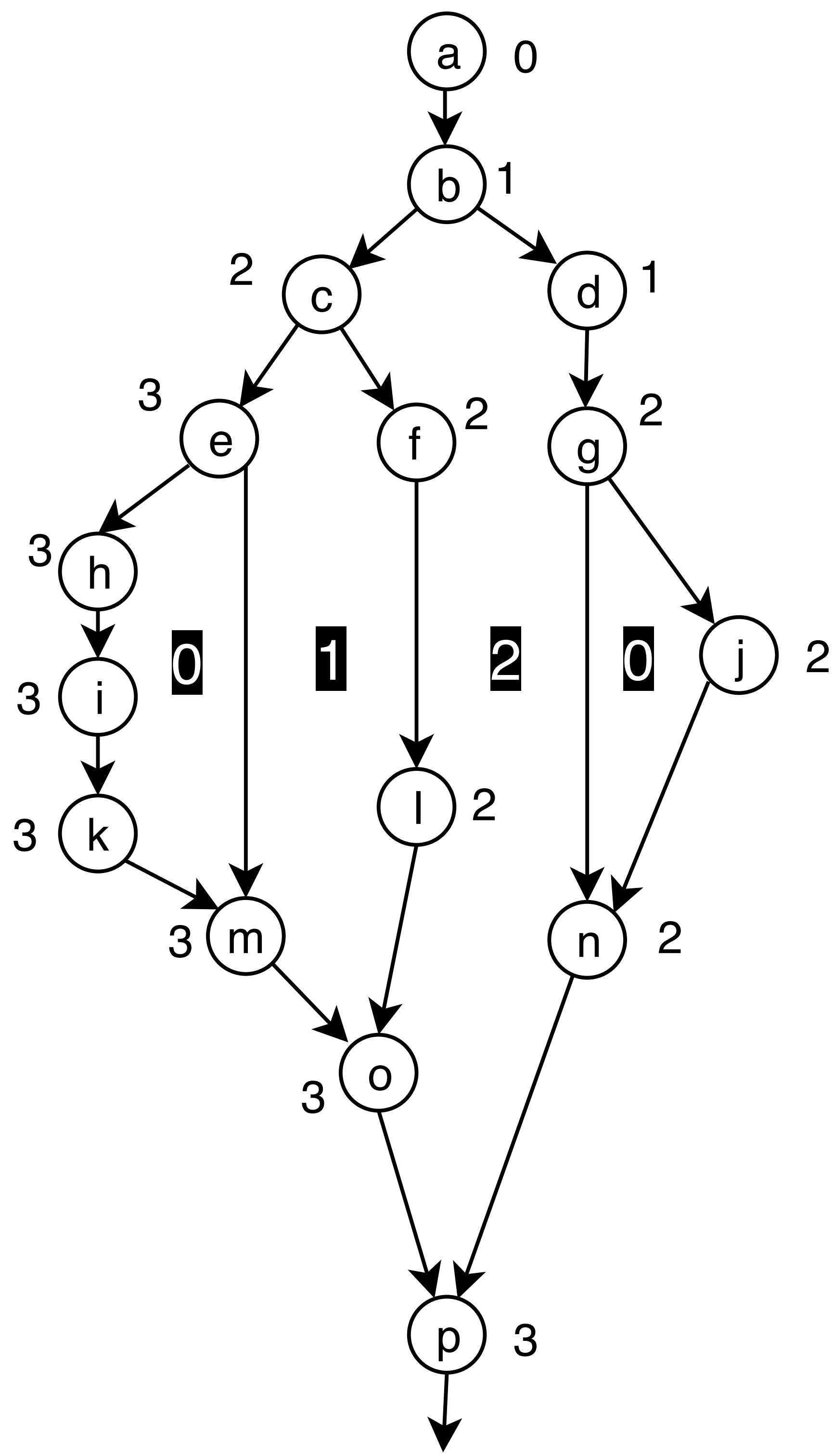
Constraint Elimination



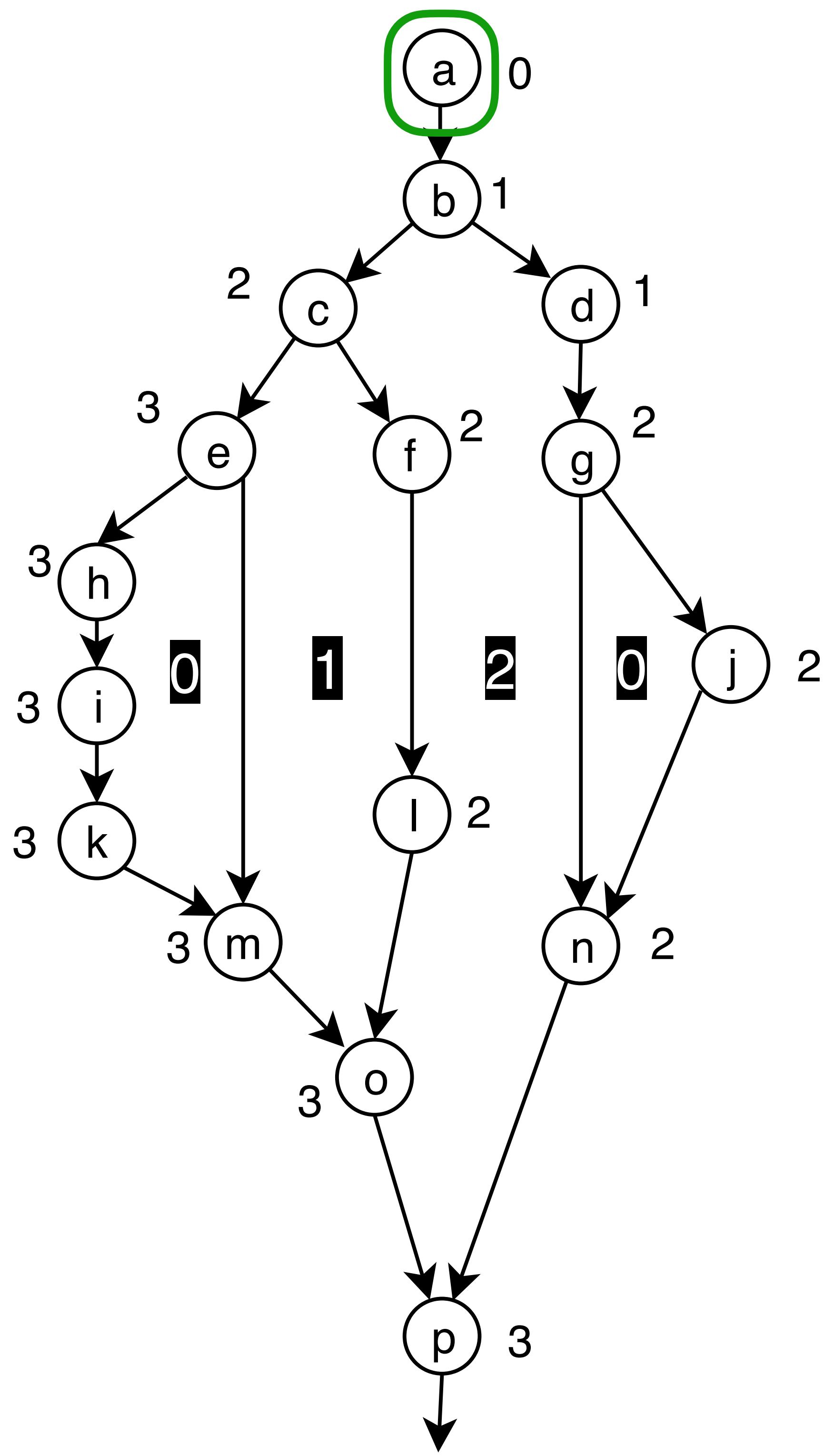
Constraint Elimination



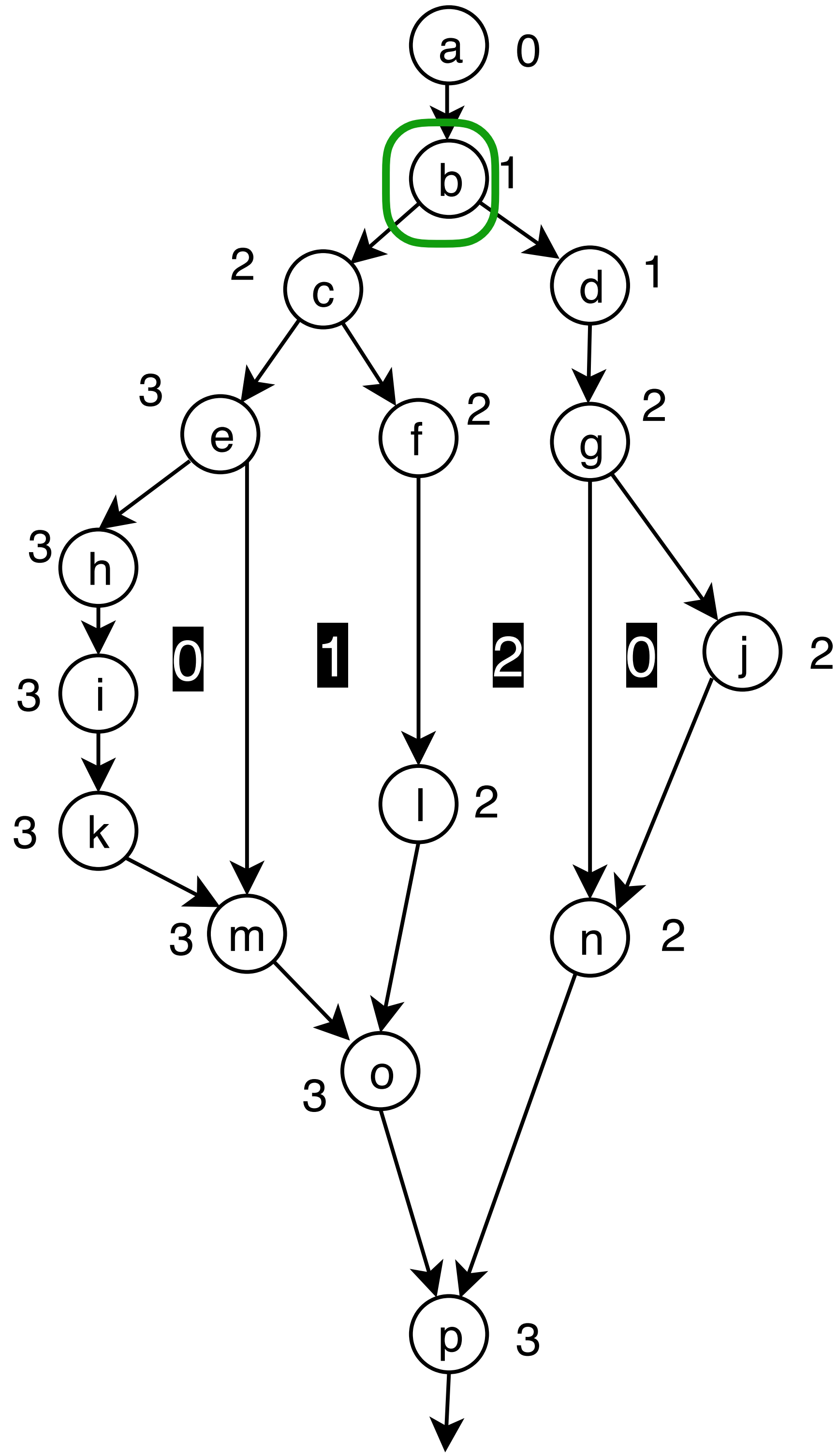



a 0
b 1
c 2
d 1
e 3
f 2
g 2
h 3
i 3
j 2
k 3
l 2
m 3
n 2
o 3
p 3
0
1
2
0

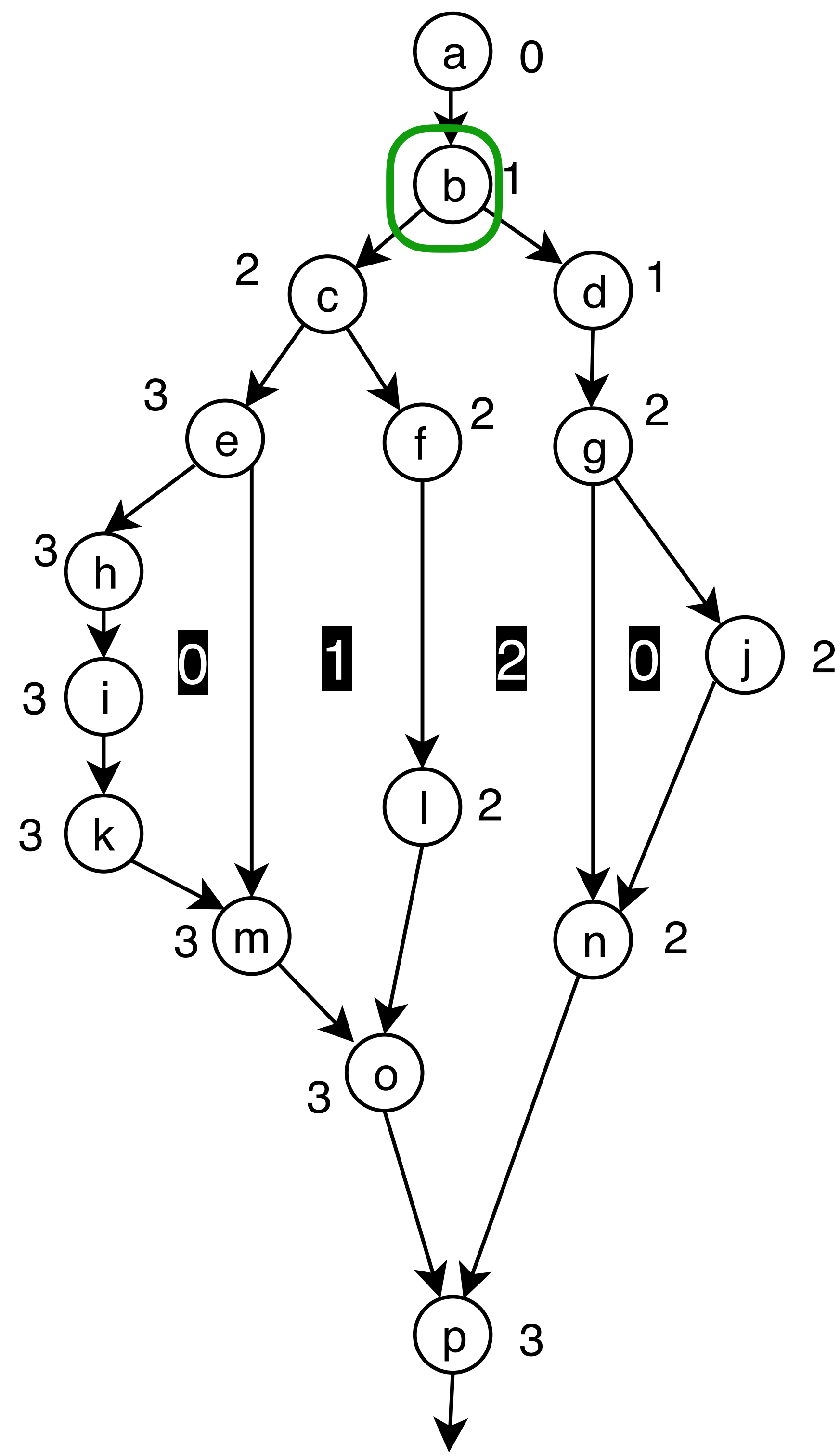
Diamonds

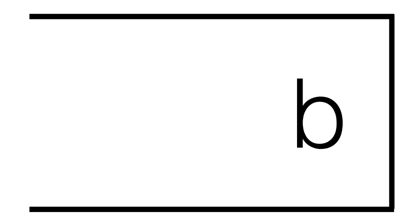


Diamonds

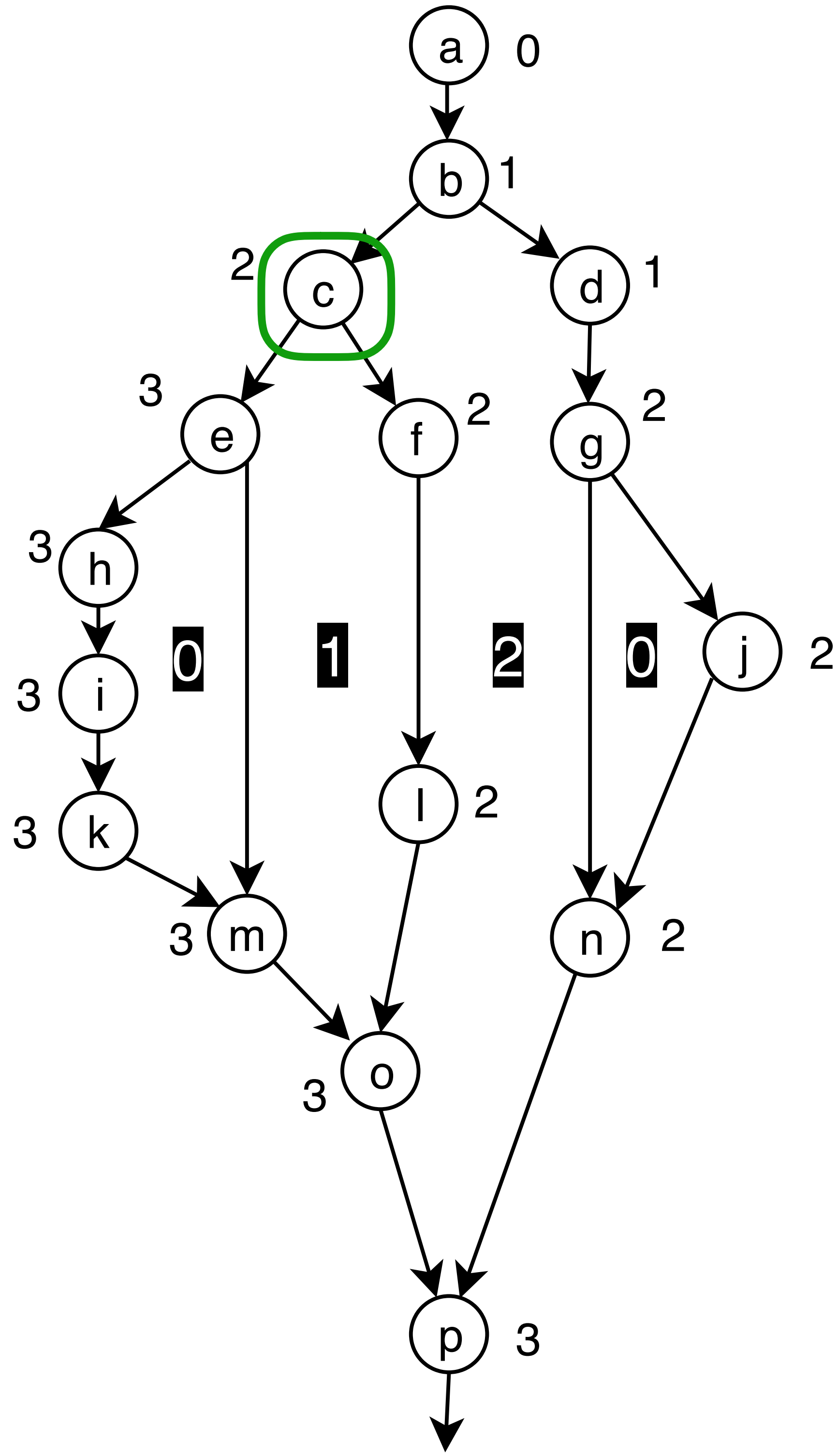


Diamonds

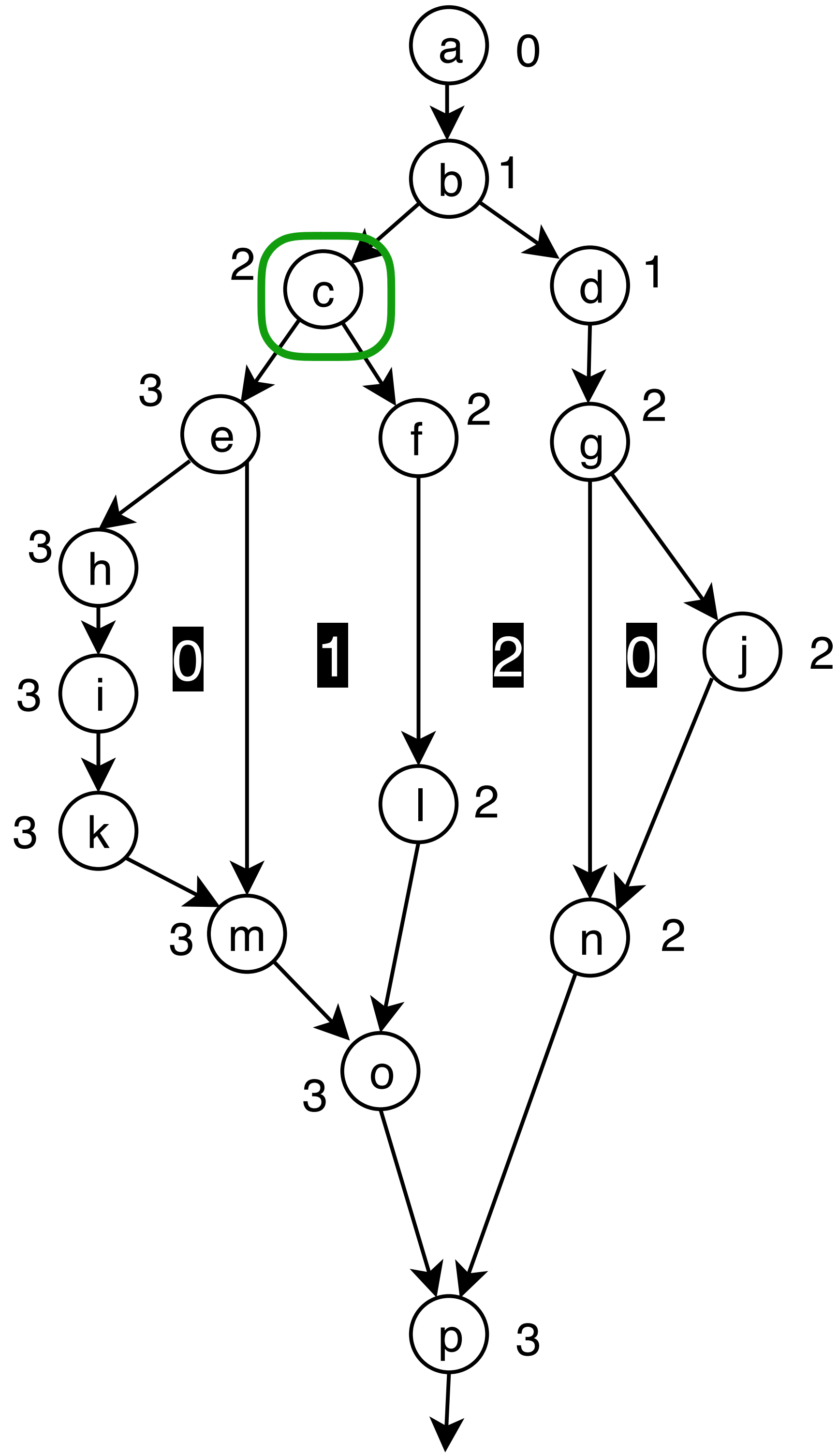


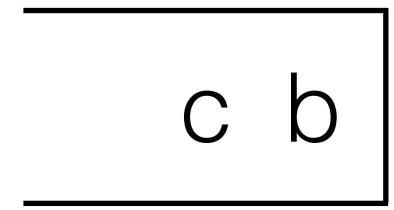


