```
Sparse Simple Constant Propagation
WorkList ← $\emptyset$
for each SSA register r:
{
    initialize Value(r) by rules discussed
    if Value(r) $\neq \top$
        WorkList ← WorkList $\cup \{r\}$
}

while WorkList $\neq \emptyset$
{
    remove some $r$ from WorkList
    for each operation $op$ that uses $r$
    {
        let $m$ be the name defined by $op$
        if Value(m) $\neq \bot$
        {
            t ← Value(m)
            Value(m) ← evaluate(op)
            if Value(m) $\neq$ t
                WorkList ← WorkList $\cup \{m\}$
        }
    }
}
rewriteUses(Value)
```

Sparse Conditional Constant Propagation

```
sccp(cfg):
    # initialization
    FlowWorkList $\leftarrow \emptyset$
    SsaWorkList $\leftarrow \emptyset$
    for each edge in cfg:
        edge.executable $\leftarrow$ false
    for each SSA register r:
        Value(r) $\leftarrow \top$

    # fill lists based on entry block
    executeBlock(cfg.entry, ...)

    # process worklists
    while FlowWorkList $\neq \emptyset$ or SsaWorkList $\neq \emptyset$:
    {
        item $\leftarrow$ select from either FlowWorkList or SsaWorkList

        # flow worklist item
        if item is *edge* and **not** edge.executable:
            edge.executable $\leftarrow$ true
            if first visit to edge.destination:
                visit-block(edge.destination) # may add edges based on insts
            else:
                visit-$\phi$($\phi$)

        # ssa worklist item
        if item is ssa register: # I actually just store the destination instructions
            for each use in item.uses:
                if use is $\phi$-function:
                    visit-$\phi$(use)
                else if block(use) contains executable edge:
                    visit-expression(use)
    }
    rewriteUses(Value)
    fixBlocksBranches()
```