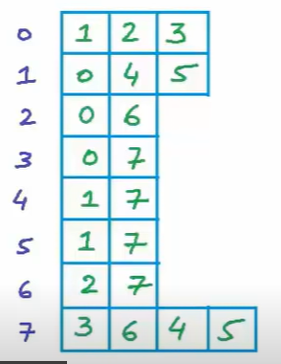
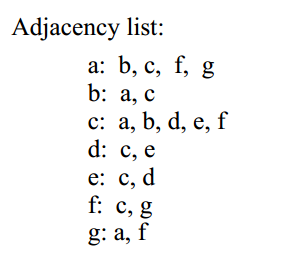
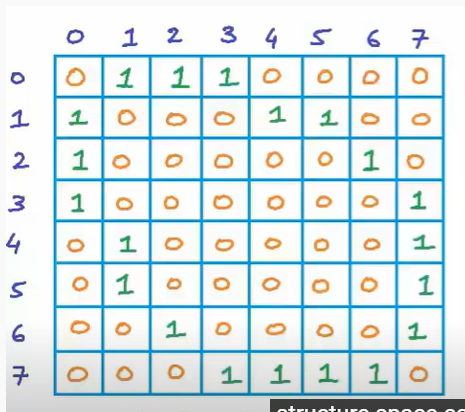
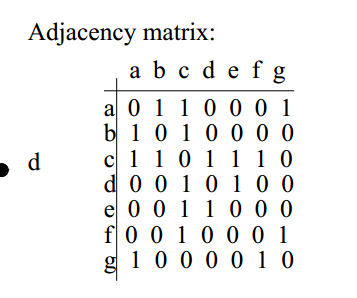
Graph representations

两种表达方式.list,matrix

a. adjacency list representation



相邻点存在一个list中

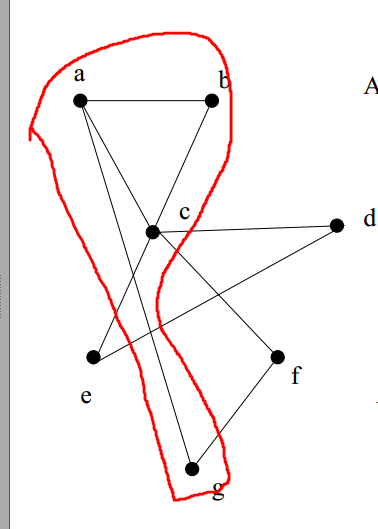


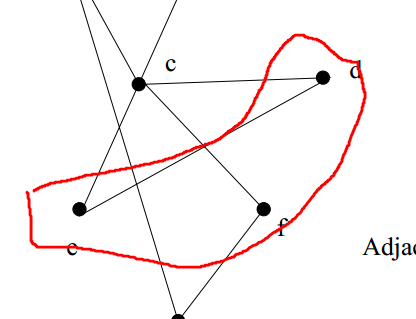
创造一个n\*n的2dArray

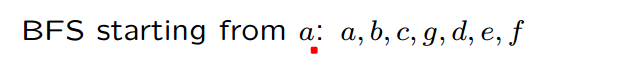
BFS, explosion形式

一层一层往外扩

以A为中心点

到a距离为1的一圈

到a距离为2的一圈



BFS是由Queue 构建的

a先找一圈，bcg

这时bcg插入queue中

queue秉持先进先出原则，

查看b，b周围没有空点，pop

查看c，c周围有d,e,f，Push进去

够了

BF会构建一个graph的tree representation,叫做BF tree

一个vertex的发现时间取决于他到原点的距离

DFS

Depth-first,

全部点都设置成unexplored

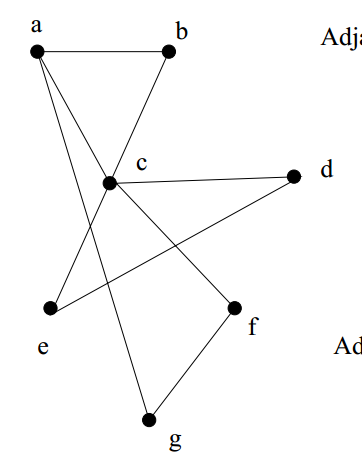
我们可以想象成有一个stack

a点开始，放入stack, a设置成explored，随意选择下一个unexplored.并放入stack,设置成explored...