Diamonds

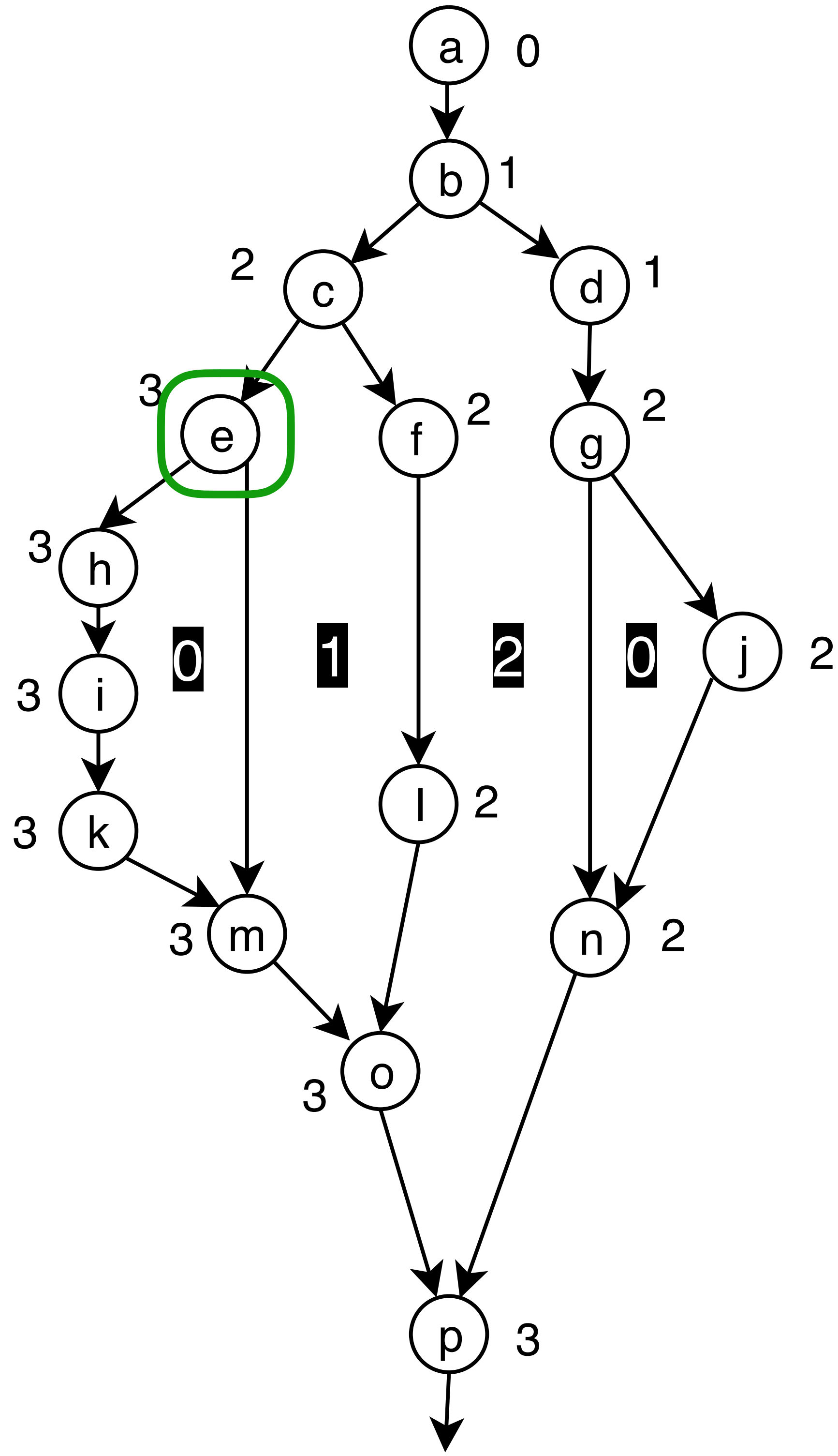


Diamonds



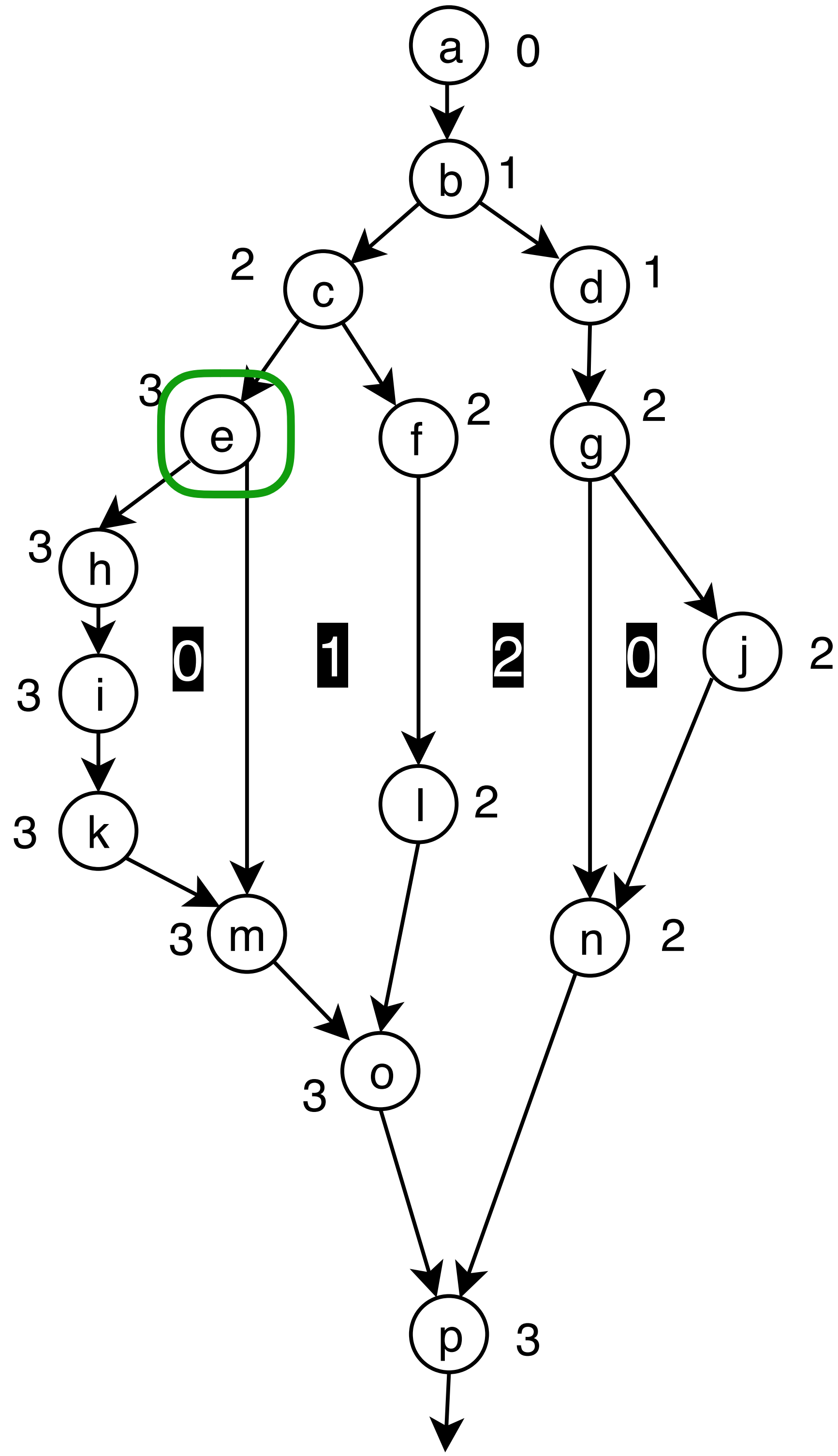


Diamonds



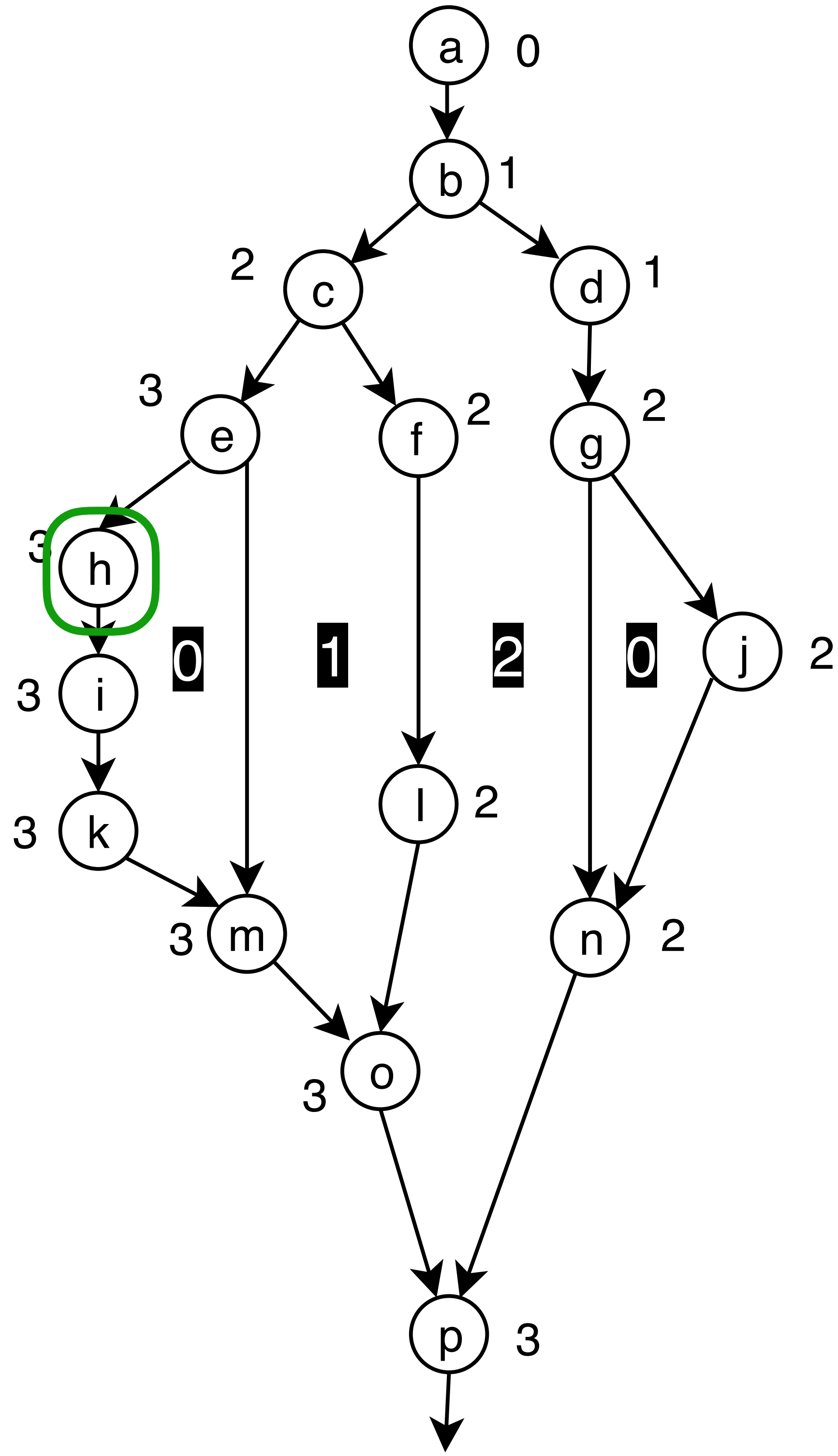
c b

Diamonds



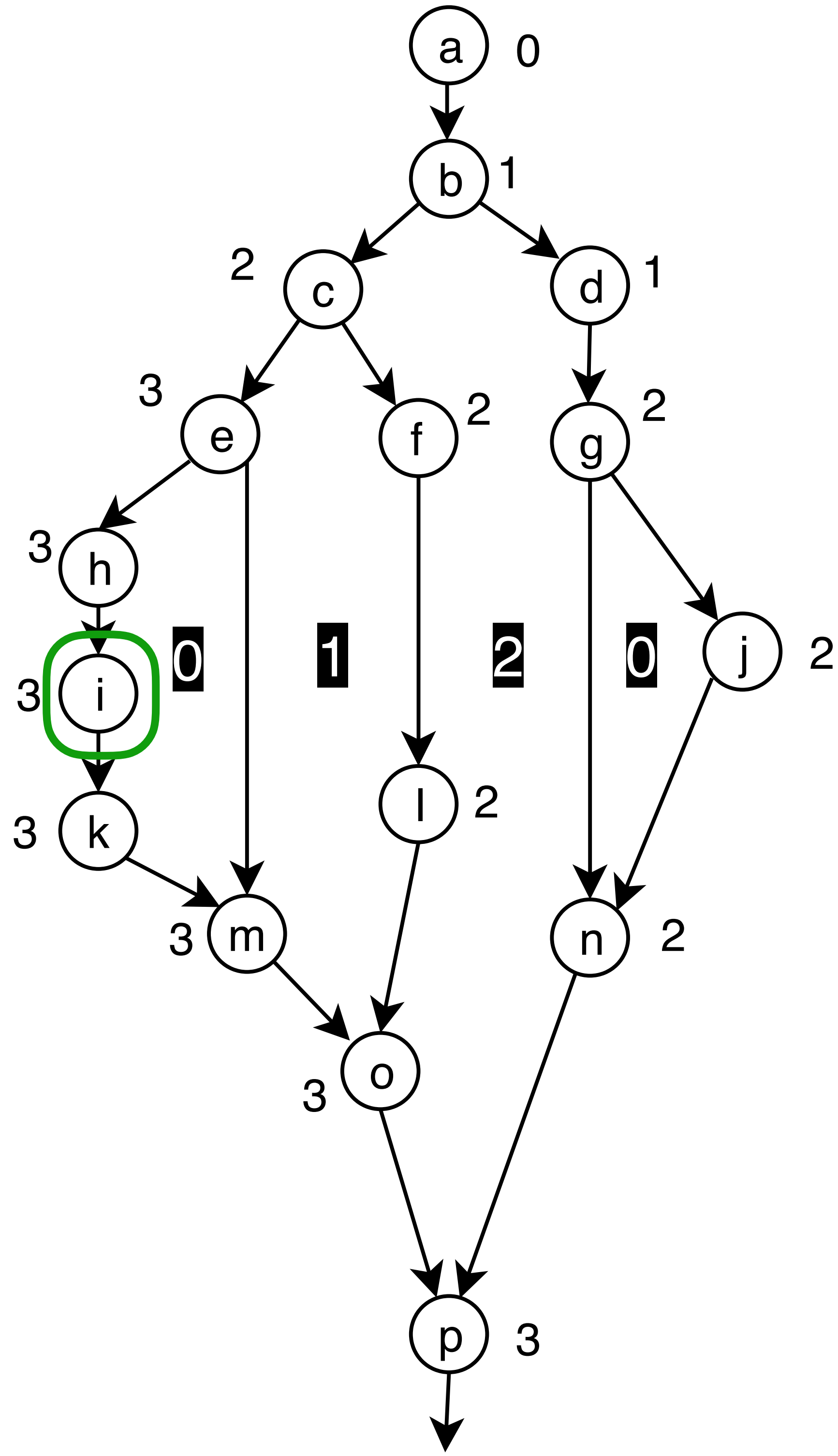
e c b

Diamonds



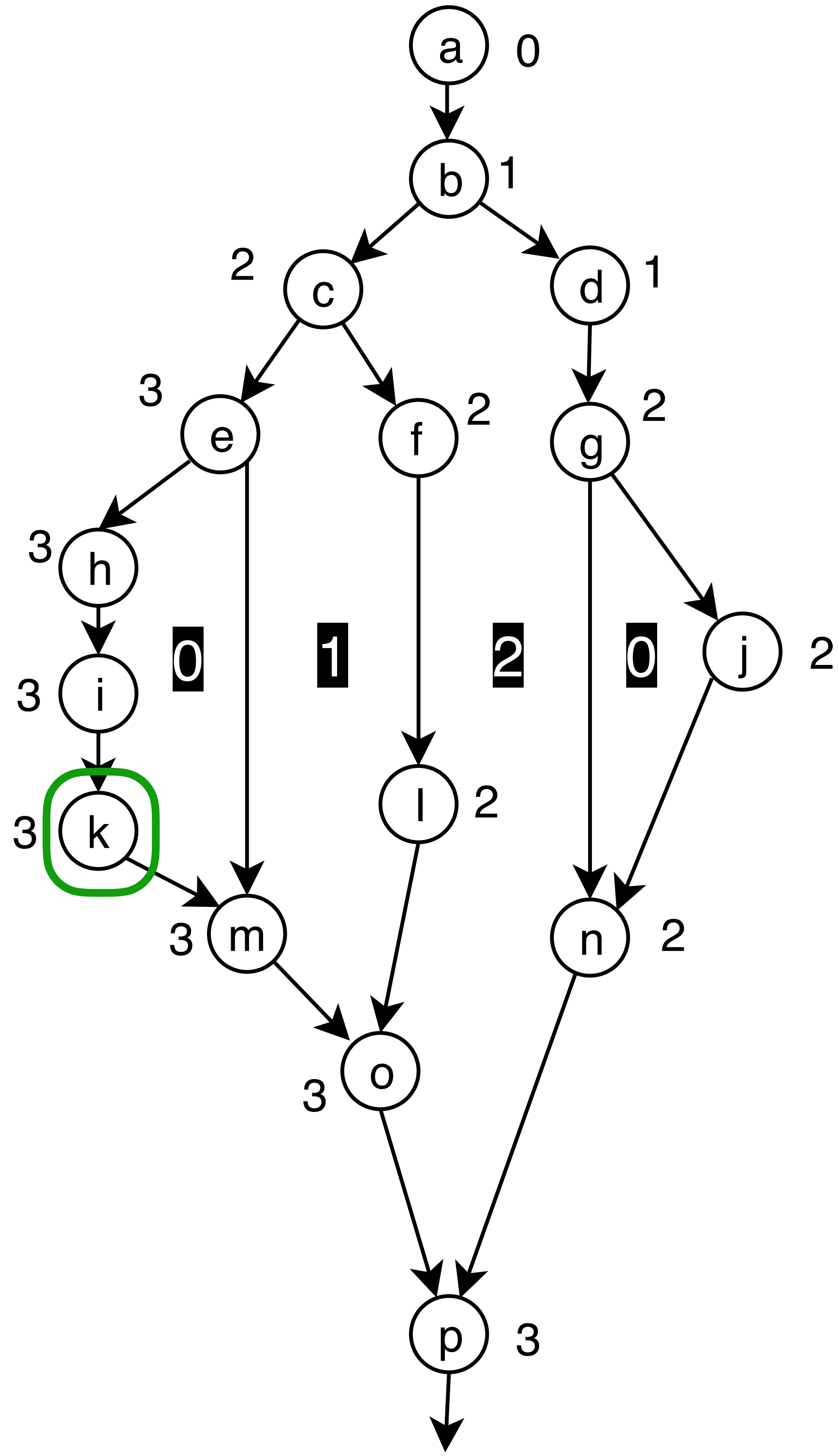
e c b

Diamonds



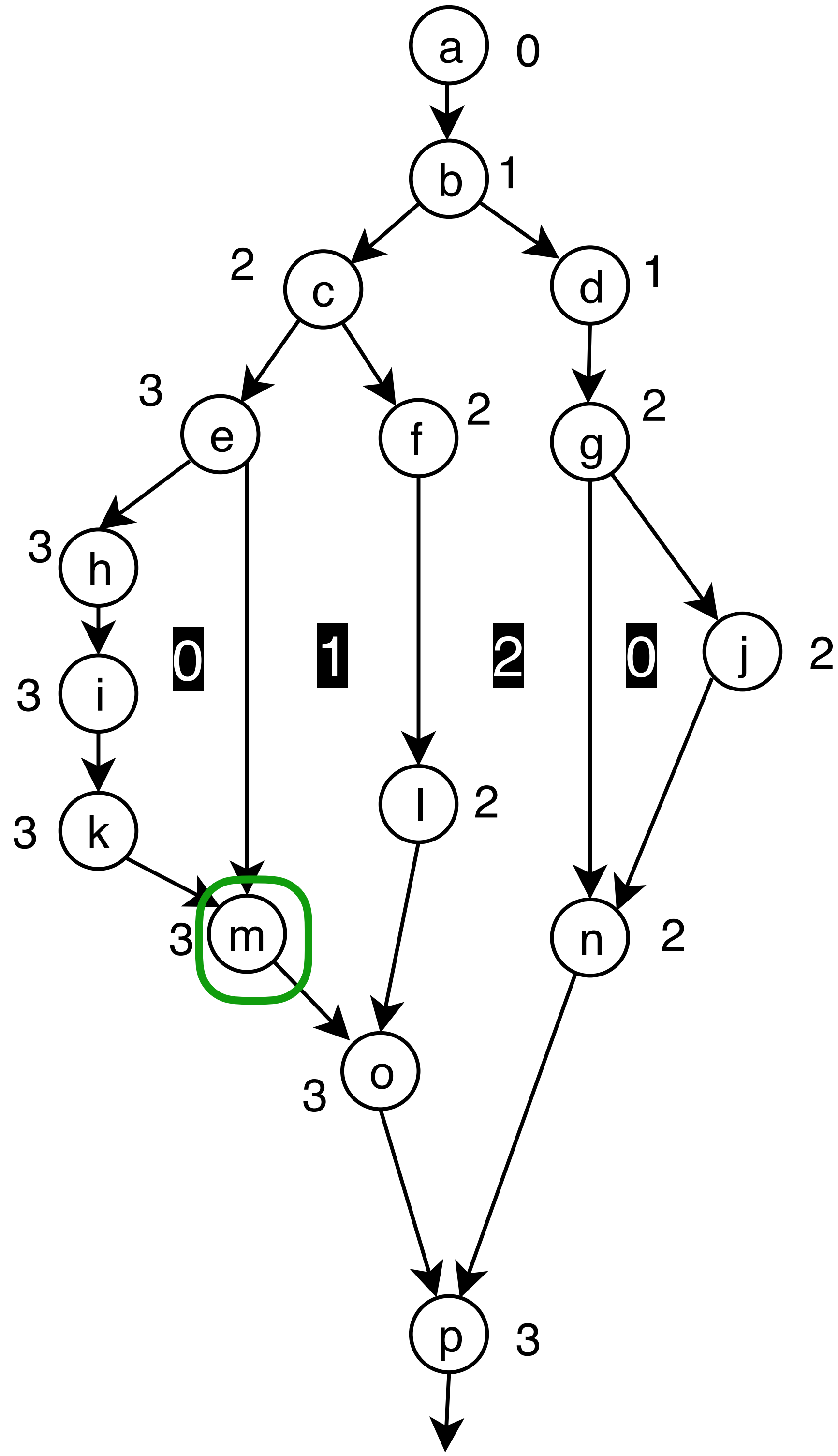
e c b

Diamonds



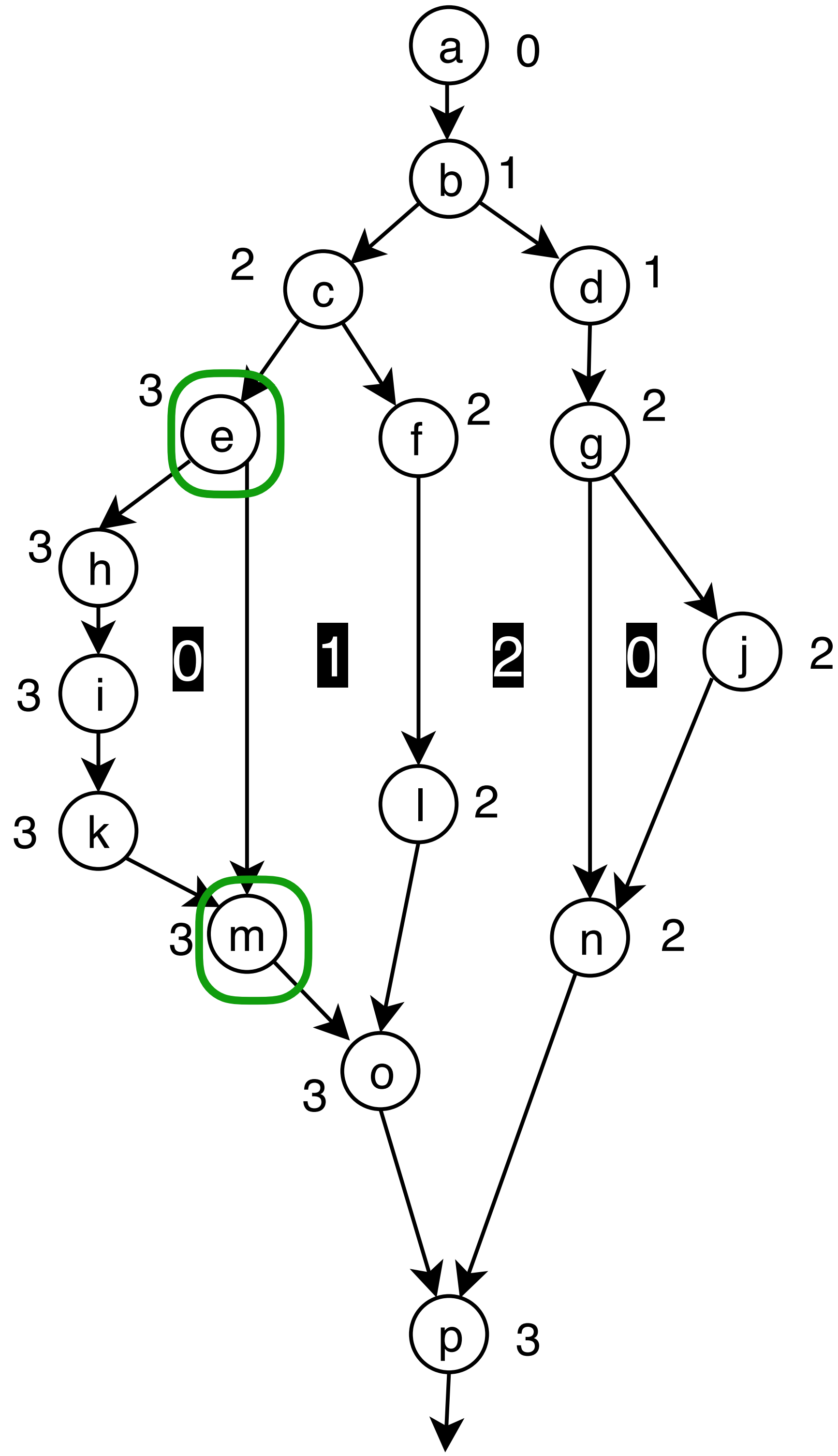
e c b

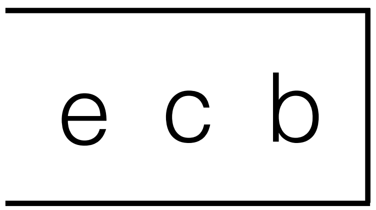
Diamonds



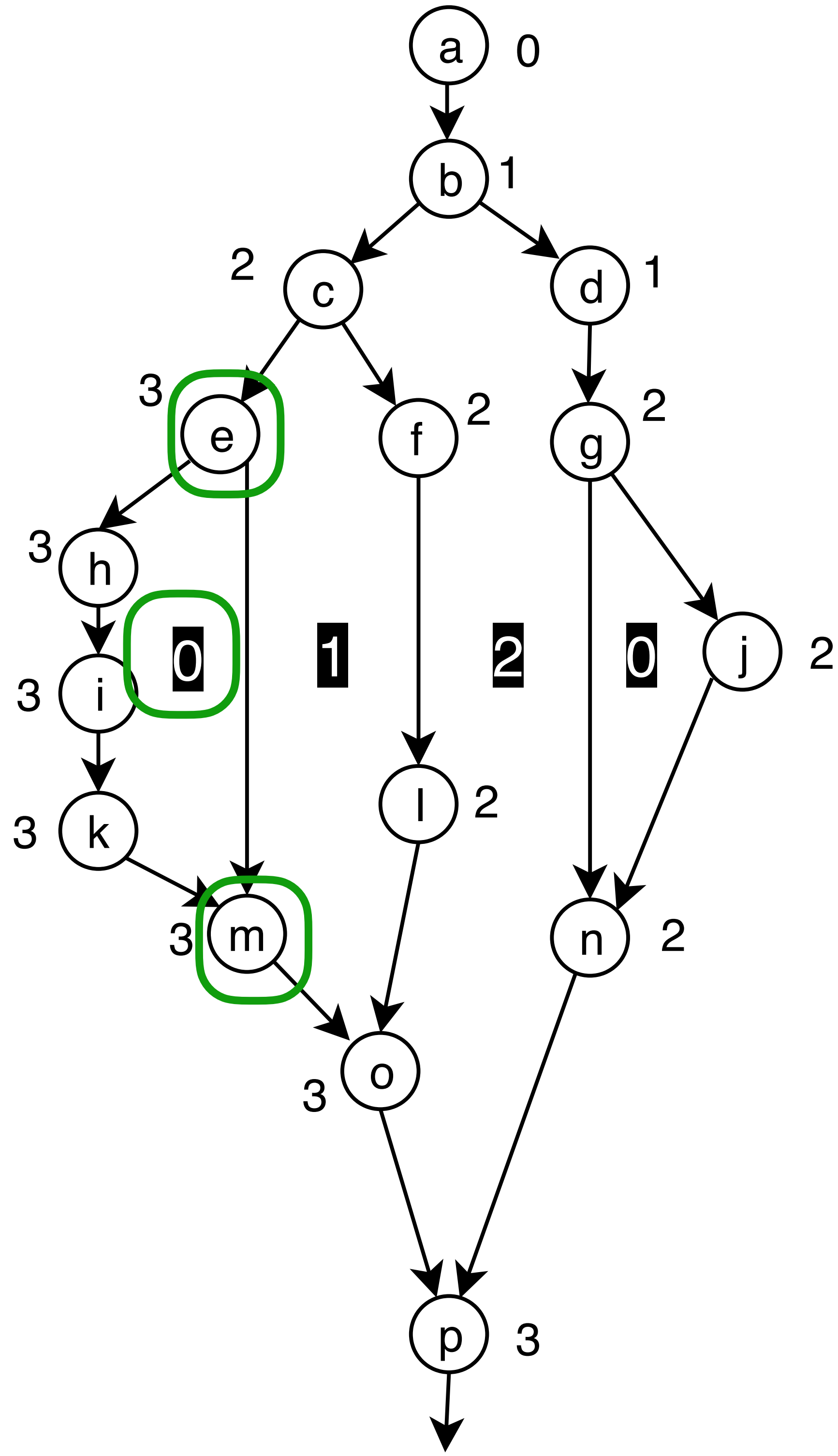
