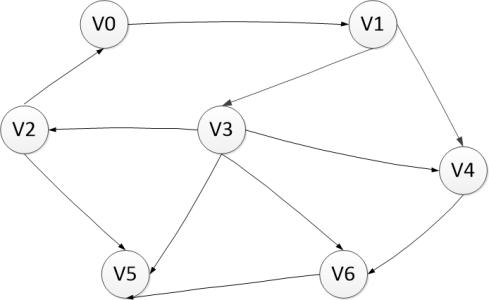
深度优先历遍

示例图：

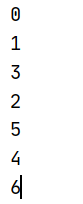


DFS的递归版本：

**public static List<Integer> diguiDfs(int[][] G,int startV){  
 List<Integer> list= new ArrayList<>();  
 boolean[] visit=new boolean[G.length];  
 Arrays.*fill*(visit,false);  
 *diguiDfs*(list,G,visit,startV);  
 return list;  
}  
  
static void diguiDfs(List<Integer> list,int[][] G,boolean[] visit,int curV){  
 list.add(curV);  
 visit[curV]=true;*//标记访问* for(int i=0;i<G[curV].length;i++){  
 if(!visit[i] && G[curV][i]==1){  
 *diguiDfs*(list, G, visit, i);  
 }  
 }  
}**

**public static void main(String[] args) {  
 int[][] G={  
 {1,1,0,0,0,0,0},  
 {0,1,0,1,1,0,0},  
 {1,0,1,0,0,1,0},  
 {0,0,1,1,1,1,1},  
 {0,0,0,0,1,0,1},  
 {0,0,0,0,0,1,0},  
 {0,0,0,0,0,1,1}  
 };  
 List<Integer> list = *diguiDfs*(G, 0);  
 list.forEach(System.*out*::println);  
}**

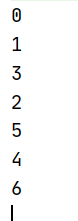
输出历遍结果



非递归的深度优先历遍：

**public static List<Integer> feidiguiDfs(int[][] G,int startV){  
 List<Integer> list=new ArrayList<>();  
 boolean[] visit=new boolean[G.length];  
 Arrays.fill(visit,false);  
 Stack<Integer> stack=new Stack<>();  
 stack.push(startV);  
 visit[startV]=true;  
 list.add(startV);  
 int v=startV;  
 while (!stack.empty()){  
 int i=0;  
 for(;i<G[v].length;i++){  
 if(!visit[i] && G[v][i]==1){  
 stack.push(i);  
 visit[i]=true;  
 list.add(i);  
 v=i;  
 i=-1;  
 }  
 }  
 v=stack.pop();  
 }  
 return list;  
}**

输出结果：



标记访问次序和退栈次序:

static List<Integer> *tuizhancixu*=new ArrayList<>();

**public static List<Integer> feidiguiDfs(int[][] G,int startV){  
 List<Integer> list=new ArrayList<>();  
 boolean[] visit=new boolean[G.length];  
 Arrays.*fill*(visit,false);  
 Stack<Integer> stack=new Stack<>();  
 stack.push(startV);  
 visit[startV]=true;  
 list.add(startV);  
 int v=startV;  
 while (!stack.empty()){  
 v=stack.peek();  
 for(int i=0;i<G[v].length;i++){  
 if(!visit[i] && G[v][i]==1){  
 stack.push(i);  
 visit[i]=true;  
 list.add(i);  
 v=i;  
 i=-1;  
 }  
 }  
 *tuizhancixu*.add(stack.pop());  
 }  
 return list;**  
}

**public static void main(String[] args) {  
 int[][] G={  
 {1,1,0,0,0,0,0},  
 {0,1,0,1,1,0,0},  
 {1,0,1,0,0,1,0},  
 {0,0,1,1,1,1,1},  
 {0,0,0,0,1,0,1},  
 {0,0,0,0,0,1,0},  
 {0,0,0,0,0,1,1}  
 };  
 List<Integer> list = *feidiguiDfs*(G, 0);  
 System.*out*.print("访问标记顺序:");  
 list.forEach(integer -> System.*out*.print(integer+" "));  
 System.*out*.println();  
 System.*out*.print("退栈次序:");  
 *tuizhancixu*.forEach(integer -> System.*out*.print(integer+" "));  
}**

输出结果：



拓扑排序：

把图中的v2 -> v0 断开

public static void main(String[] args) {  
 int[][] G={  
 {1,1,0,0,0,0,0},  
 {0,1,0,1,1,0,0},  
 {0,0,1,0,0,1,0},  
 {0,0,1,1,1,1,1},  
 {0,0,0,0,1,0,1},  
 {0,0,0,0,0,1,0},  
 {0,0,0,0,0,1,1}  
 };  
 List<Integer> list = *feidiguiDfs*(G, 0);  
 System.*out*.print("访问标记顺序:");  
 list.forEach(integer -> System.*out*.print(integer+" "));  
 System.*out*.println();  
 System.*out*.print("退栈次序:");  
 *tuizhancixu*.forEach(integer -> System.*out*.print(integer+" "));  
 System.*out*.print("\n拓扑排序:");  
 Collections.*reverse*(*tuizhancixu*);  
 *tuizhancixu*.forEach(integer -> System.*out*.print(integer+" "));  
}

输出结果：