直到遇到一个点，他周围没有unexplored，那么把这个点pop出stack,（回退到前一个点）

如果有unexplored,继续我们的算法，如果没有，再pop

直到遇到有Unexplored点

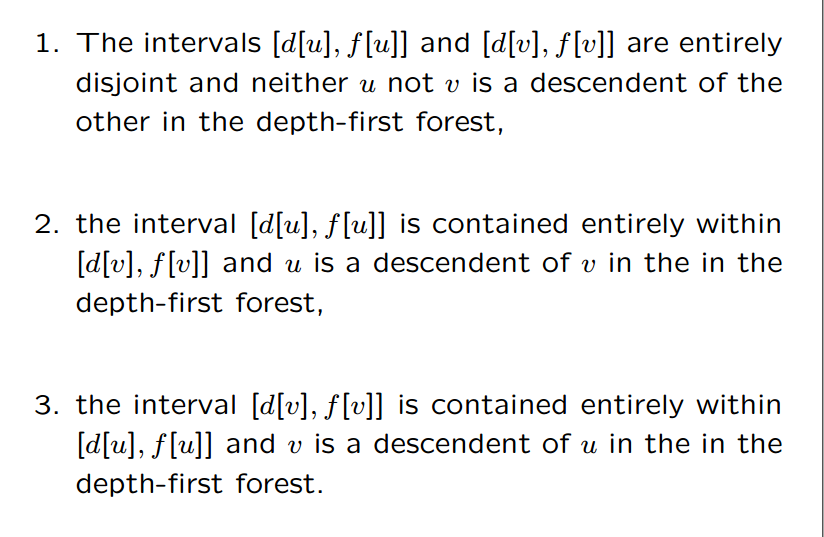




abcde都很正常，e这时cd都是explored,pop e，stack的顶为d,还是不行,pop d，stack点为c，f,g OK

与BFS相同，每个vertex所用时与他被visit的顺序有关