e c b

Diamonds



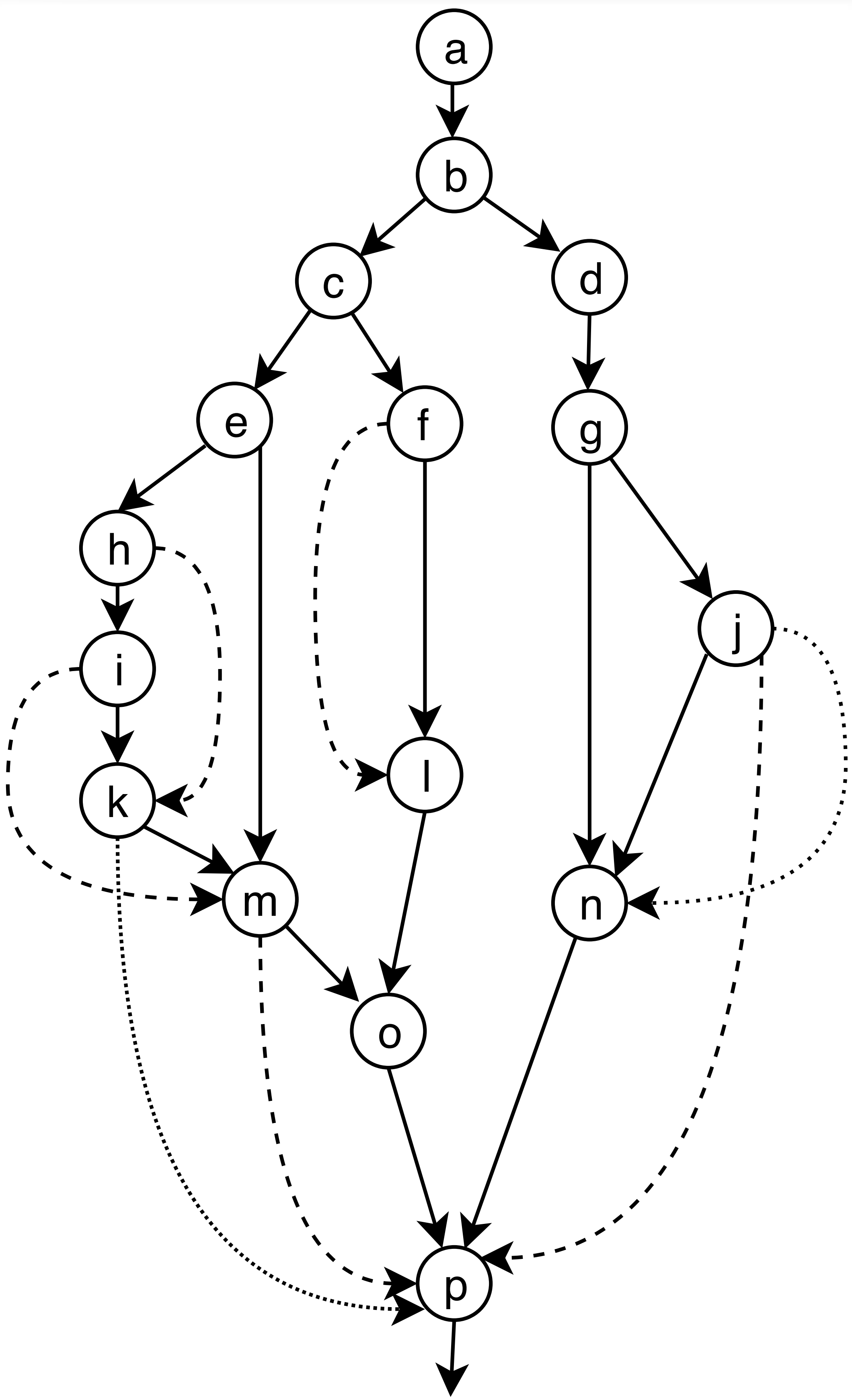


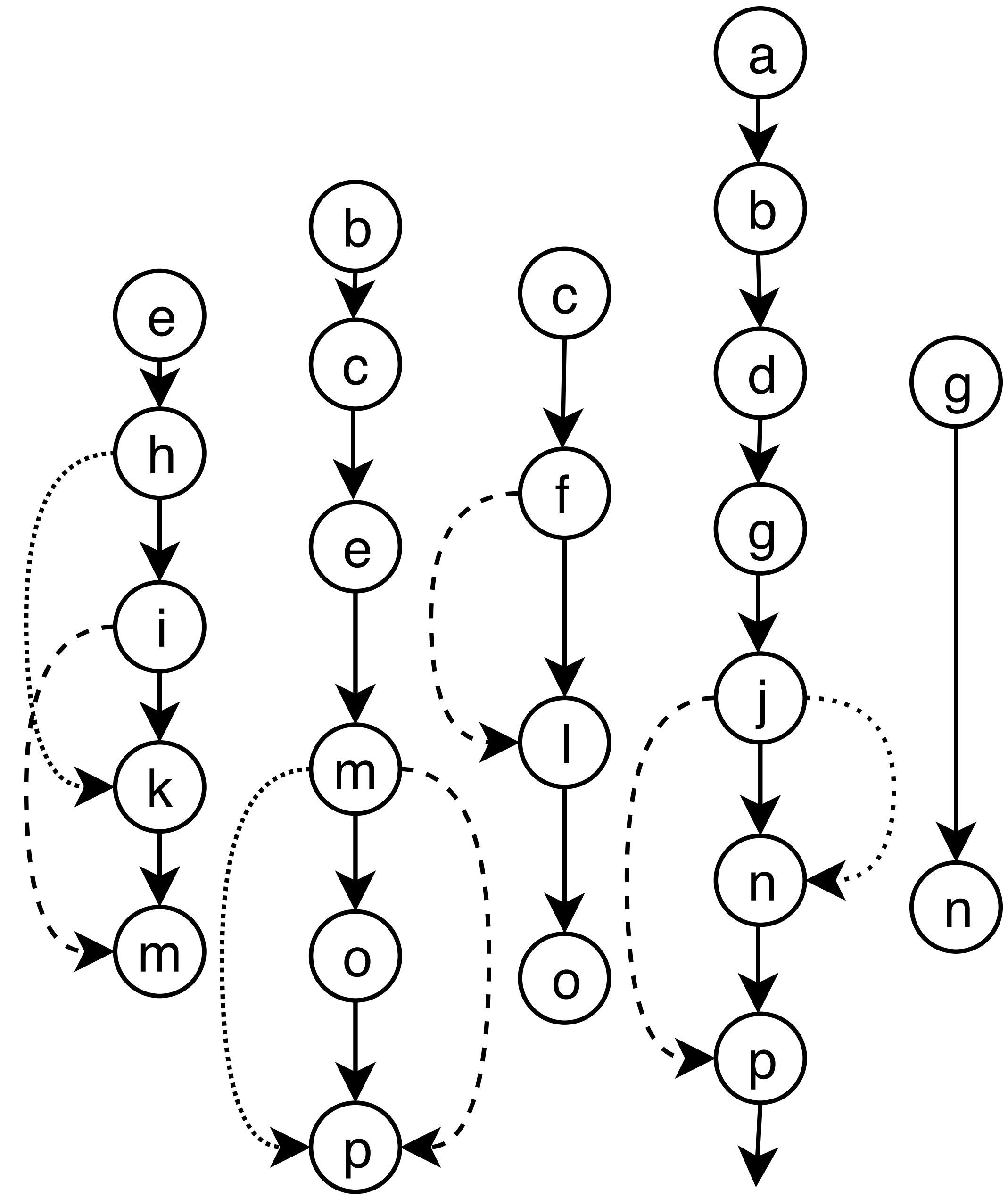
Diamonds



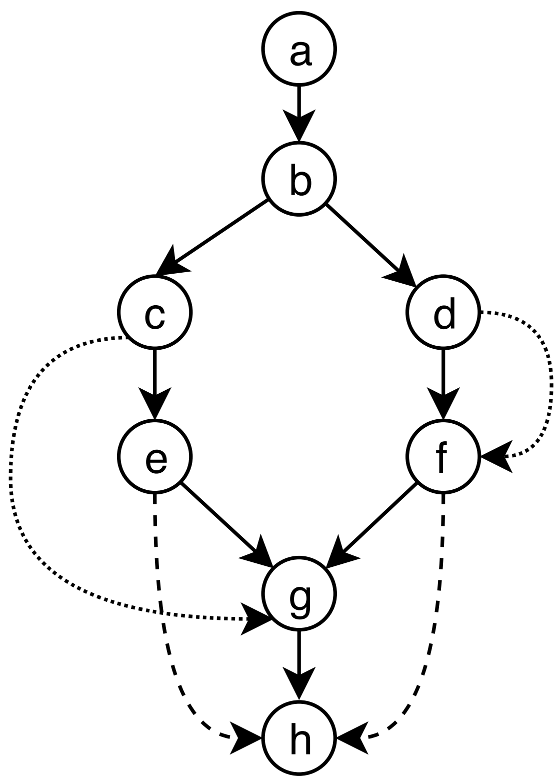
c b

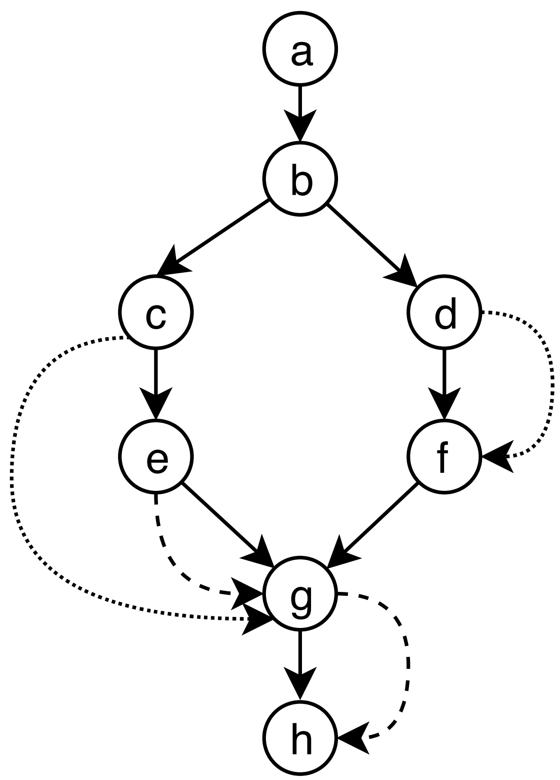
Decomposing Diamonds to Simple Paths



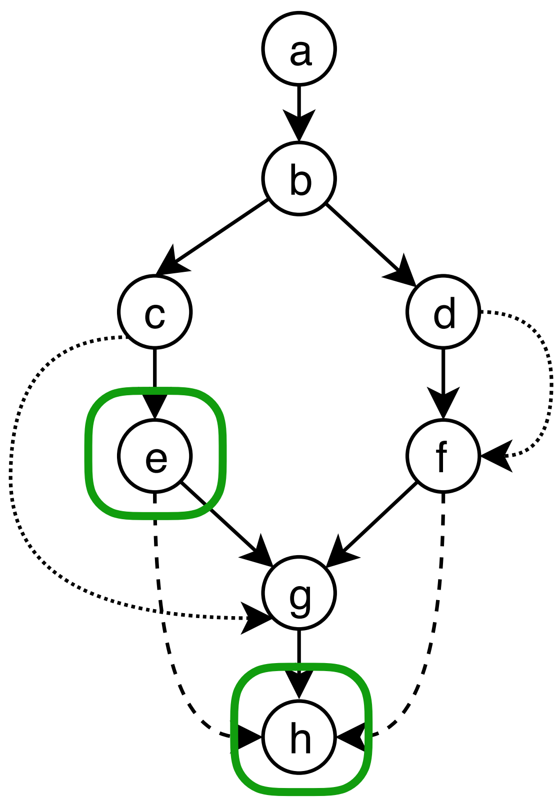


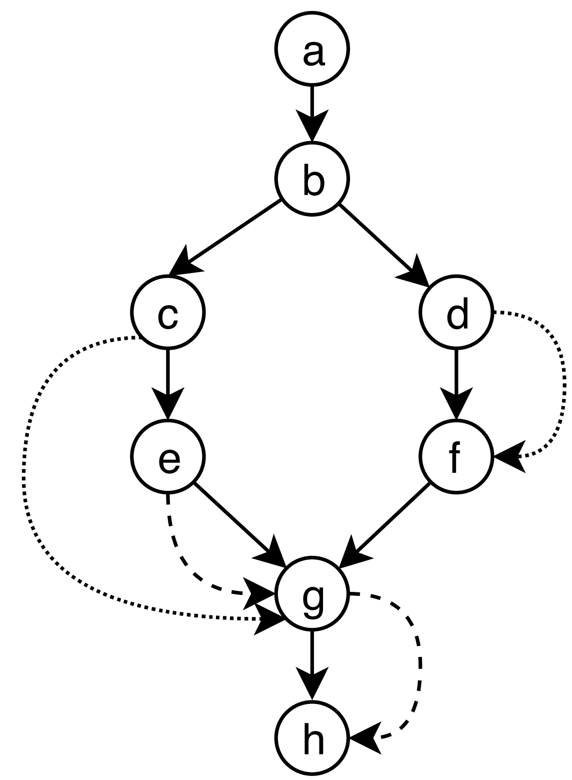
Diamond Decomposition: Absorbing and Emitting Paths



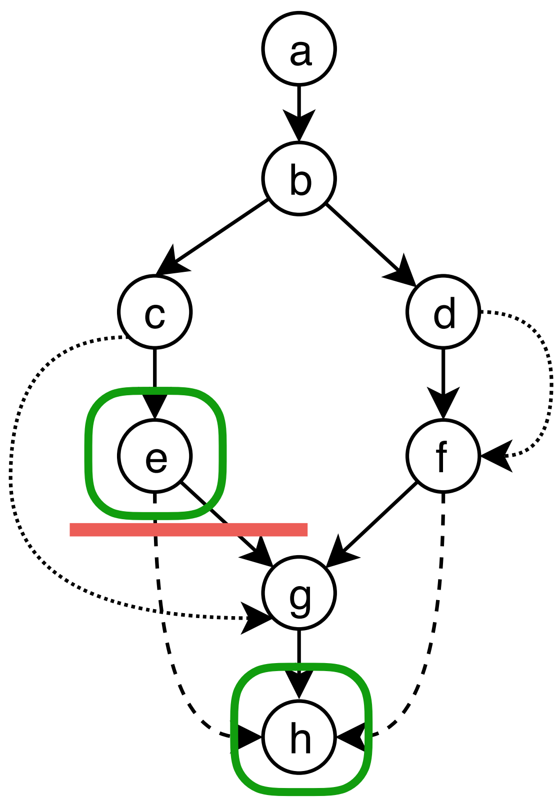


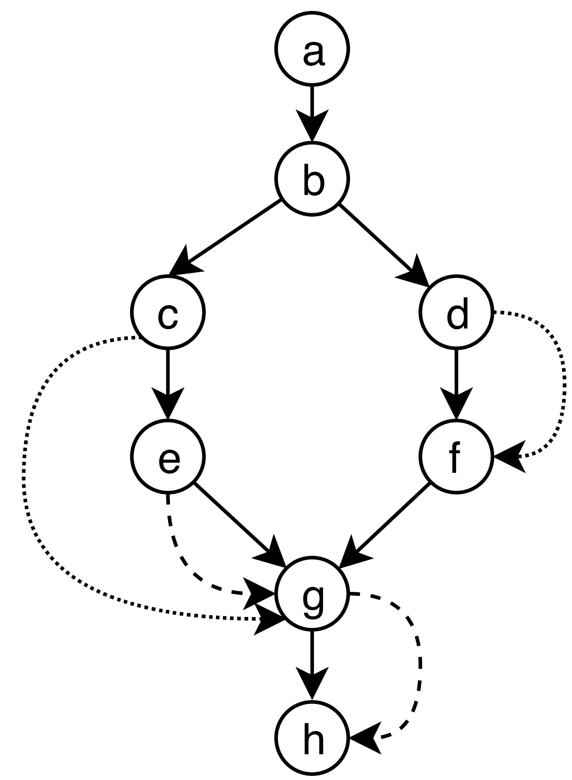
Diamond Decomposition: Absorbing and Emitting Paths



