1)
Backward validation is better when you have fewer queries than facts as backwards validation will result in a number of trees <= to the number of queries.
Forwards validation will result in a number of tree >= to the number of known facts.

2)

```
getPrems(KB, Q) returns list of lists of premises //(e.g. a list of clauses)
        prems = new list;
        for clause in KB
                if(c.conclusion == Q)
                        prems.add(C.premises);
        return prems;

backwardsEntails(KB, Q, his)
        if(agenda.contains(Q))
                return true;
        prems = getPrems(KB, Q);
        for pl in prems
                entails = false;
                for p in pl
                        if(his.contains(p)) return false; //to avoid loops
                        his.add(p);
                        if(!backwardsEntails(KB, p, his))
                                entails = false;
                                break;
                        else
                                entails = true;
                                his.remove(p);
                if(entails)
                        agenda.add(Q);
                        return true;
        return false;

pl-bc-Entails(KB, Q) =
        his = new list;
        return backwardsEntails(KB, Q, his);
```

3)
Since A requires C and is in a cyclical dependence with G, KB does not entail A. The algorithm will stop the cyclical recursion by finding C in the history when trying to evaluate G, and return that C is false.

4)
If we assume that getPrems(), agenda.contains() & his.contains() could be converted to have constant complexity (e.g. via hashset), this would run in linear time since each premise is at most evaluated as being true or false once.