对于DFS来说，任意两个点必然满足一下关系之一



Topological sort

拓扑排序

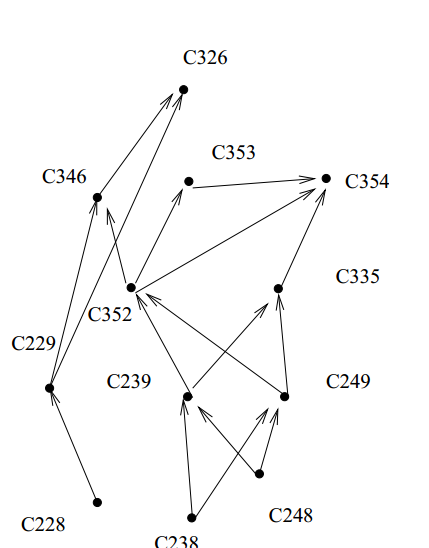
他只针对DAG directed acyclic graph有效，

他把所有点排序出一个order



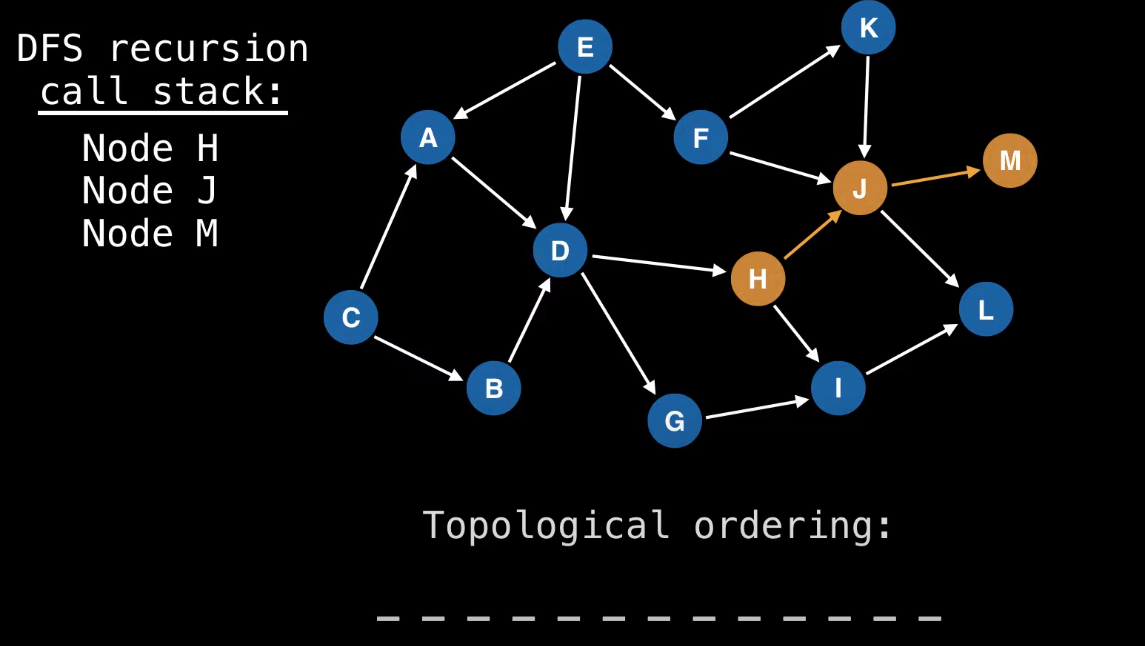
这样如果存在ui uj间存在edge，那么必然是i小于j

例如找课啥的，我们可以通过拓扑排序找到必修课