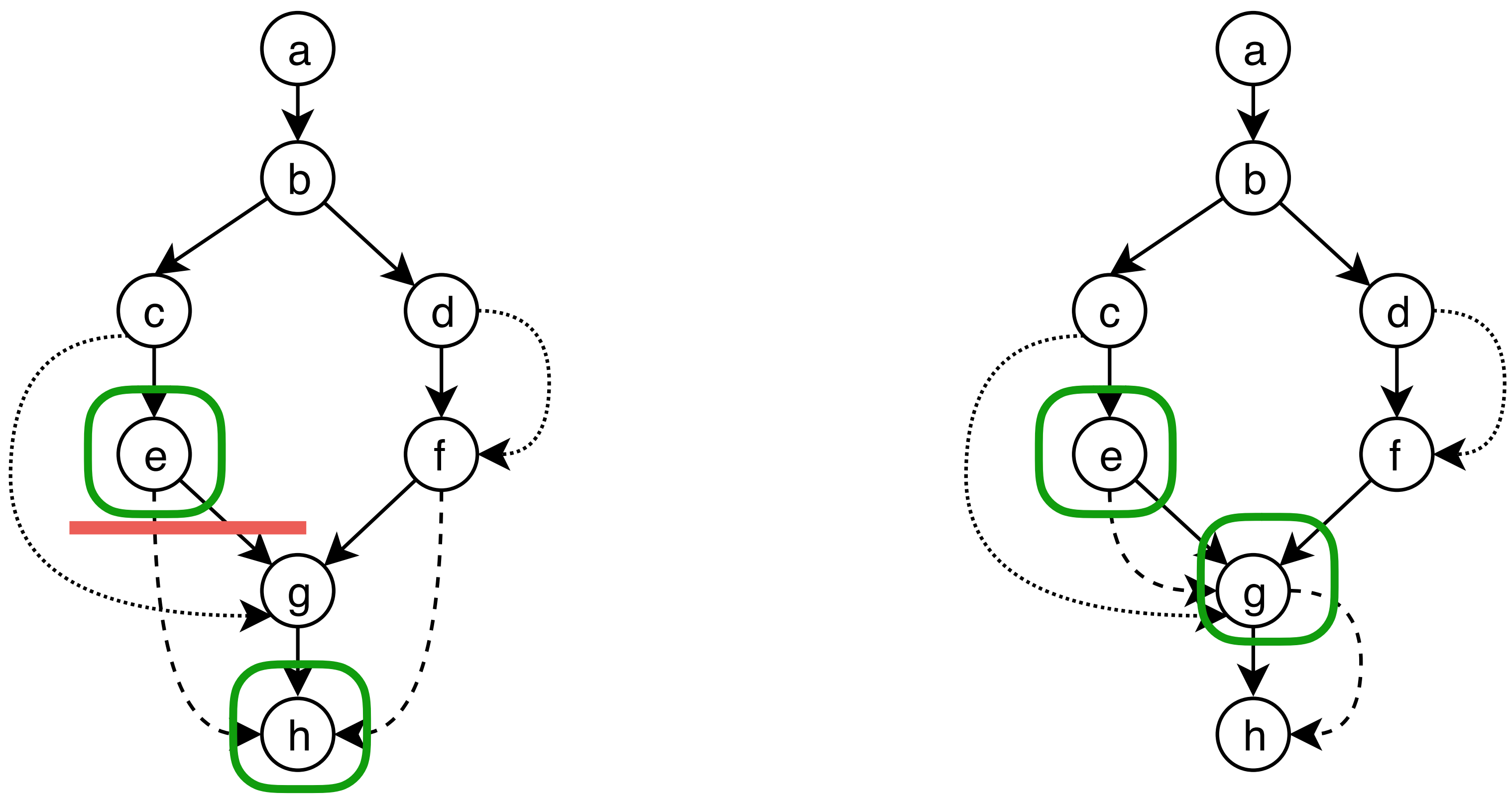


Diamond Decomposition: Absorbing and Emitting Paths

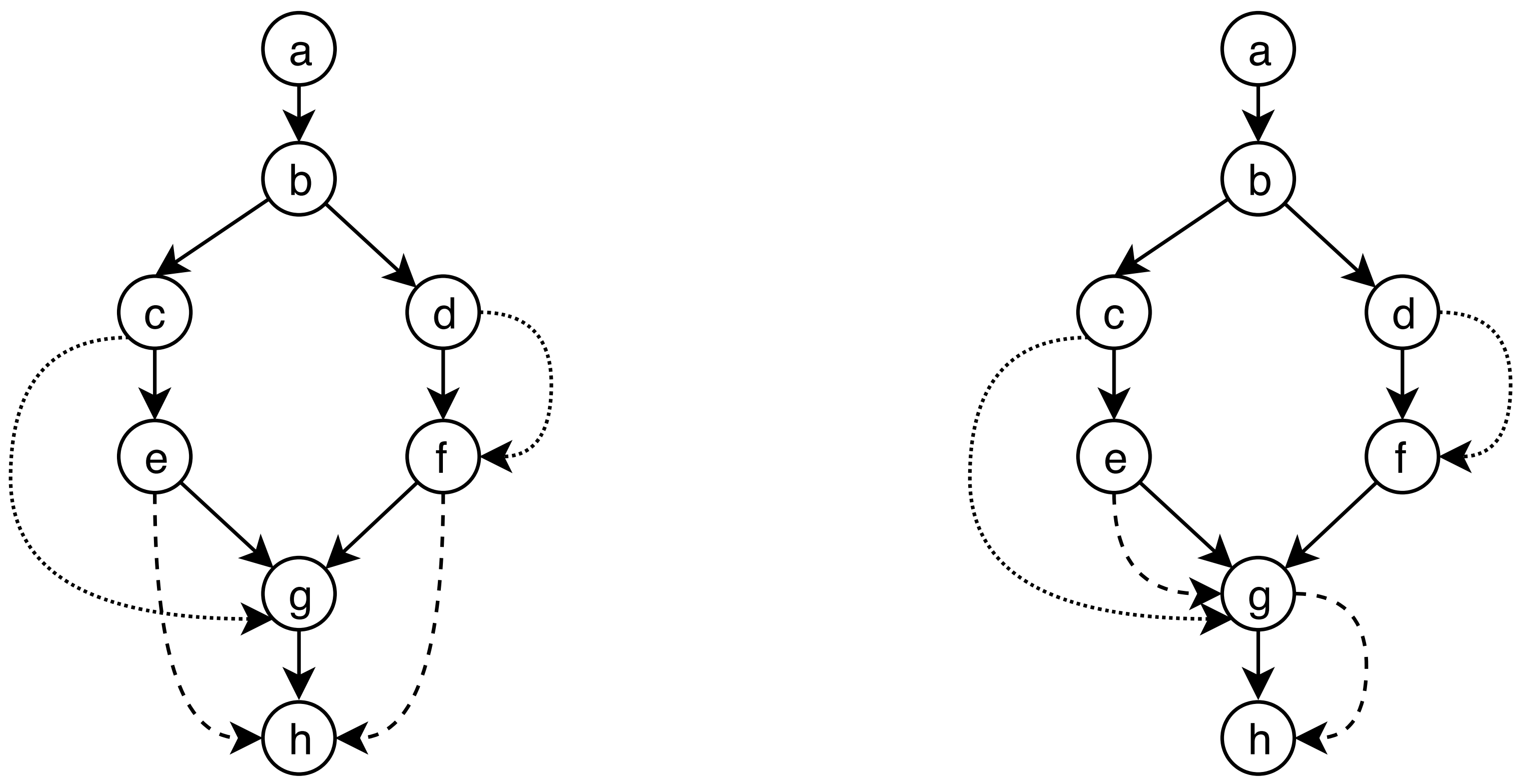




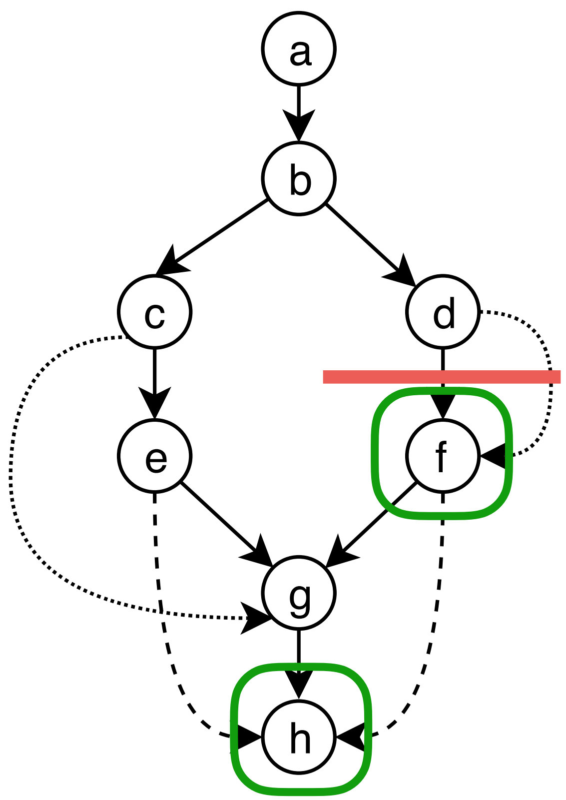
Diamond Decomposition: Absorbing and Emitting Paths

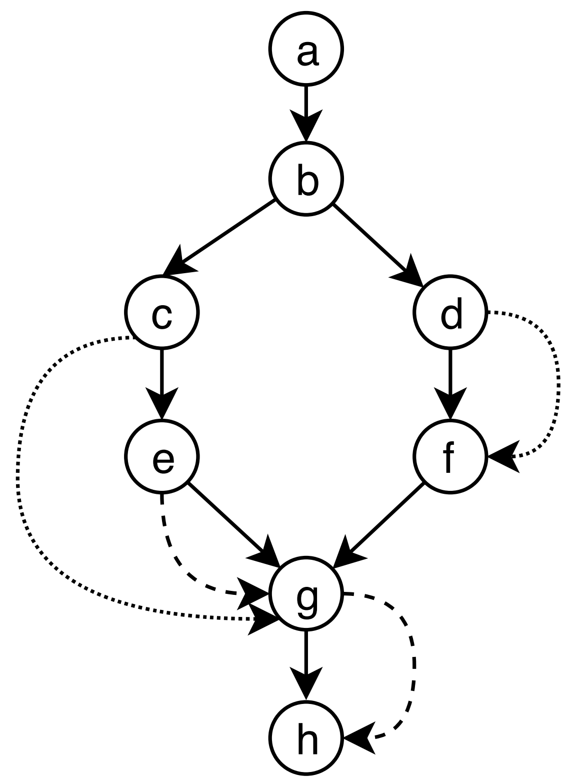


Diamond Decomposition: Absorbing and Emitting Paths

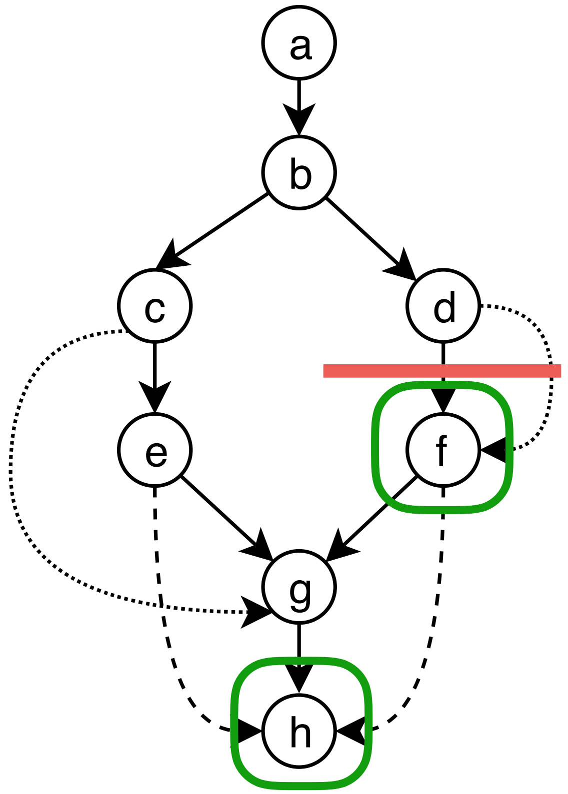


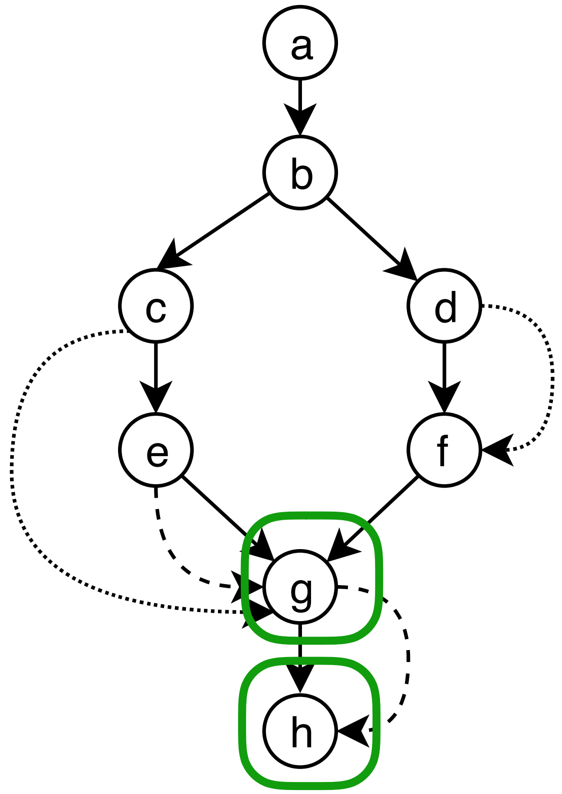
Diamond Decomposition: Absorbing and Emitting Paths



