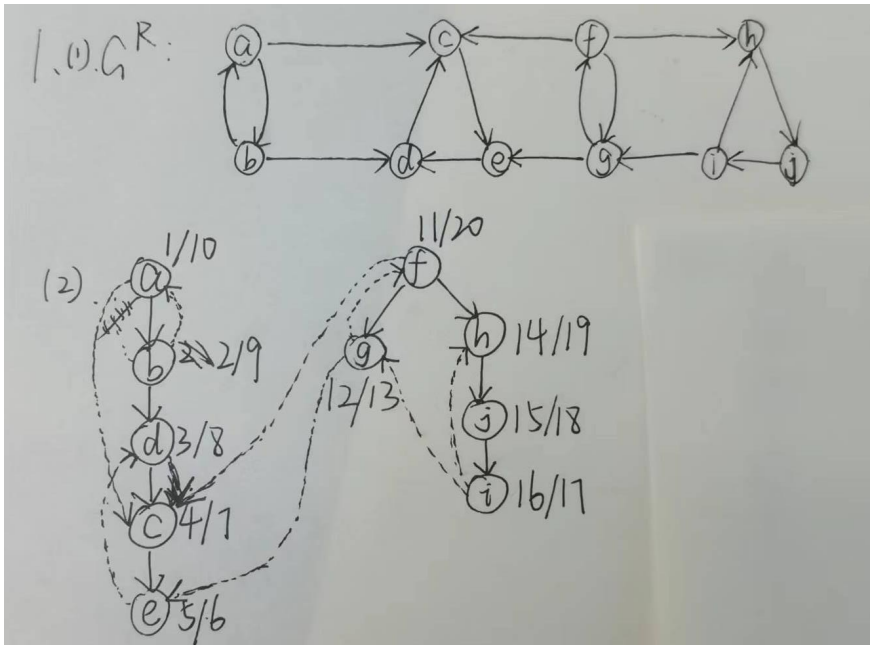
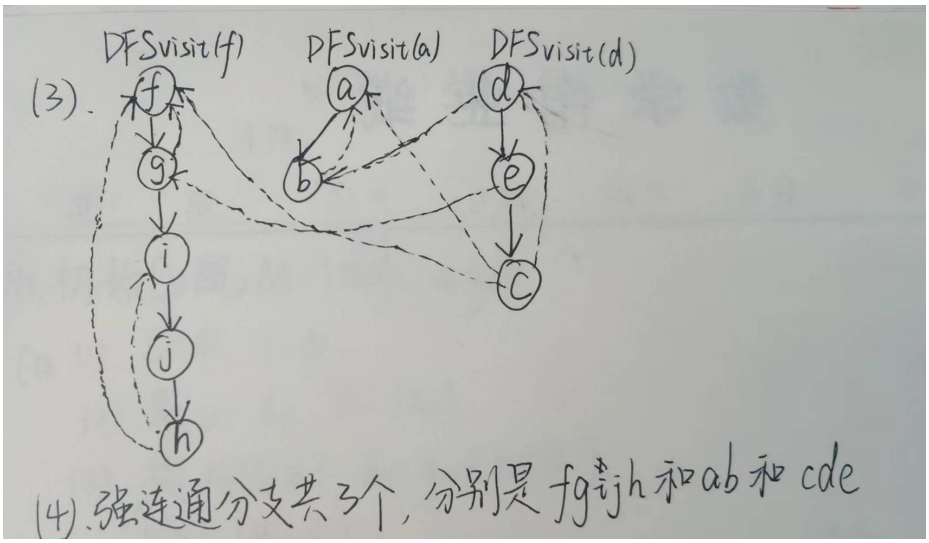
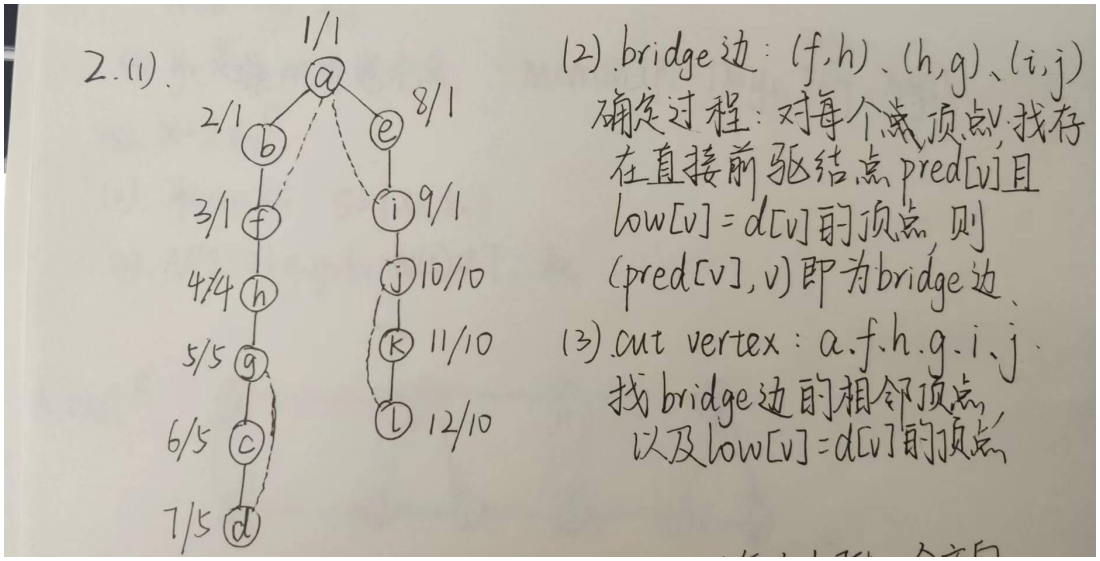
1.


