Давайте определим граф как список преемников, определенный как тип (мы используем так называемый синтаксис записи, который дает нам возможность использовать сопоставление с образцом как Graph v n, а также определяет две функции peaks::Graph a -> [a] и successors::Graph a -> [(a, [a])], который используется для доступа к вершинам и преемникам без сопоставления с образцом).

data Graph a = Graph {peaks :: [a], successors' :: [(a, [a])]} deriving Show

Напишите функцию topoSort :: Eq a => Graf a -> [a], которая возвращает список вершин указанного графа в топологическом порядке.

Примеры входов и выходов:

- сначала определяем графики для тестов

graf1 = Graf [1,2,3][(1,),(2,),(3,)]

graf2 = Graf [1,2,3,4,5][(1,),(2,),(3,),(4,),(5,)]

graf3 = Graf [1,2,3,4,5,6,7,8,9][(1,),(2,),(3,),(4,),(5,),(6,),(7,),(8,),(9,)]

1.
-> topoSort graf1
[1,3,2]

2.
-> topoSort graf2
[1,3,2,5,4]

3.
-> topoSort graf3
[6,7,8,2,1,5,3,4,9]