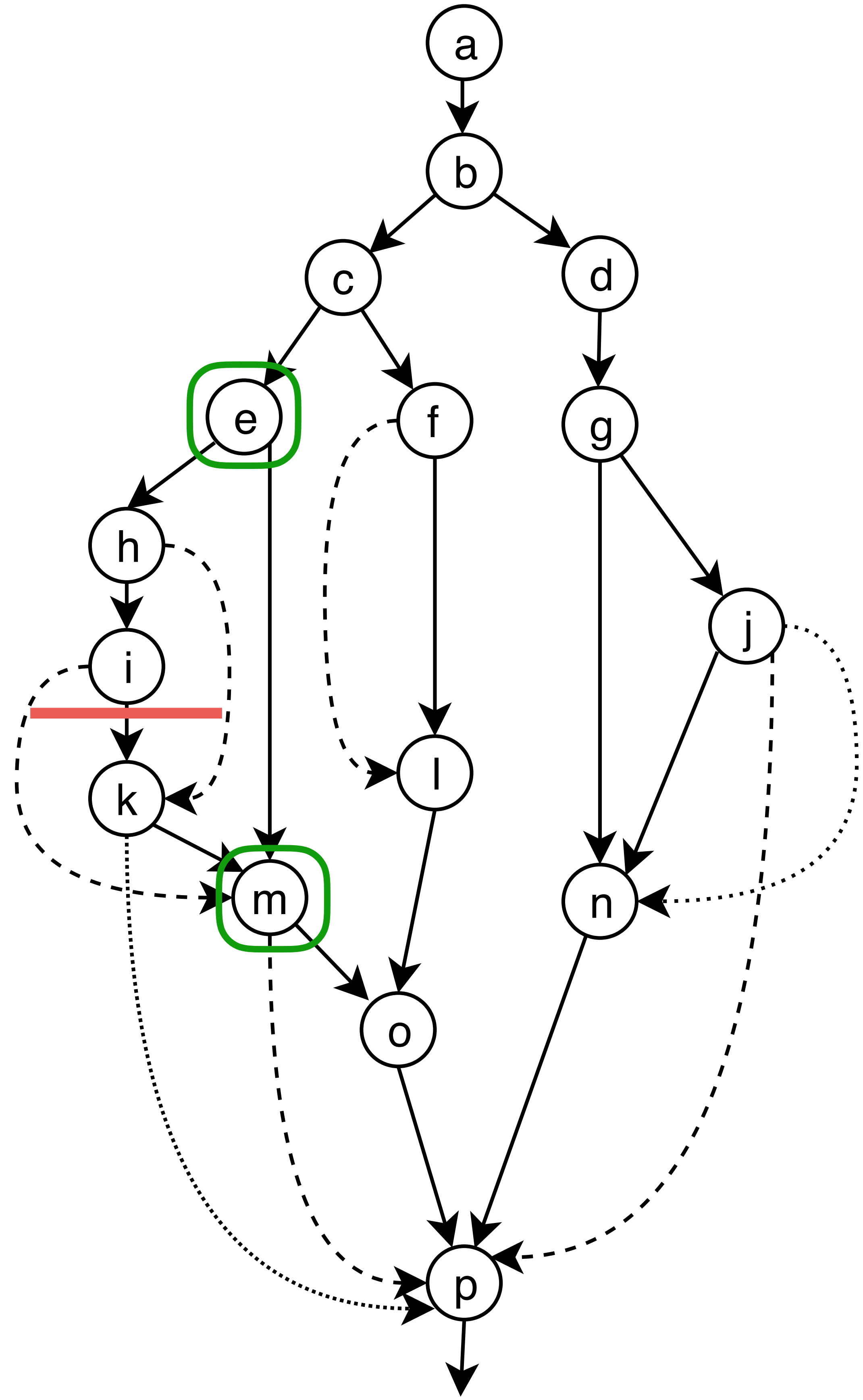


Diamond Decomposition: Absorbing and Emitting Paths

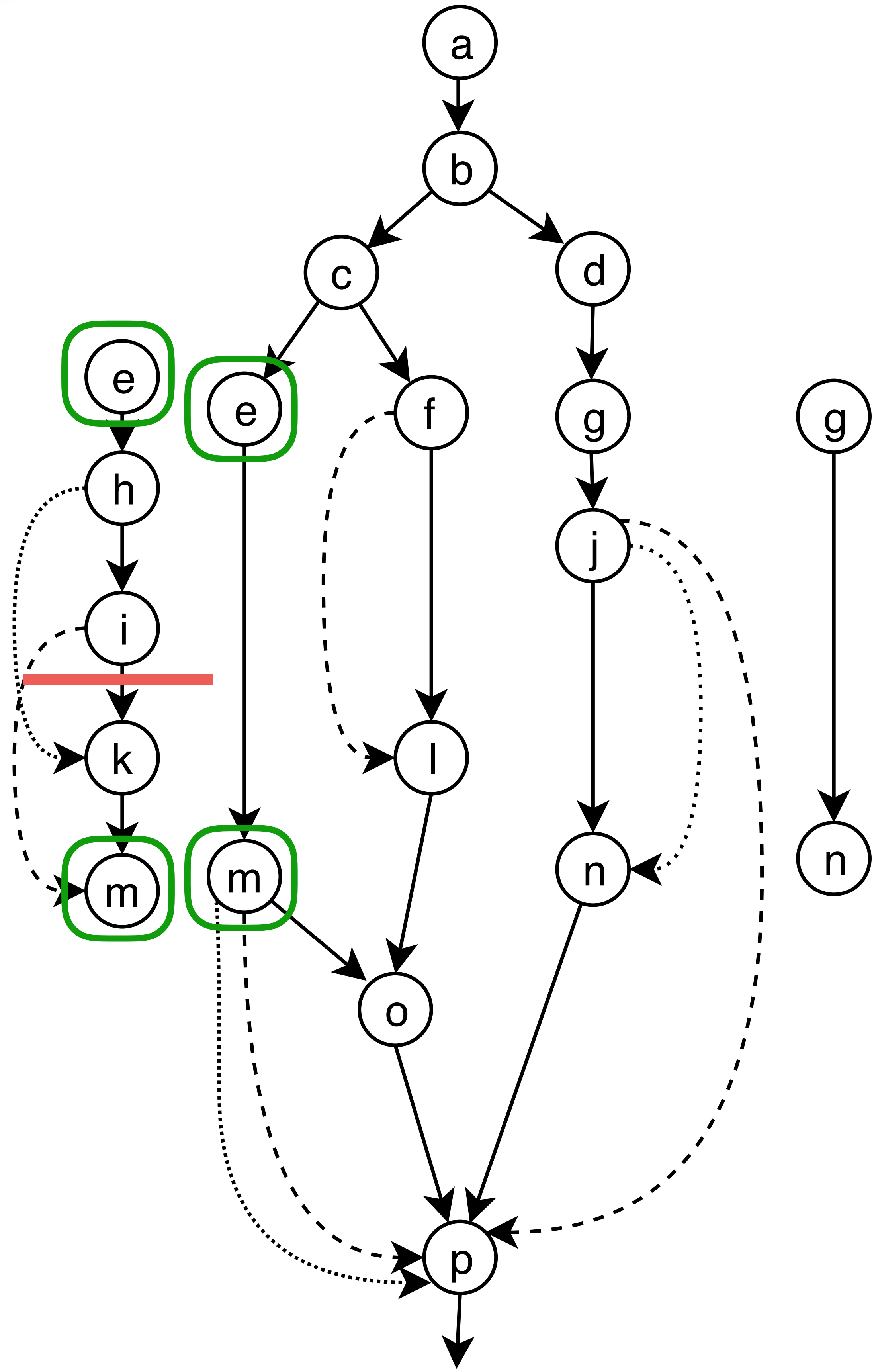




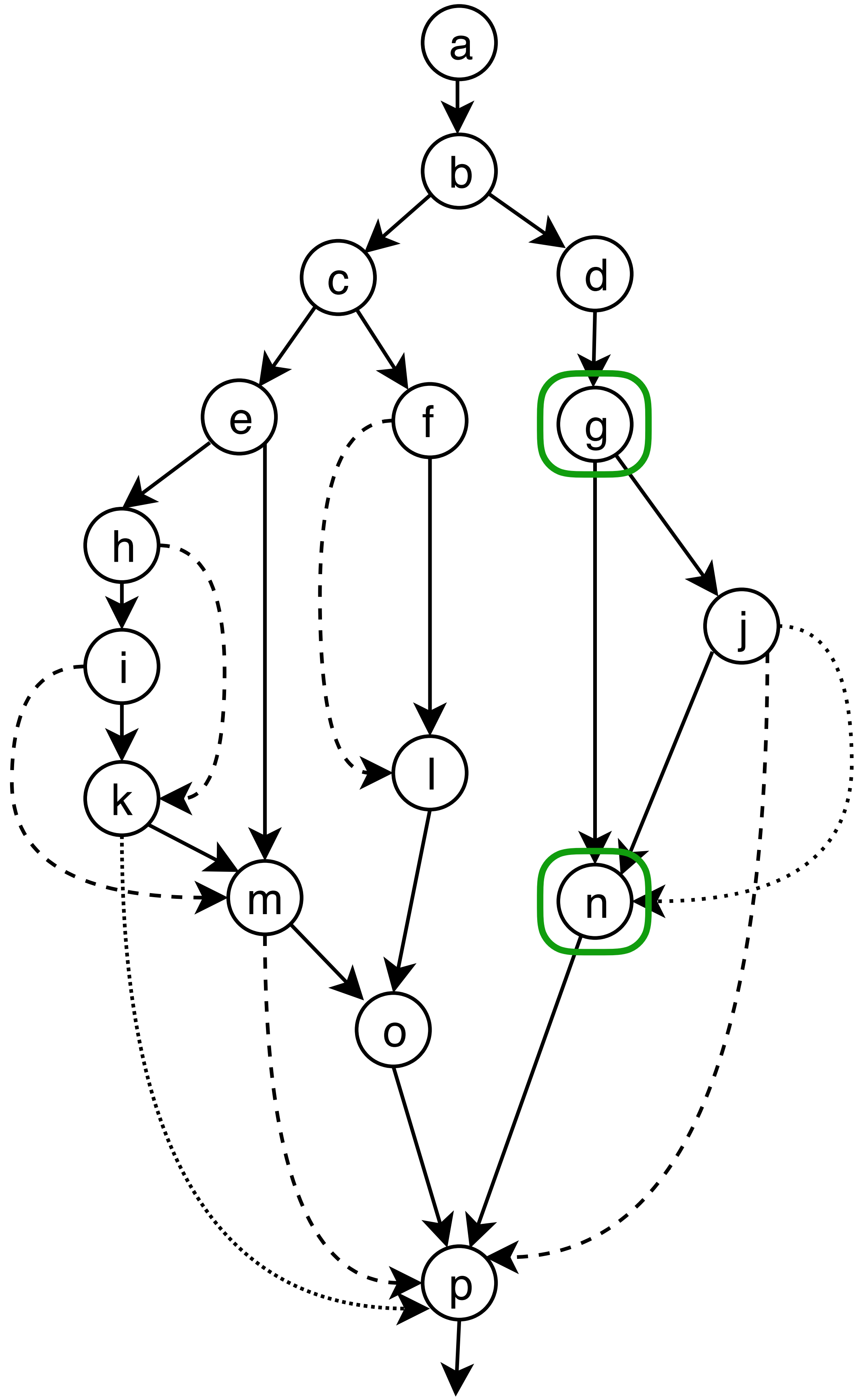
Diamond Decomposition, Level 0



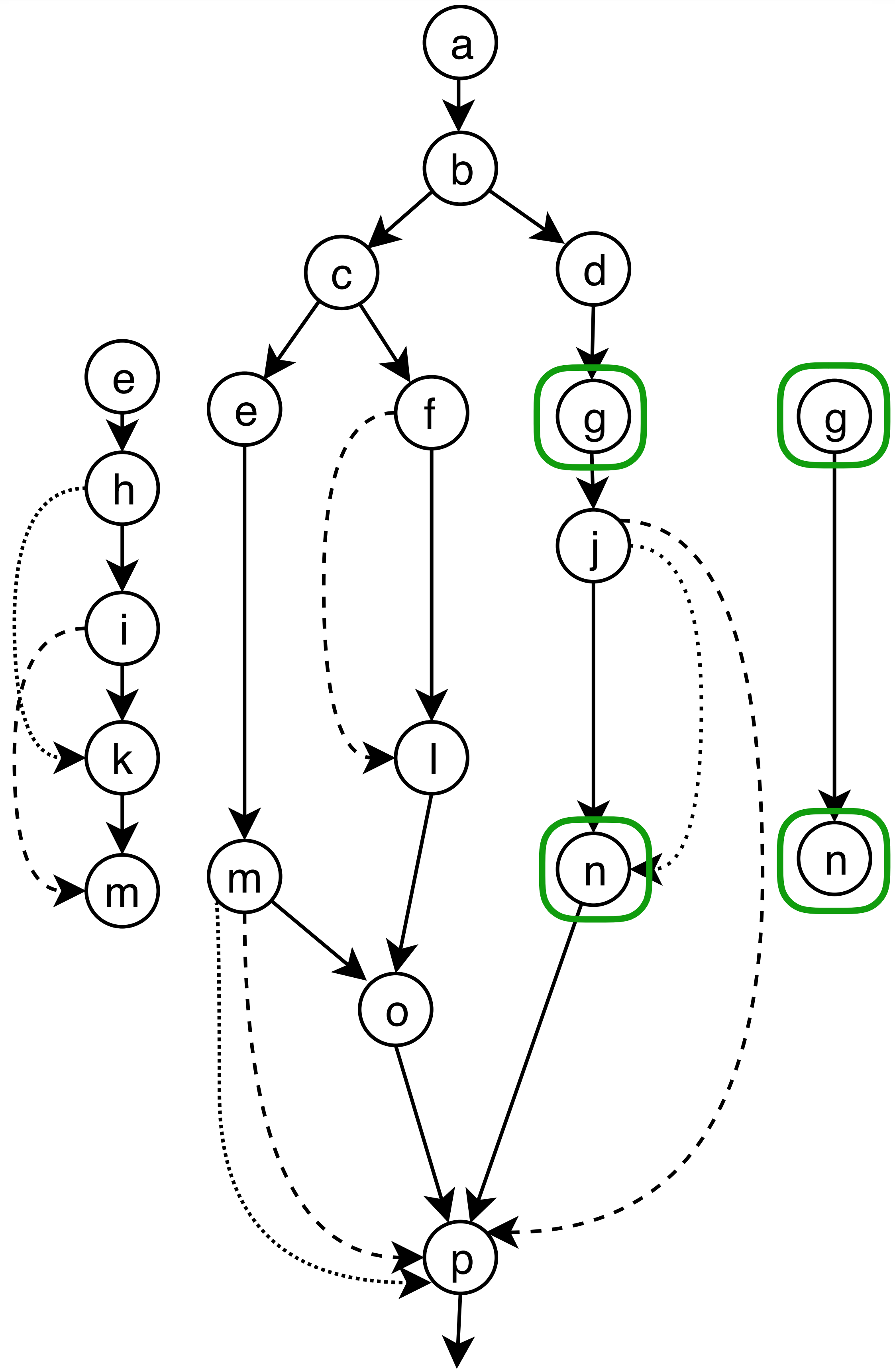
Diamond Decomposition, Level 0



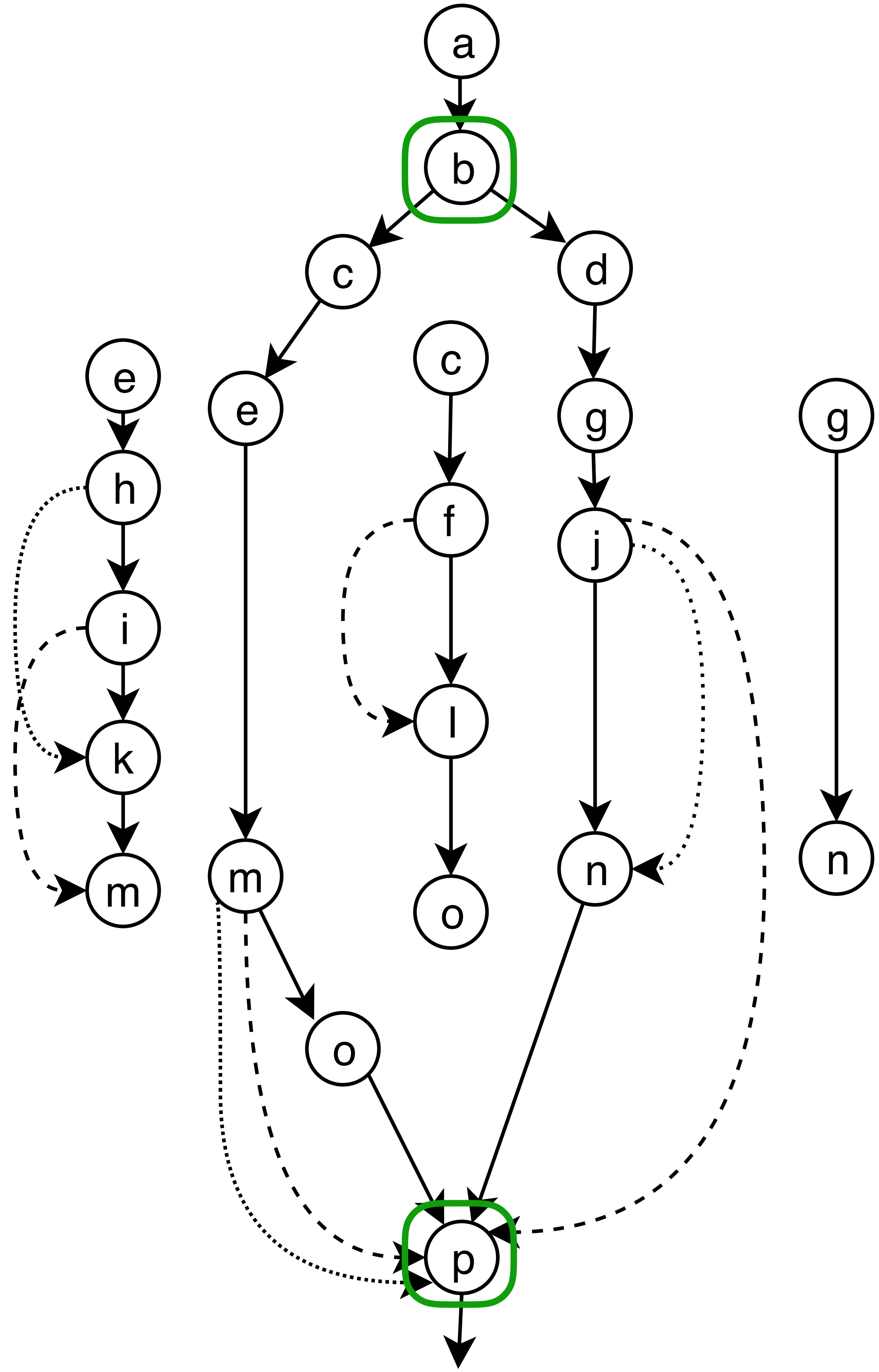
Diamond Decomposition, Level 0



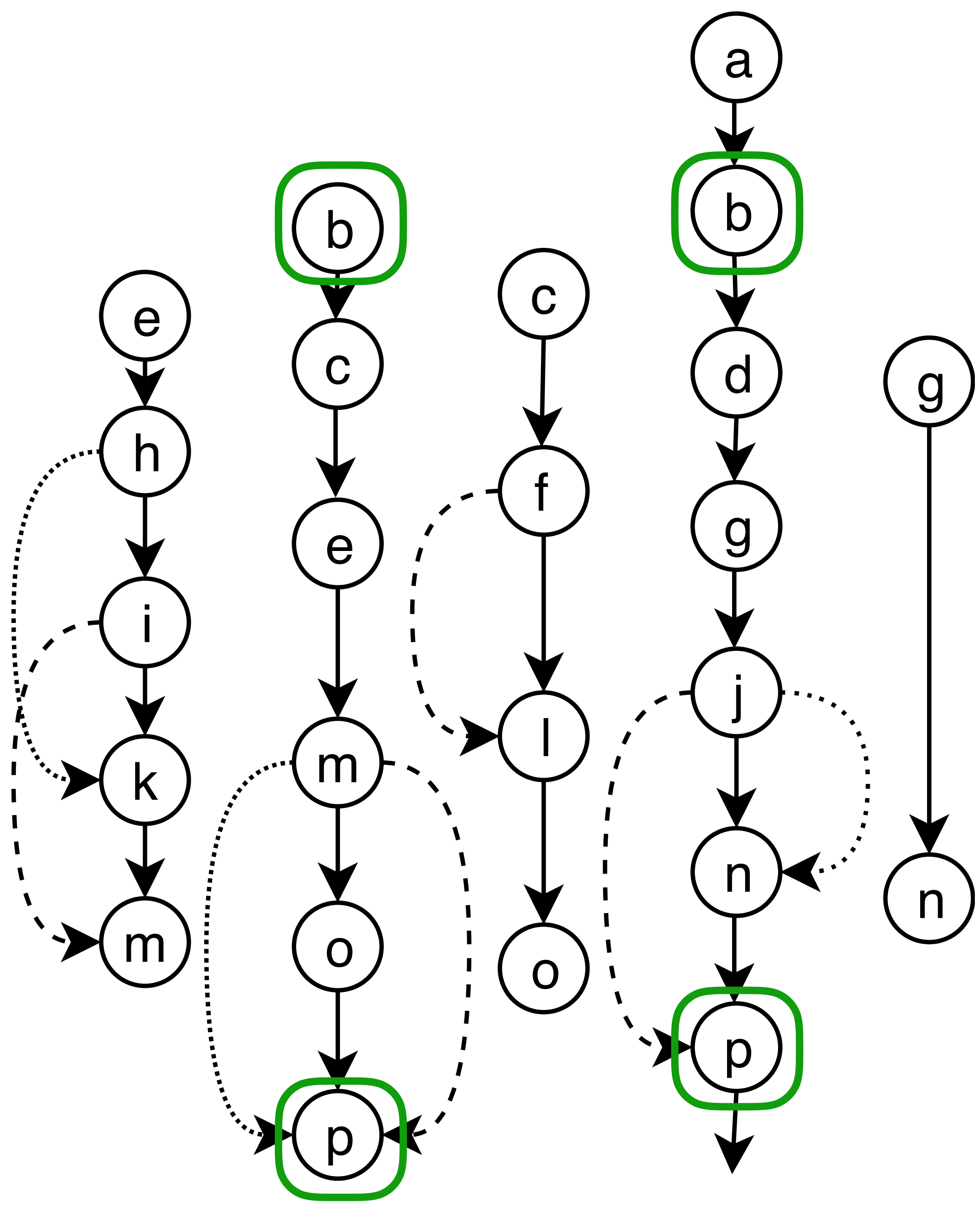
Diamond Decomposition, Level 0



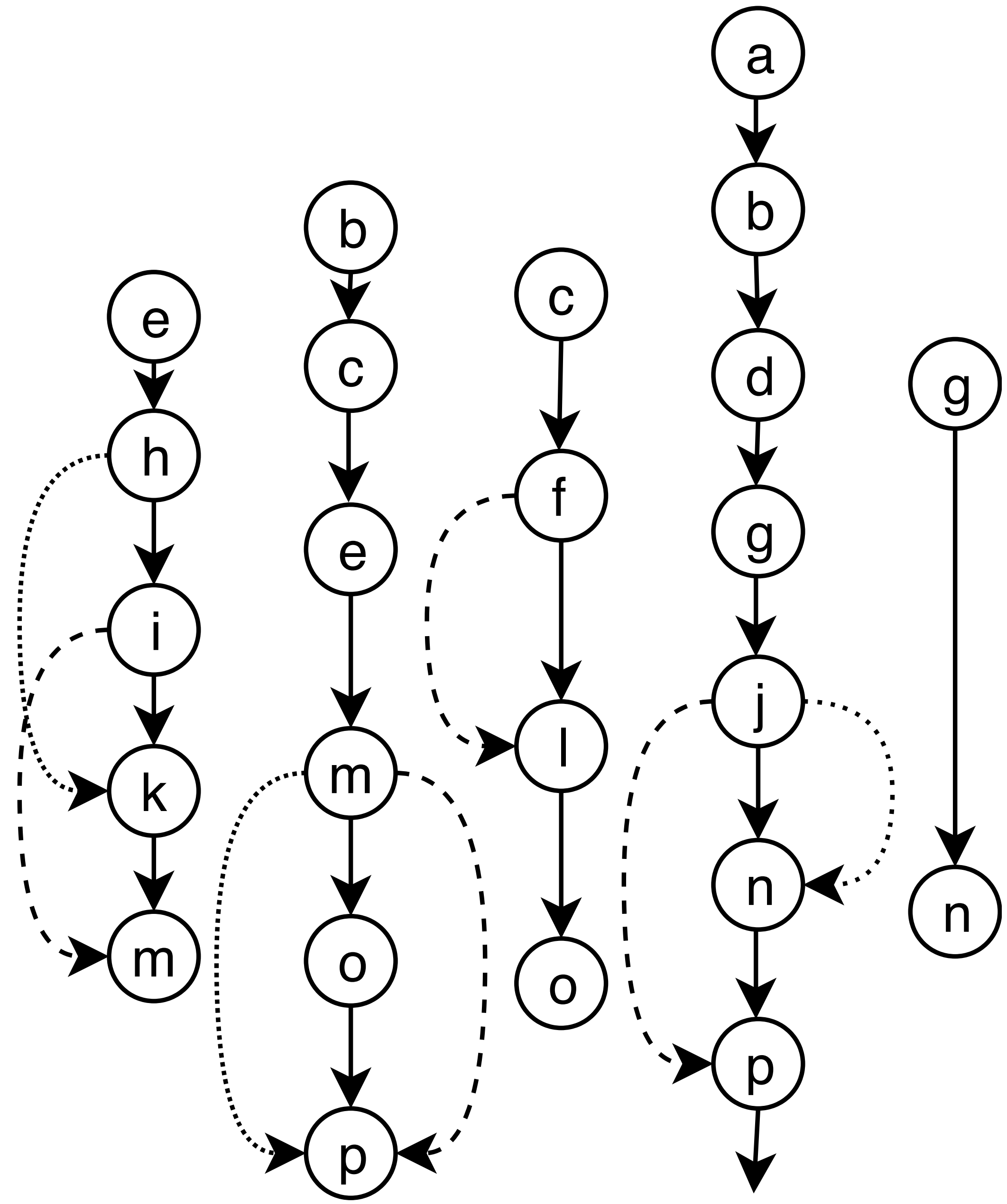
Diamond Decomposition, Level 1



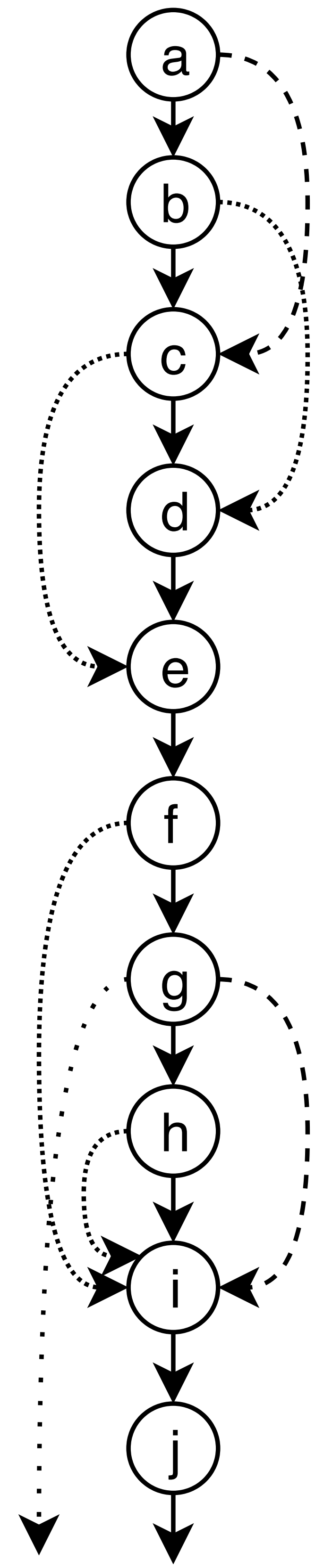
Diamond Decomposition, Level 2



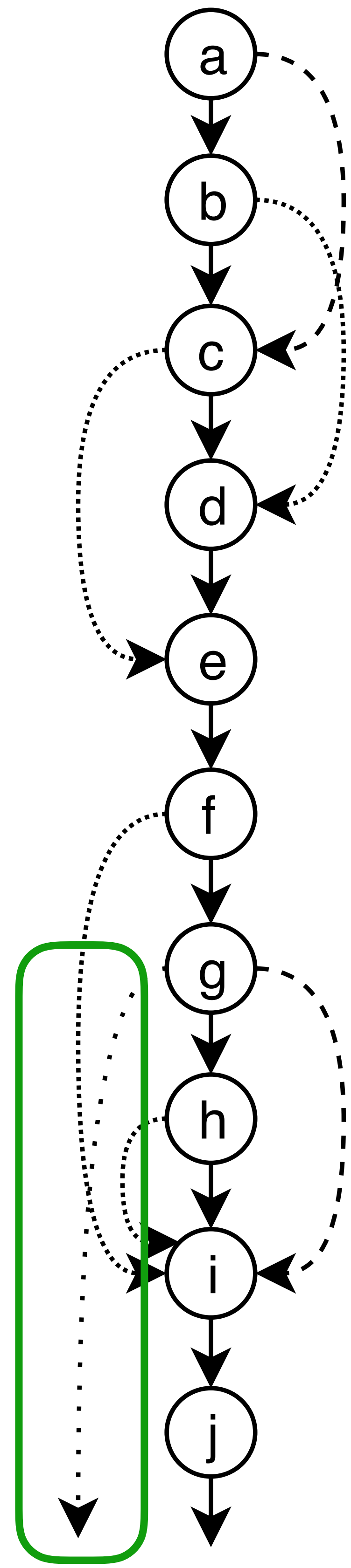
Fence Insertion for Simple Paths



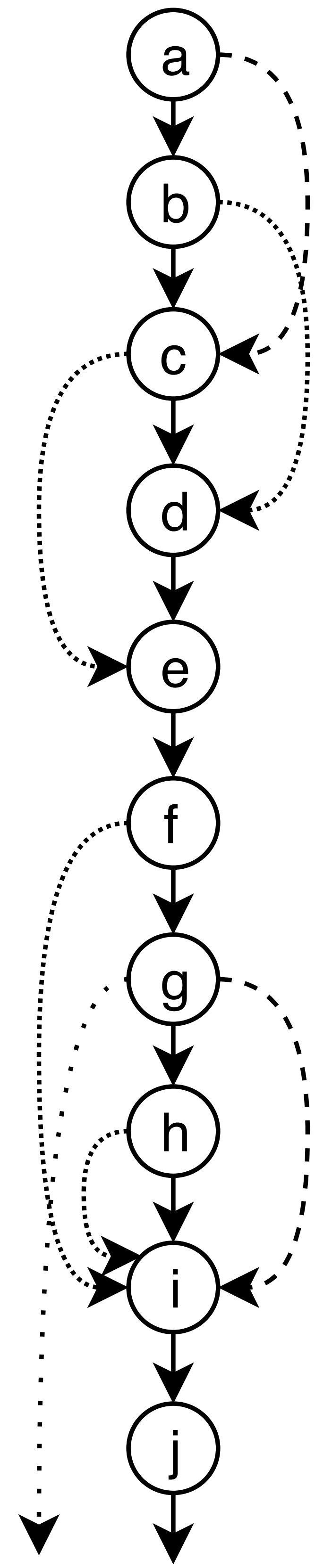