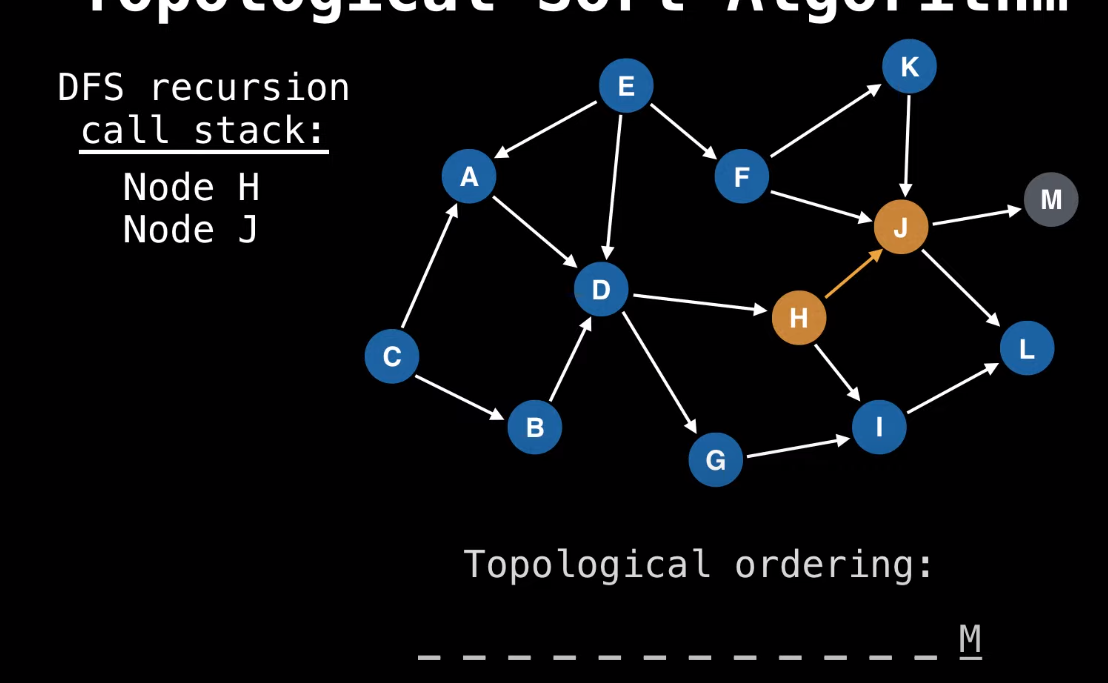


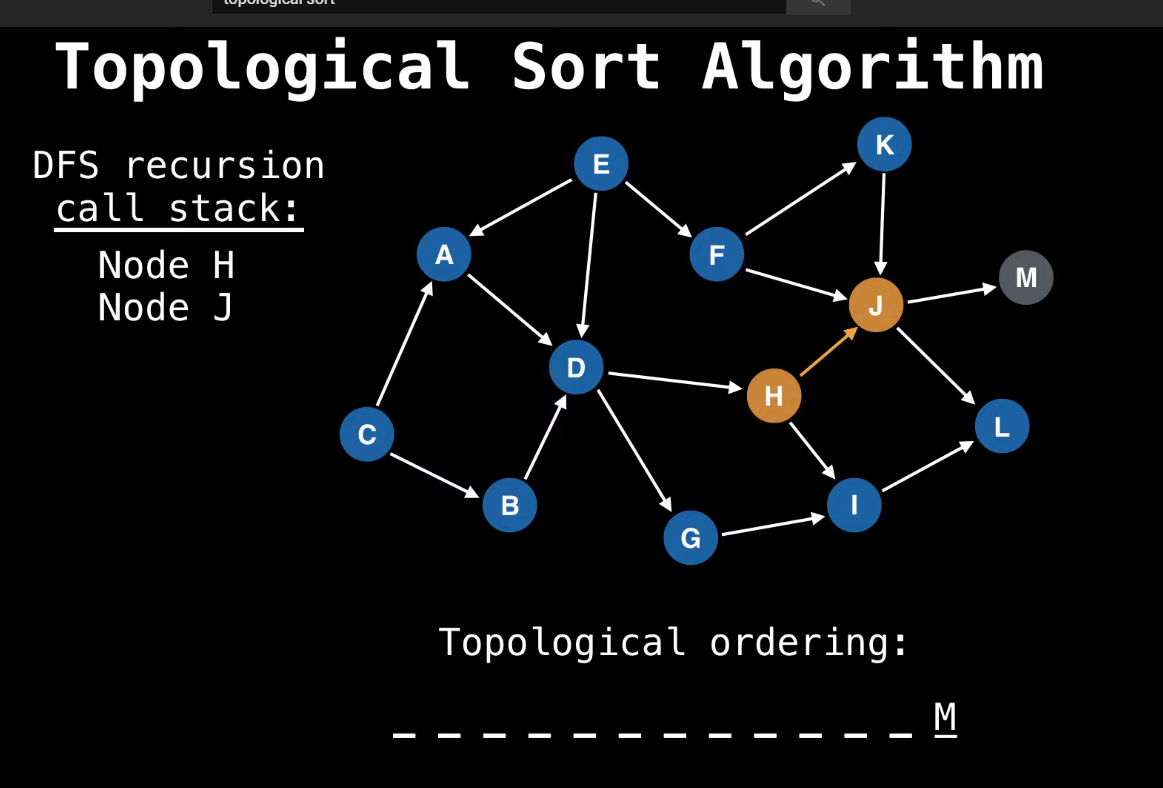
咋进行拓扑排序·

任选一个点例如H，进行DFS

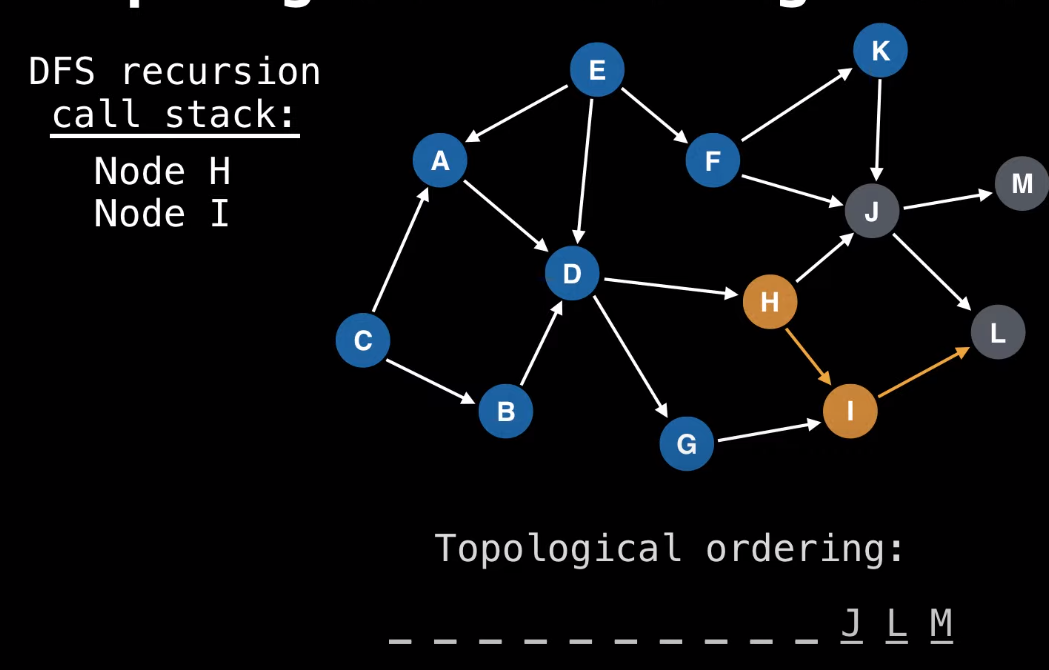


M back track，设置成灰色，stack pop出来，放进topological ordering末尾

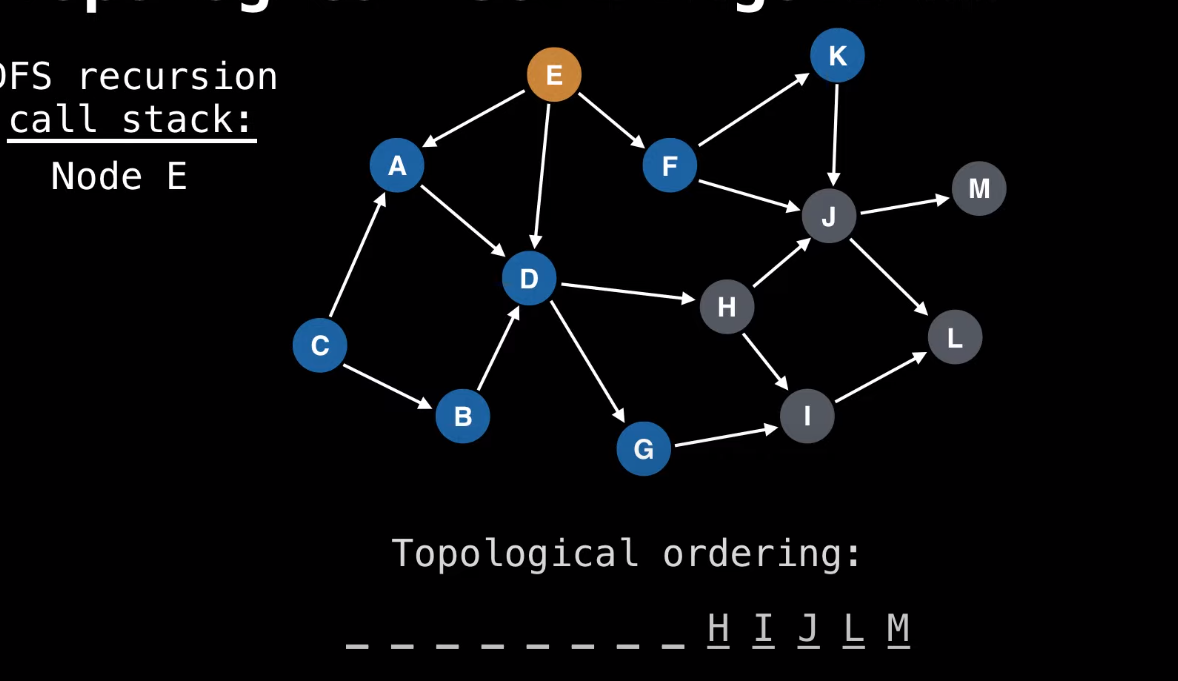