1.(1). G^R:
(2).
1/10
2/9
3/8
4/7
5/6
11/20
12/13
14/19
15/18
16/17


(3).
DFSvisit(f)
DFSvisit(a)
DFSvisit(d)
(4). 强连通分支共3个，分别是fgijh和ab和cde

2.


2.(1).
1/1
2/1
3/1
4/4
5/5
6/5
7/5
8/1
9/1
10/10
11/10
12/10
(2). bridge边：(f,h)、(h,g)、(i,j)
确定过程：对每个点顶点v，找存在直接前驱结点pred[v]且low[v]=d[v]的顶点，则(pred[v],v)即为bridge边。
(3). cut vertex：a.f.h.g.i.j.
找bridge边的相邻顶点，以及low[v]=d[v]的顶点

3.算法思路：如果无向图 G 可以通过为每一个边附一个方向，使得所有点都不是 source 点，则图 G 必含有环（回路），且所有顶点要么在环（回路）上，要么与环（回路）相连通。即只需要图 G 的每个连通分支都包含环（回路）即可。
因此，在 DFS 检测是否有环的算法基础上，对每个连通分支进行判断，只要有一个连通分支无环，图 G 就不符合要求，就可以停止遍历。只有当所有连通分支遍历完，均包含环时，才能判定图 G 满足要求

```
DFS(G) {
    time = 0
    for each (u in V)
        mark[u] = undiscovered
    for each (u in V) {
        int flag = 0;//标记是否有环，有则为 1；无则为 0
        if (mark[u] == undiscovered)
            DFSVisit(u, &flag)
        if( ! flag){   //只要任意一个连通分支无环，则图 G 不满足要求
            printf("No\n");
            return;
        }
    }
    printf("Yes\n");//遍历完所有连通分支，均有环，则图 G 满足要求
}
DFSvisit(u, int * flag) {
    mark[u] = discovered
    d[u] = ++time
    for each (v in Adj(u)) {
        if (mark[v] == undiscovered) {
            pred[v] = u
            DFSvisit(v, flag)
        } else if (mark[v] != finished && v != pred[u]) {
             *flag = 1;   //有环，将 flag 置为 1
        }
    }
    mark[u] = finished
    f[u] = ++time
}
```