Fence Insertion for Simple Paths



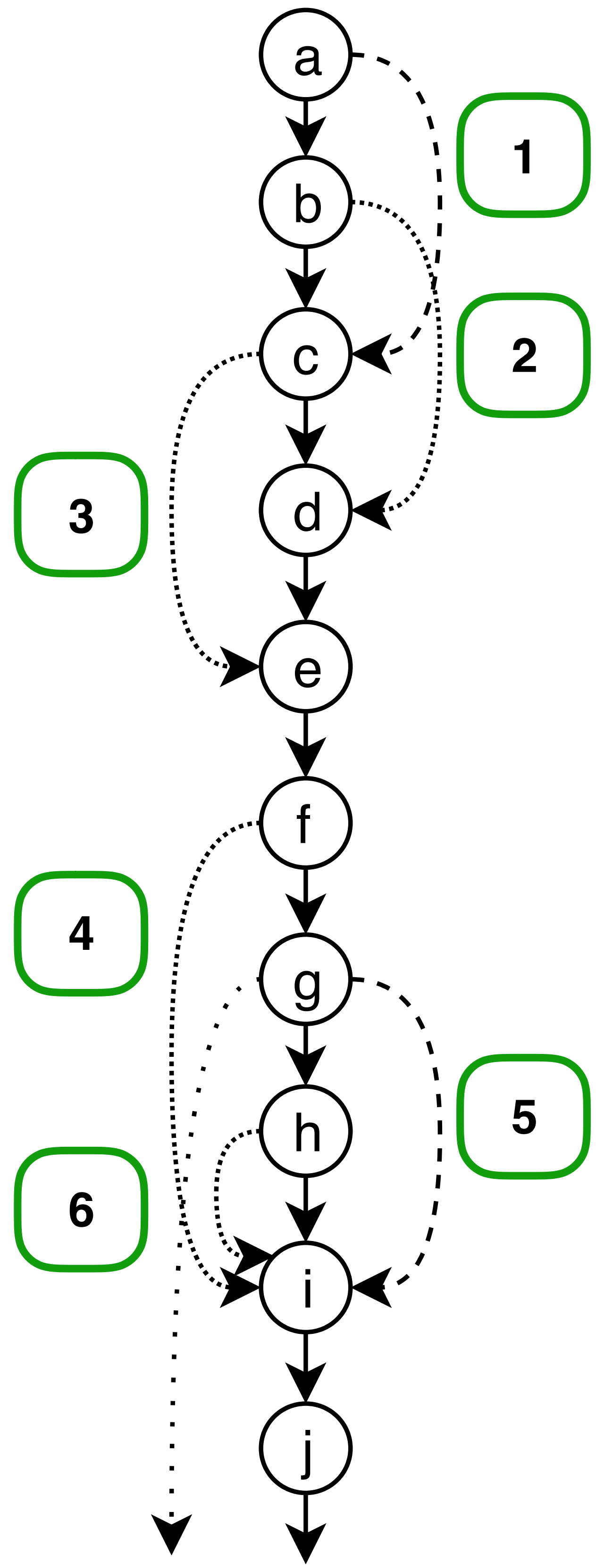
Fence Insertion for Simple Paths



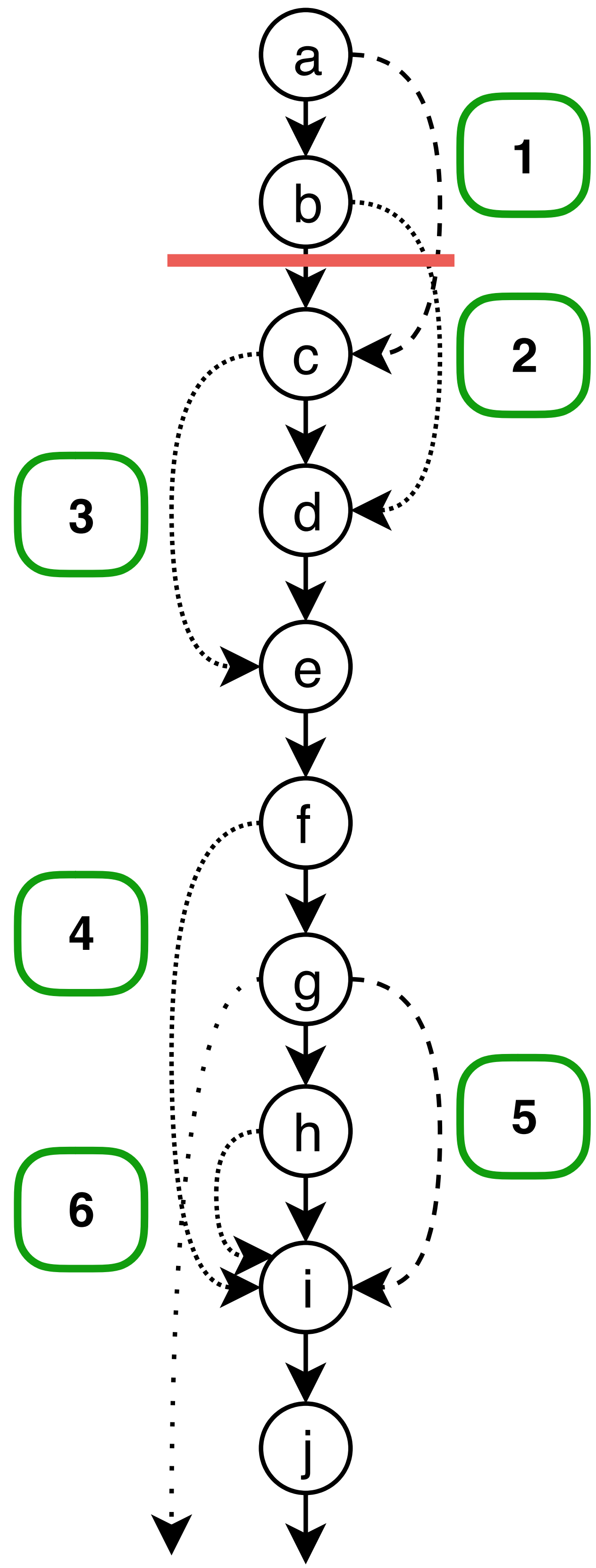
Fence Insertion for Simple Paths



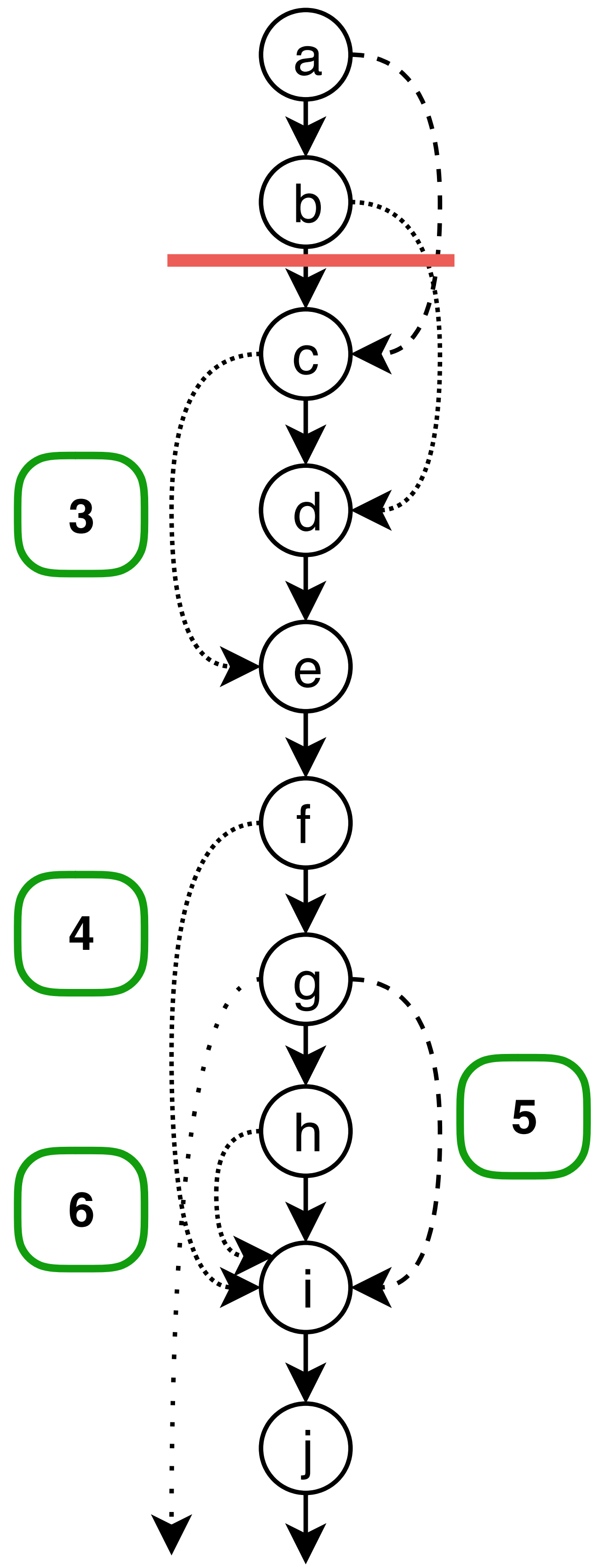
Fence Insertion for Simple Paths



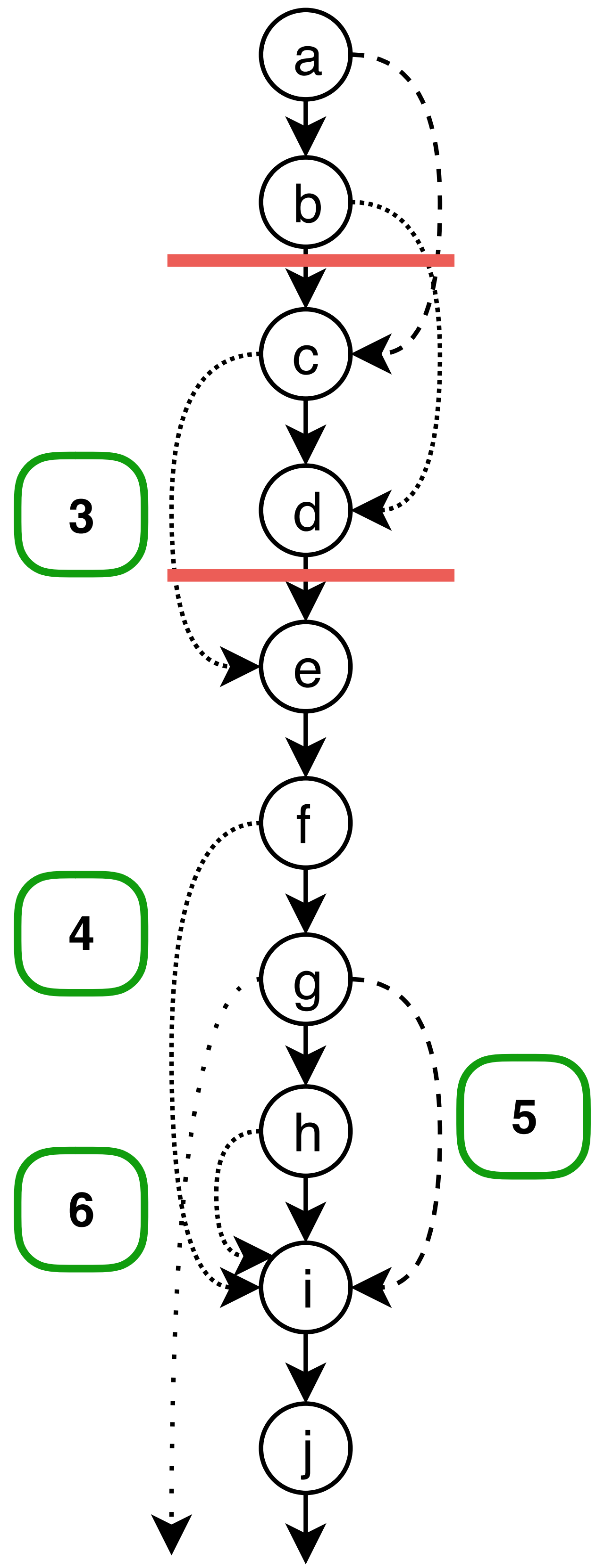
Fence Insertion for Simple Paths



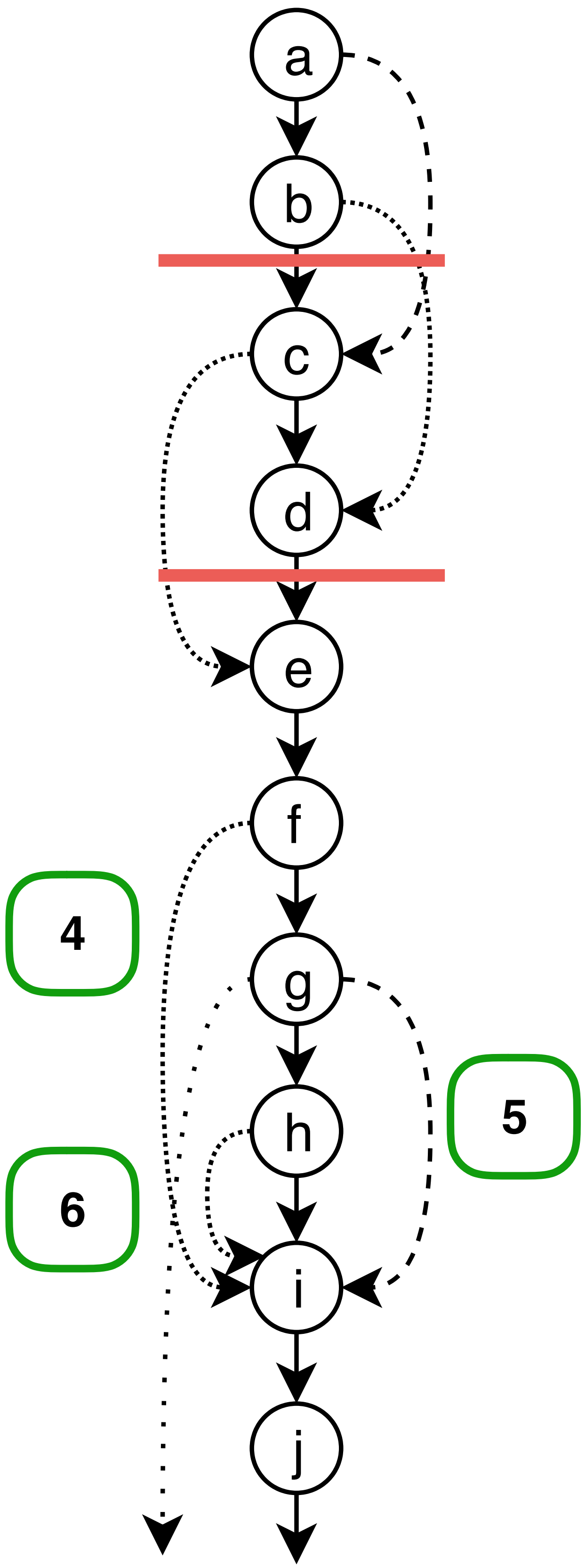
Fence Insertion for Simple Paths



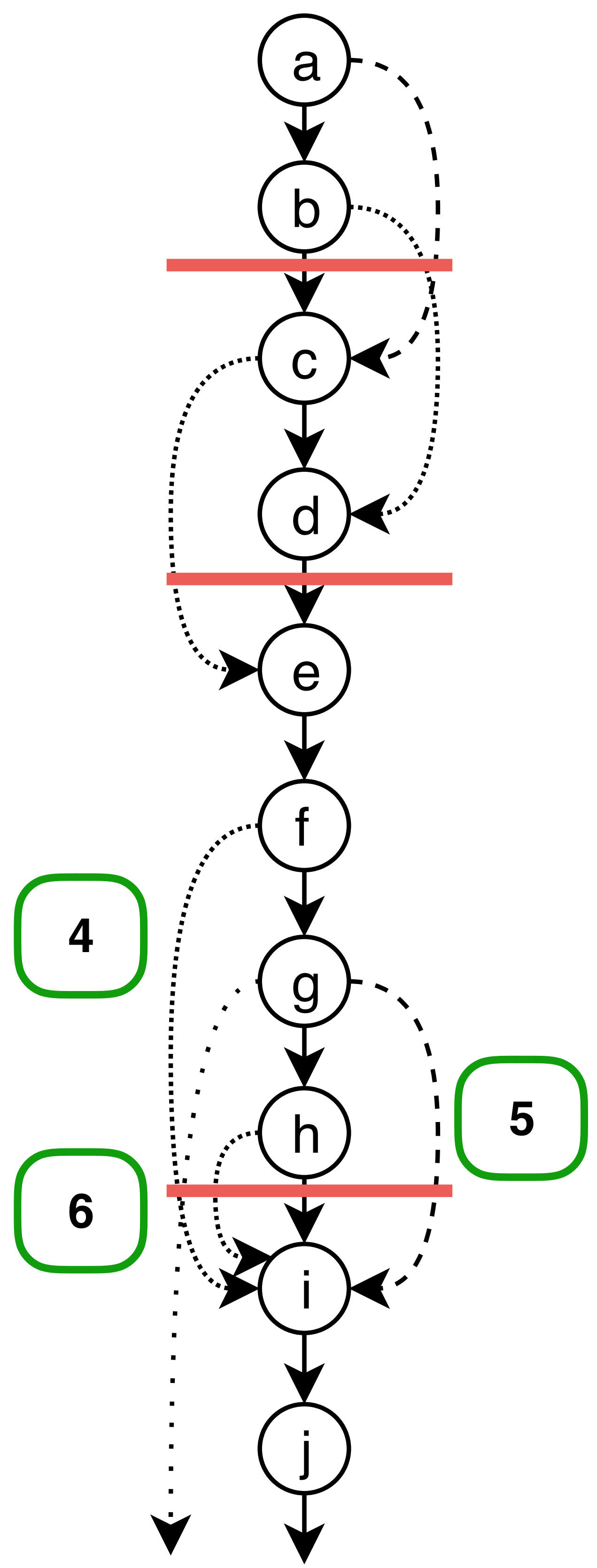
Fence Insertion for Simple Paths



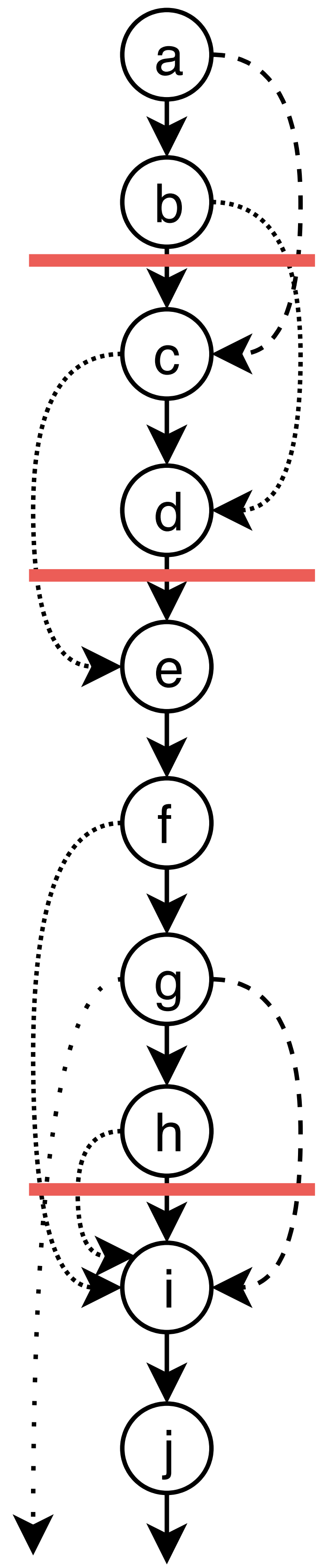
Fence Insertion for Simple Paths



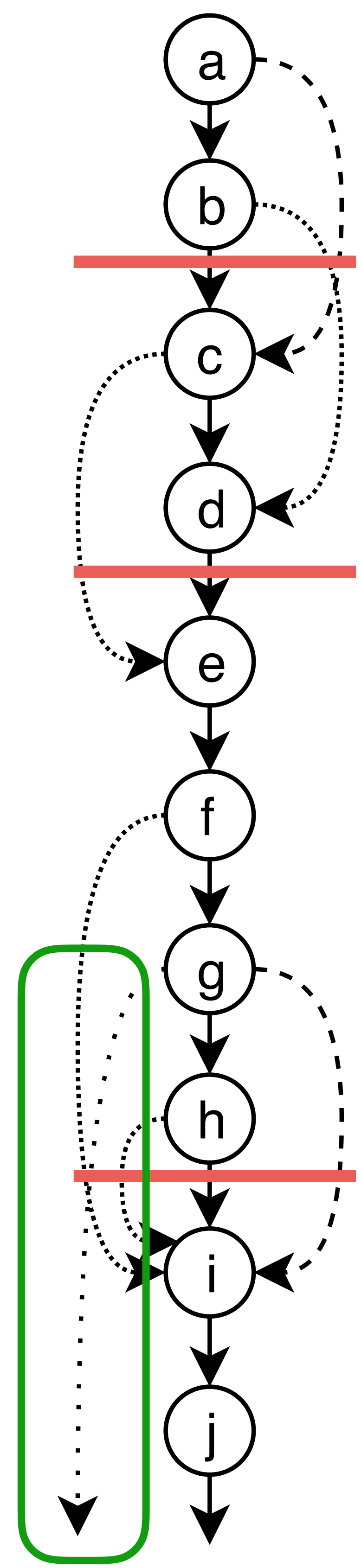
Fence Insertion for Simple Paths



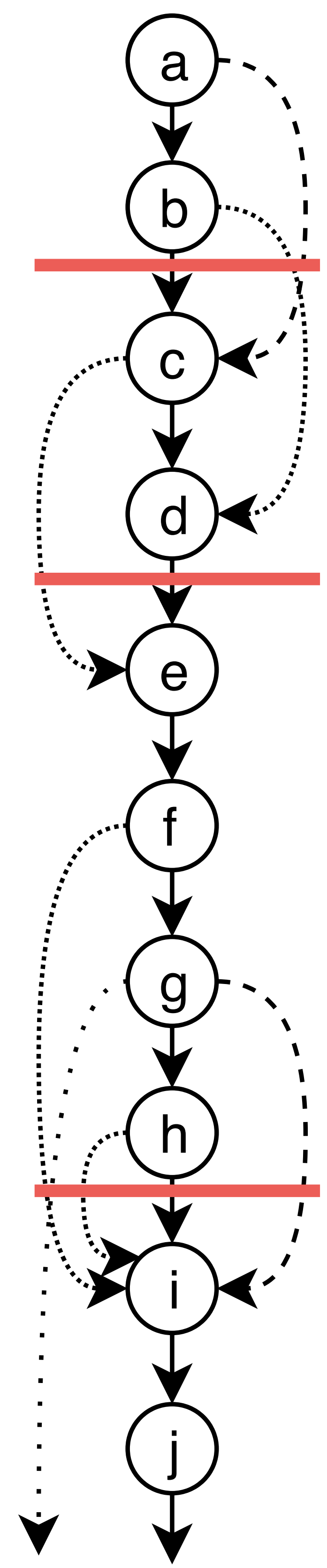
Fence Insertion for Simple Paths



