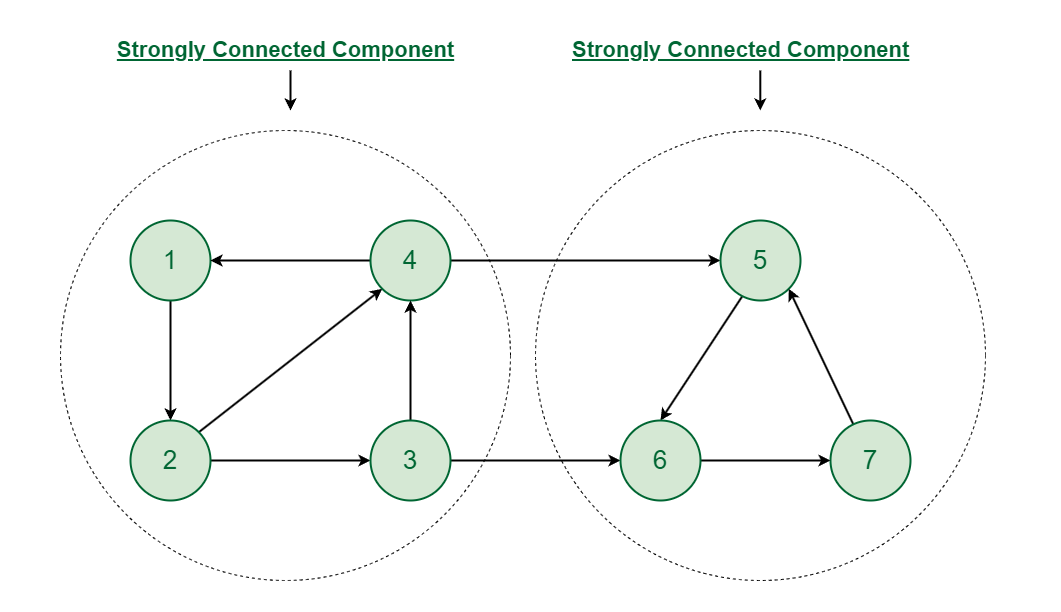
**Алгоритъм на Kosaraju за силно свързани компоненти и топологично сортиране**

(Изготвил за курса по СДП, спец. „Математика и информатика“: Михаела Папаринова, спец. „Анализ на данни“, 2023 г.)

1. Силно свързан компонент е компонентът на насочен граф, който има път от всеки връх до всеки друг връх в този компонент.

Например, графът по-долу има два силно свързани компонента {1,2,3,4} и {5,6,7}, тъй като има път от всеки връх до всеки друг връх в същия силно свързан компонент.



Два върха се наричат ​​свързани, ако между тях има път. Междувременно понятието „силно свързан“ е приложим само за насочени графи. Подграф на насочен граф се счита за силно свързан тогава и само тогава, ако за всяка двойка върхове A и B, съществува път от А до B и от B до A. DFS не работи за откриване на свързани компоненти. Ако започнем от първия връх можем да стигнем до всички други върхве, но ако започнем от другите върхове, няма да можем да стигнем до всички останали.

Ако направим DFS от всеки връх, това ще ни отнеме O(V\*(V+E)) време.

Оптимално решение е да използваме алгоритъма на Kosaraju.

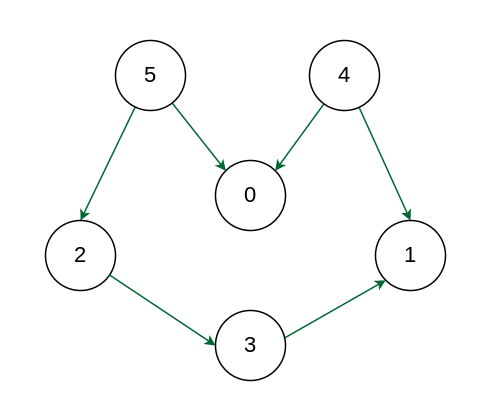
1. Инициализирайте всички върхове като непосетени.
2. Направете DFS обхождане на графa, като се започва от произволен връх v. Ако DFS обхождането не посети всички върхове, тогава върнете False.
3. Намерете транспонирания или обърнатия (reverse) граф.
4. Маркирайте всички върхове като непосетени в обърнатия граф.
5. Направете DFS обхождане на обърнатия граф, започвайки от същия връх v (Същото като стъпка 2). Ако обхождането на DFS не посети всички върхове, върнете False. В противен случай върнете True.

Идеята е, че ако всеки връх може да бъде достигнат от връх v и всеки връх може да достигне v, тогава графът е силно свързан. В стъпка 2 проверяваме дали всички върхове са достижими от v. В стъпка 4 проверяваме дали всички върхове могат да достигнат v (В обърнатия граф, ако всички върхове са достижими от v, тогава всички върхове могат да достигнат v в оригиналния граф).