然回J进L，排L排J，回H进I，排I，回H排H



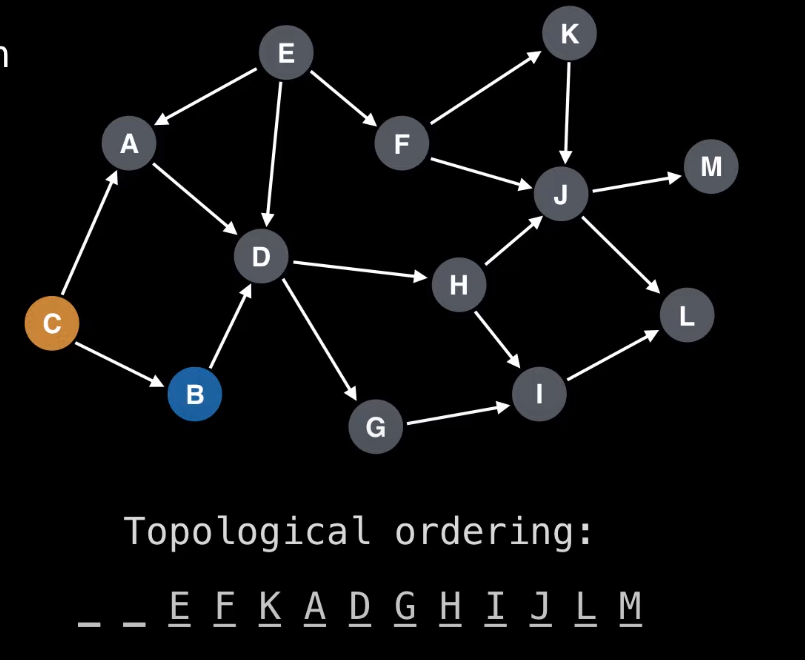
然后再任选一点重复·，例如E



EADG

排G排D排A回E

FK，排K排F 排E

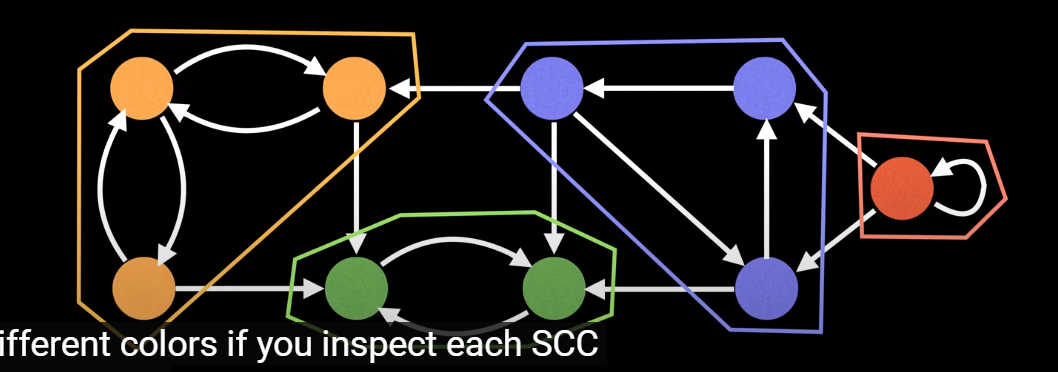


最后CB排B排C

Strongly connected components

两个点有一条直接从A到B的path，也有一条直接从B到A的path，，叫做strongly connected