Fence Insertion for Simple Paths



Fence Insertion for Simple Paths

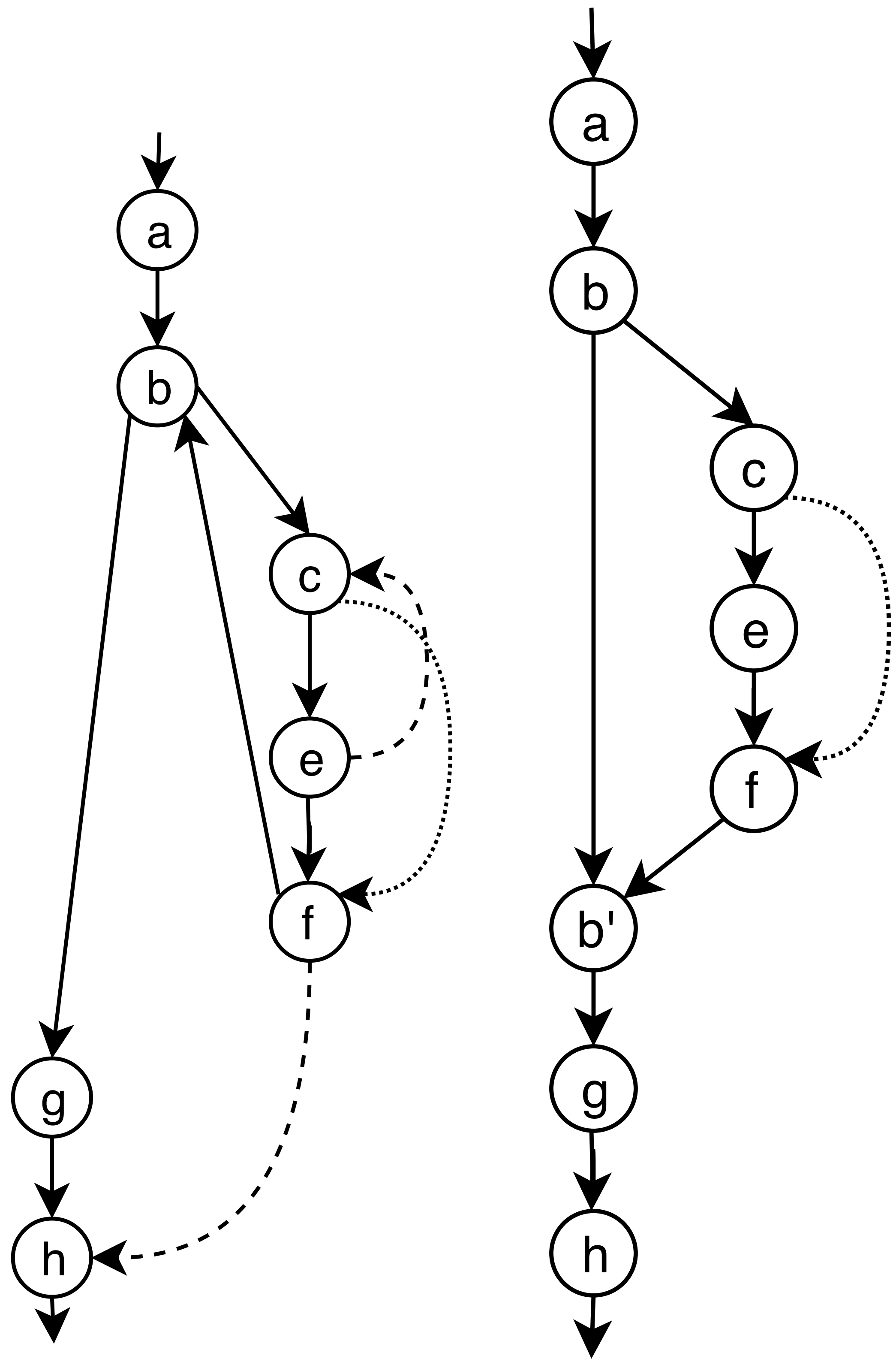


Optimality and Complexity

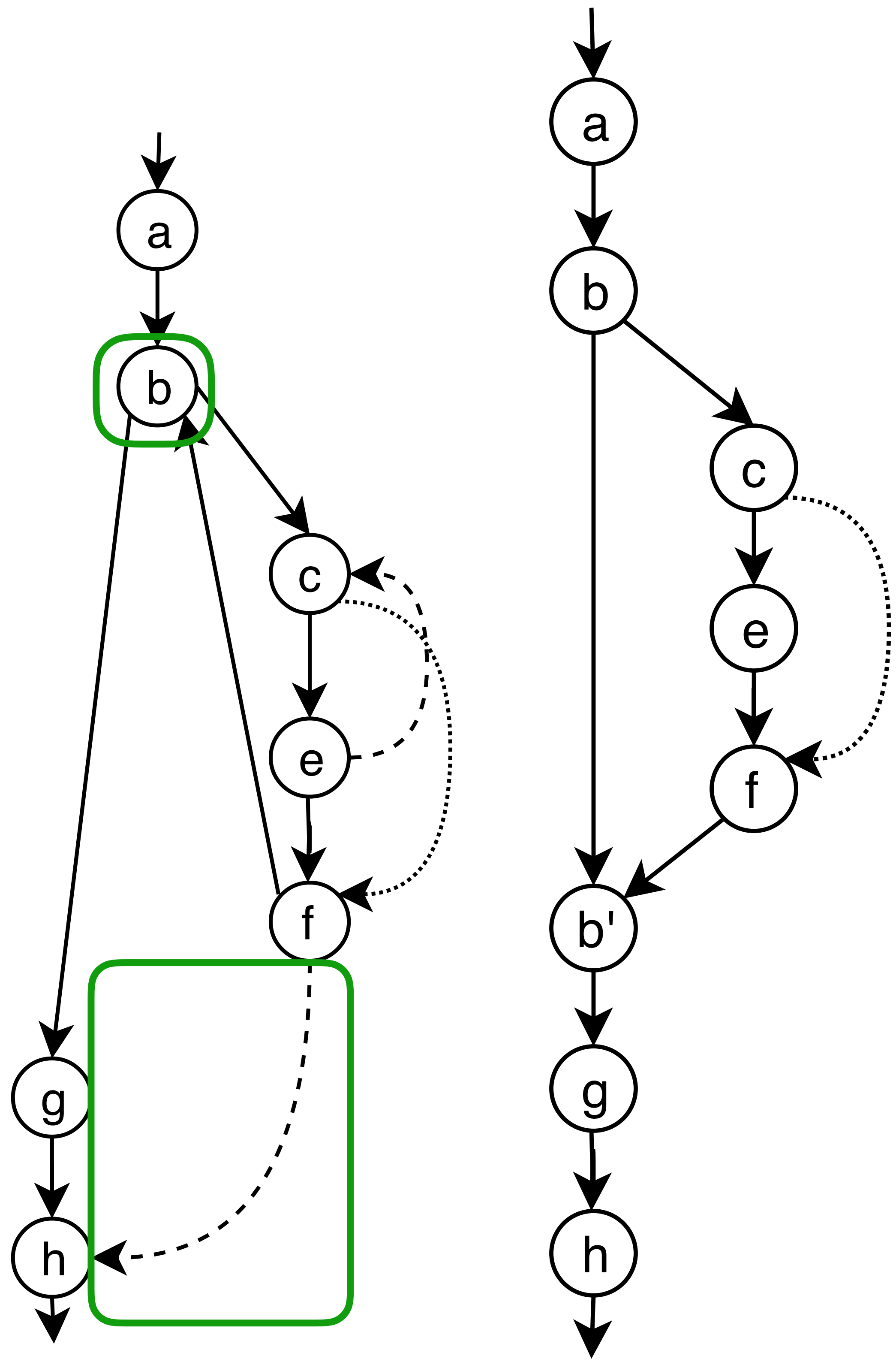
1. Decomposing Diamonds
    1. Absorbing path: The spanning constraint can be covered with no extra fence in the path.
    2. Emitting: If the extra fence is put outside the path, it may cover other overlapping constraints.
2. Simple Paths
    1. The size of the optimum solution is at least the size of every set of non-overlapping constraints.
    2. The constraints that lead to addition of fences are non-overlapping.

$$\mathcal{O}(|C|log|C| + |C||V| + |V|log|V|) \ time \ and \ \mathcal{O}(|C| + |V|) \ space \ complexity.$$