Сложност по време:  O(V+E)

Сложност по памет: O(V)

1. Топологичното сортиране за насочен ацикличен граф е линейно подреждане на върховете, така че за всеки насочен ръб u-v, върхът u идва преди v в подредбата. Важи само за насочен ацикличен граф.



**Изход:**5 4 2 3 1 0  
**Обяснение:** Първият връх в топологичното сортиране винаги е връх с входяща степен 0 (връх без входящи ръбове). Топологичното сортиране на следната графика е “5 4 2 3 1 0”. Може да има повече от едно топологично сортиране за един граф. Друго топологично сортиране на следната графика е „4 5 2 3 1 0“.

* Използвайте временен стек.
* Първо рекурсивно извиквайте топологично сортиране за всички негови съседни върхове, след което първоначалния връх се добавя в стека.
* Накрая отпечатайте съдържанието на стека.

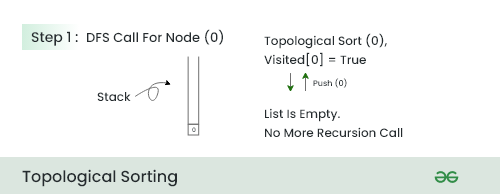
**Забележка:** Един връх се добавя в стека само когато всички негови съседни върхове (и техните съседни върхове и т.н.) вече са в стека.

Как ще работи кодът?

* Създайте стек за съхраняване на върховете.
* Инициализирайте масив с размер N, за да запазите запис на посетените върхове.
* Стартиране на цикъл от 0 до N :
* Ако върхът не е маркиран като посетен, то извикайте рекурсивната функция за топологично сортиране и:
  + Маркирайте текущия връх като посетен.
  + Стартирайте цикъл на всички върхове, които имат насочен ръб към текущия връх.
  + Ако върхът не е маркиран като посетен:
    - Рекурсивно извиквайте функцията за топологично сортиране на върха.
  + Избутвайтe текущия връх в стека.
* Отпечатайте всички елементи от стека.

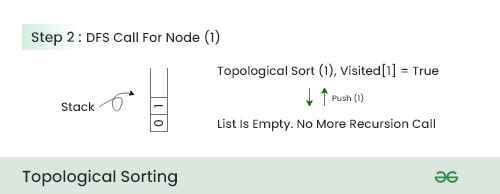
Стъпка 1:

* Стартираме DFS от връх 0, защото има нула входящи върхове.
* Добавяме връх 0 в стека и преминаваме към следващия връх с минимален брой съседни върхове, т.е. връх 1.



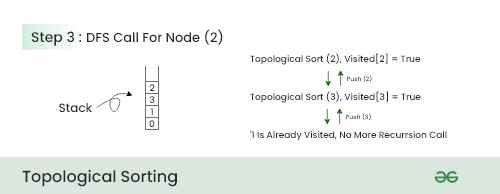
Стъпка 2:

* В тази стъпка, тъй като няма съседен на този връх, добавяме връх 1 в стека и преминаваме към следващия връх.



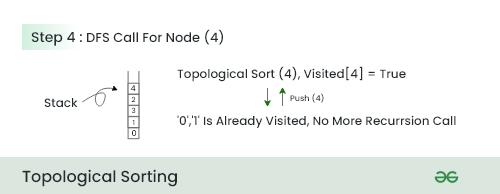
Стъпка 3:

* В тази стъпка избираме връх2, защото има минимален брой съседни върхове след 0 и 1.
* Извикваме DFS за 2 и добавяме всички върхове, които излизат от връх 2 в обратен ред.
* Добавяме 3 и след това добавяме 2.



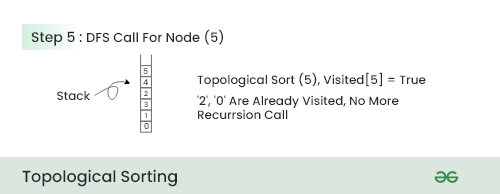
Стъпка 4:

* Сега извикваме DFS за връх 4
* Тъй като 0 и 1 вече са в стека, просто добавяме връх 4 в стека и връщаме.



Стъпка 5:

* В тази стъпка, тъй като всички съседни възли на 5 вече са в стека, ние добавяме връх 5 в стека и връщаме.



**Стъпка 6:**Отпечатваме всички елементи на стека.

Сложност по време:  O(V+E)

Сложност по памет: O(V)

**Източници**

1. Geeks for Geeks, Strongly Connected Components, <https://www.geeksforgeeks.org/strongly-connected-components/>, последнаааа редакция: 05.07.2024, последно посетен: 05.10.2024

2. Geeks for Geeks, Topological Sorting, <https://www.geeksforgeeks.org/topological-sorting/>, последна редакция: 23.07.2024, последно посетен: 05.10.2024