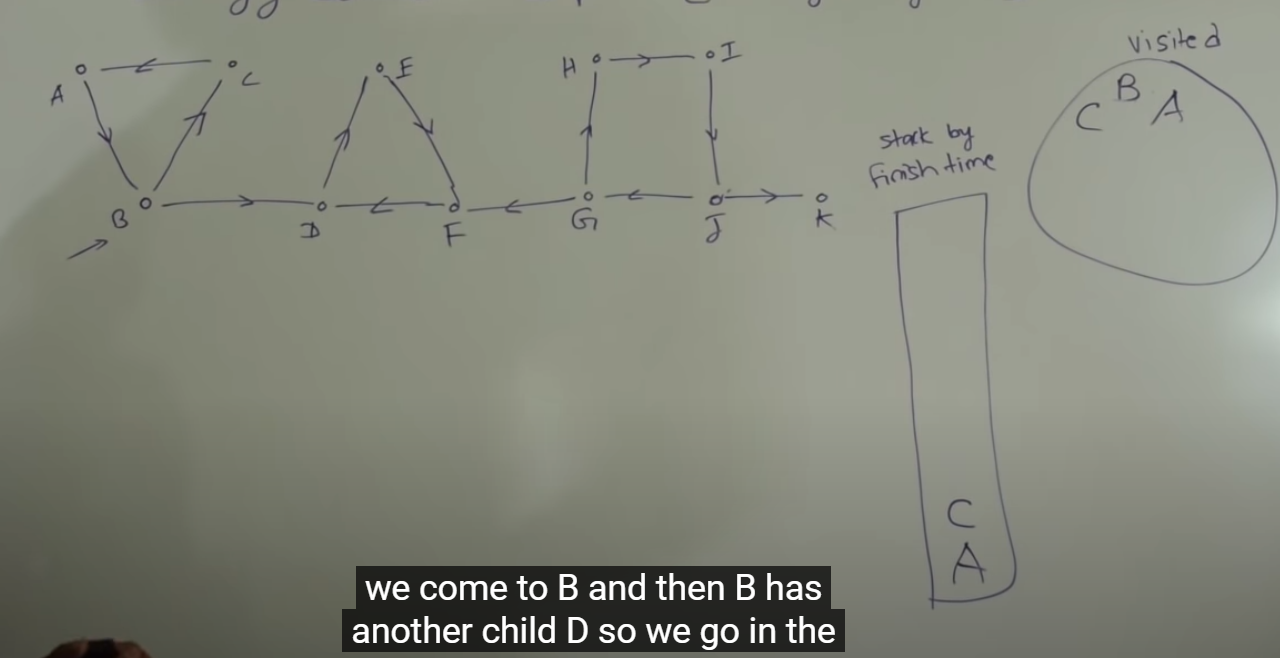
换句话说，有循环



Strongly connected component就是这样一组 vertices

这组vertices的size不能更大（不然·子component也有可能是strongly connected component）

有一点像拓扑，任选一个点开始DFS

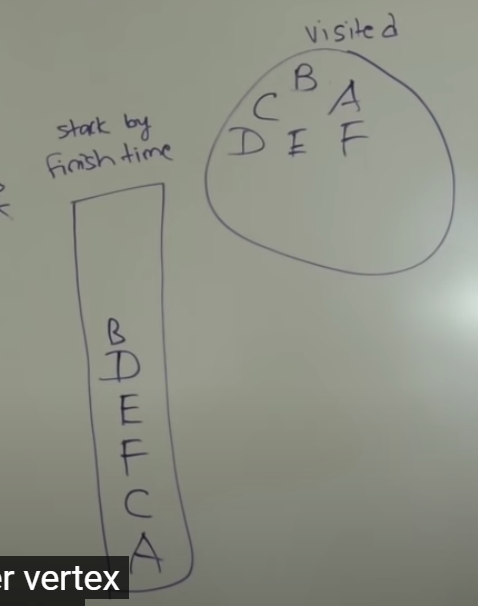


B开头，BCA,这时B已经visited了

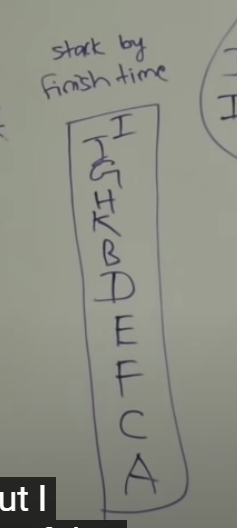
所以A是第一个vertex to finish,所以是第一个被push进去的

B继续DEF

F PUSH,E PUSH D PUSH B PUSH



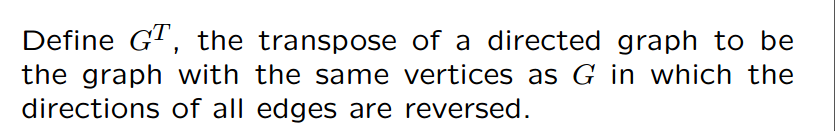
选I继续,IJK , K PUSH, GH, H PUSH, G PUSH ,J PUSH ,I PUSH