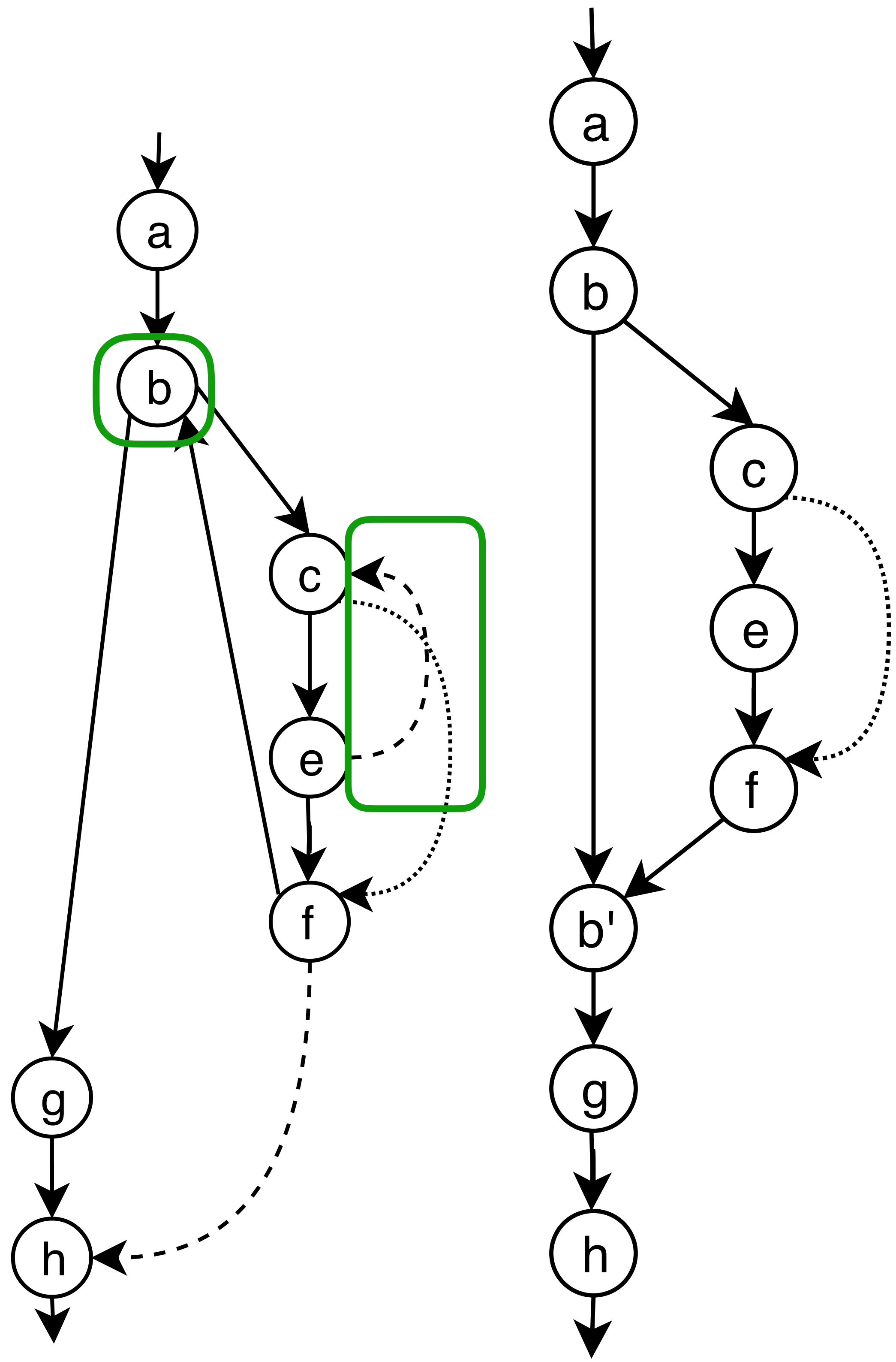
Converting a loop to a diamond



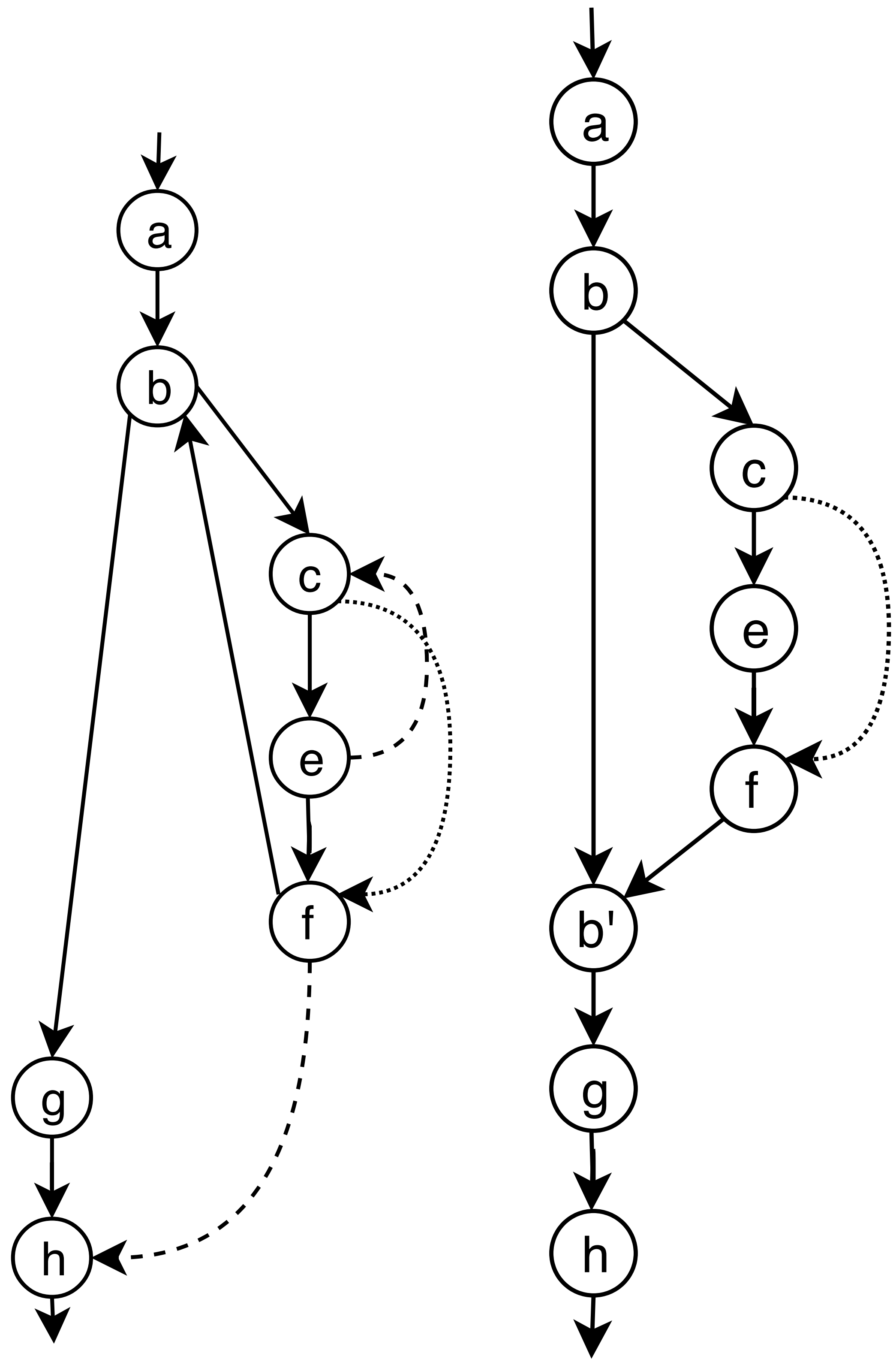
Converting a loop to a diamond



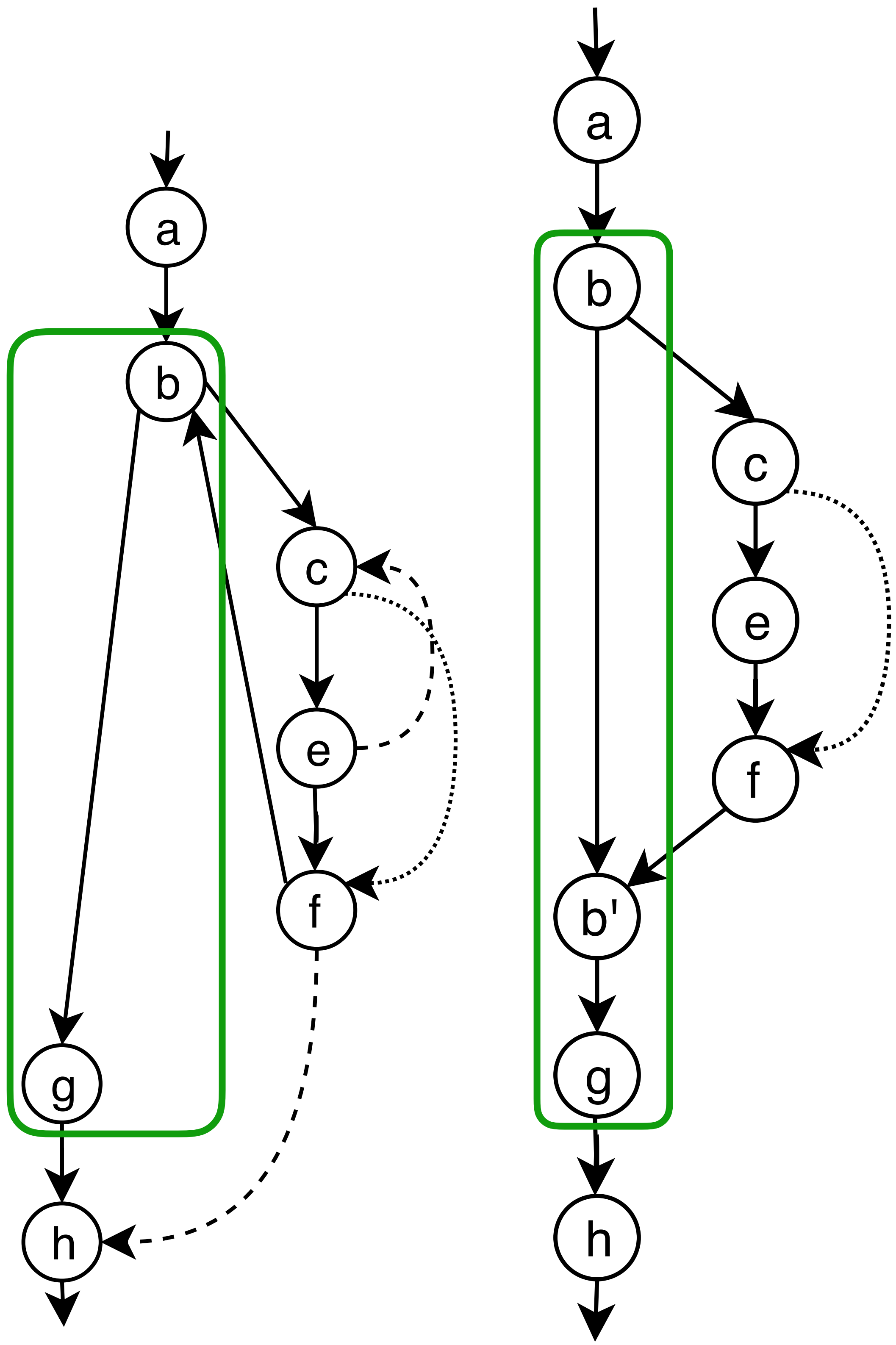
Converting a loop to a diamond



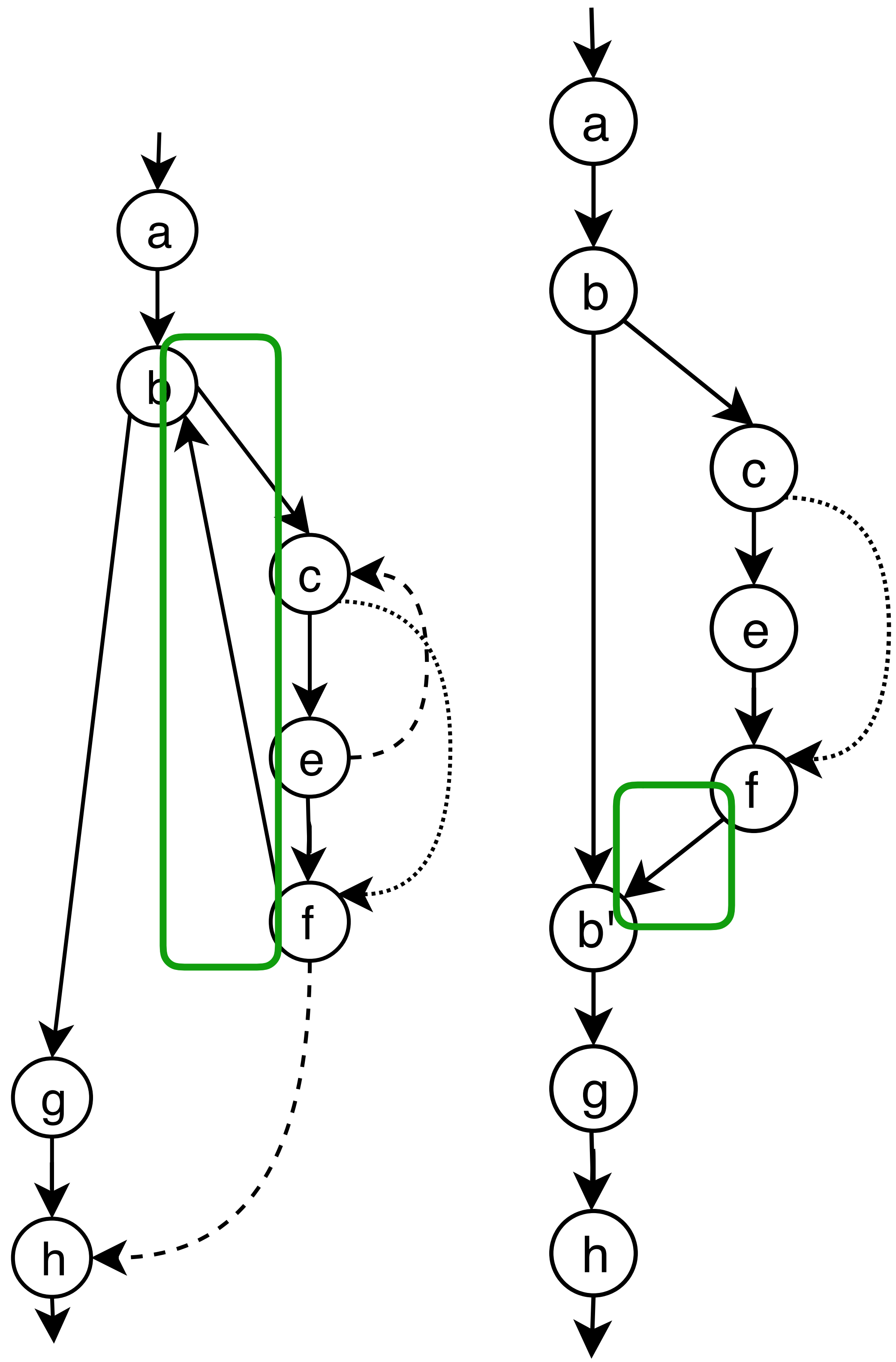
Converting a loop to a diamond



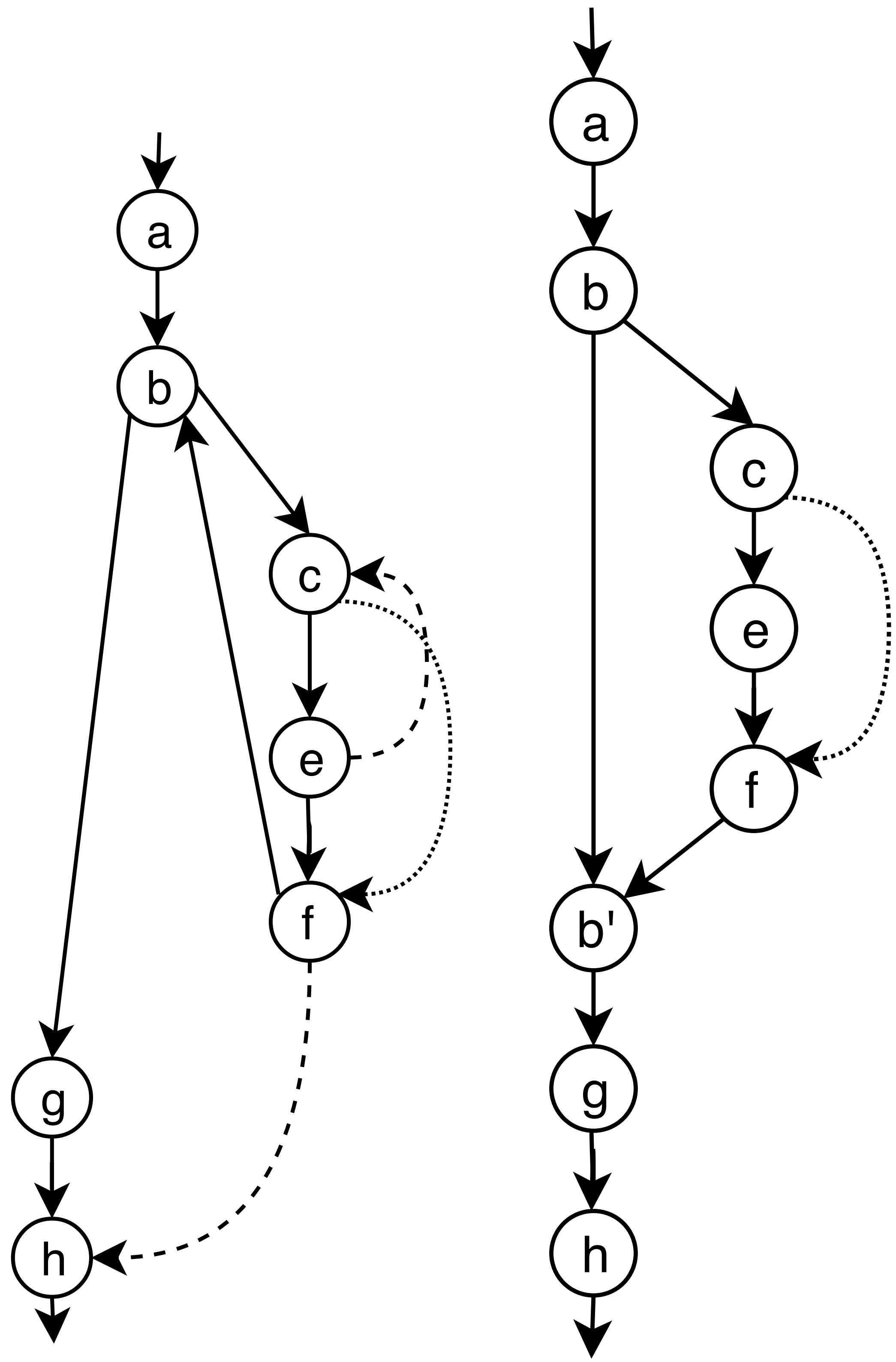
Converting a loop to a diamond



Converting a loop to a diamond



Converting a loop to a diamond



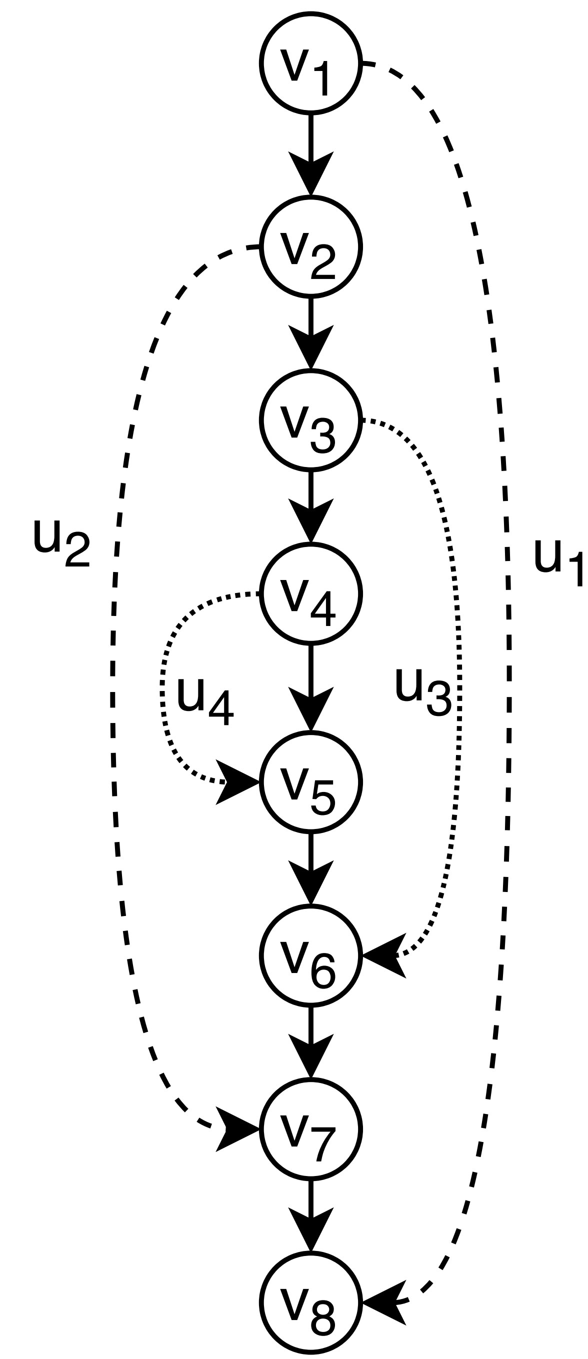
Multi-type Fence Insertion Problem is NP-hard

$\langle CT, FT, G, C \rangle$

$U = \{u_1,\, u_2,\, \ldots,\, u_n\}$

$S = \{S_1, S_2, S_3, \ldots, S_k\}$

$CT = U$
$FT = S$



Polynomial-time Fence Insertion For Structured Programs

M. Taheri, A. Pourdamghani, M. Lesani
University of California, Riverside
Sharif University of Technology