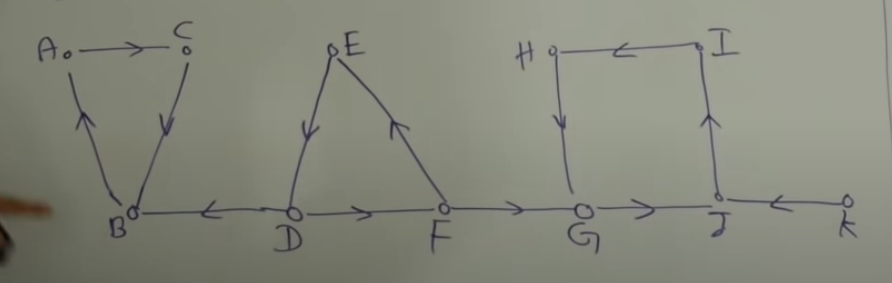


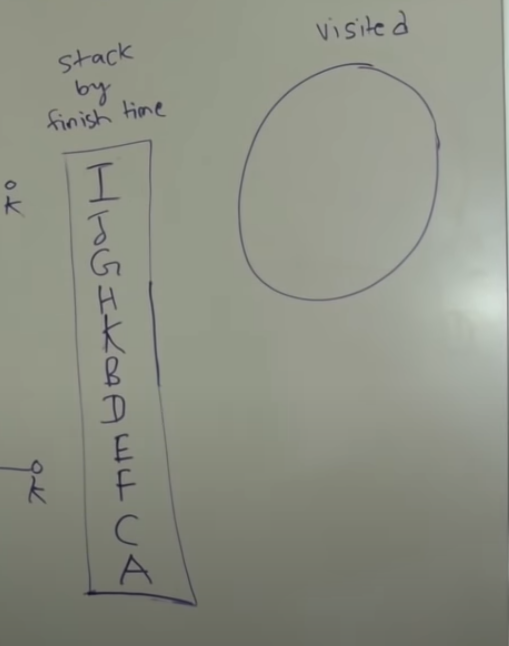
这是我们第一遍loop

STEP2

构建GT,他是相同的vertices,方向相反





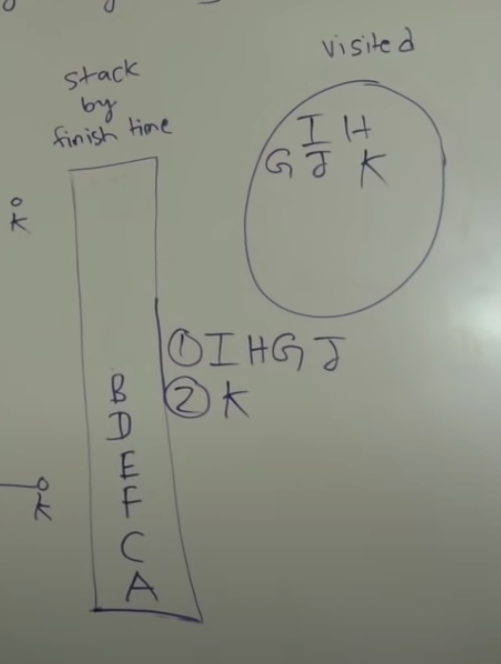


通过decreasing order从上到下处理STACK

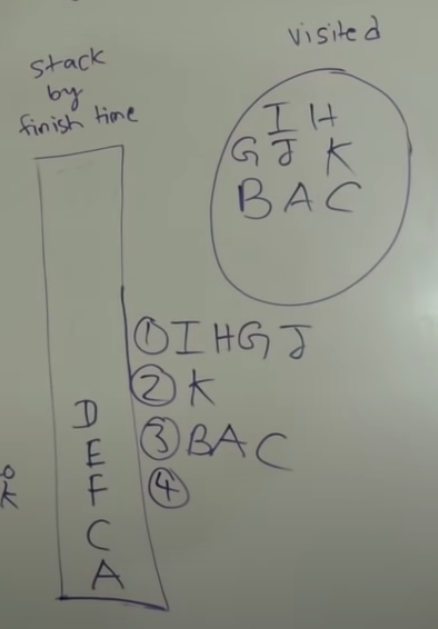
还是使用DFS，但不是随机选取点，而是根据栈顶

I， IHGJ , STACK POP出IJGH

2. 栈顶现在是K，K指向J，visited，舍去， pop

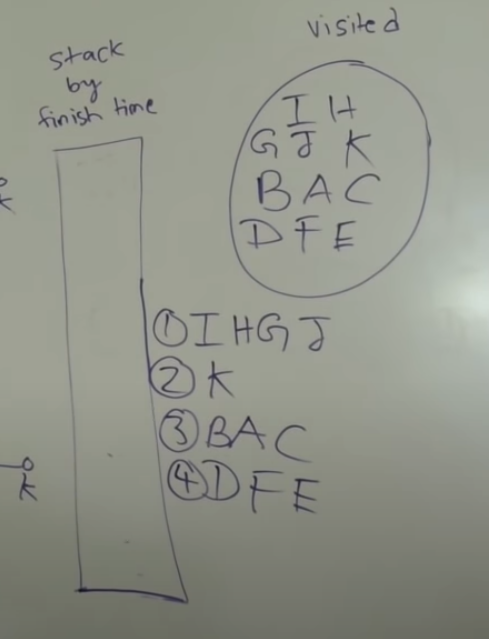


3.栈顶是B.BAC, 只能pop 出一个B



4.栈顶是D,DEF

最后剩CA，已经visited，直接舍去



IHGJ,BAC,DFE就是我们的三个SCC

时间复杂度等于（n+m）就是处理点与edge的时间

Space Complexity is O(v)