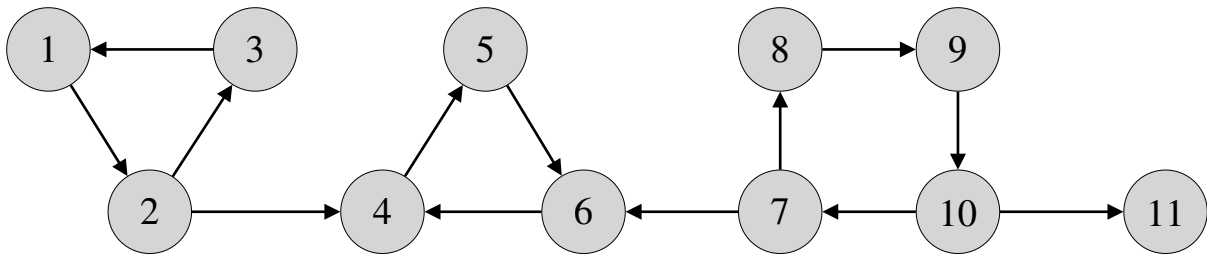


Алгоритъм на Kosaraju-Sharir за силно свързани компоненти (в насочен граф)



Силно свързана компонента е такава компонента, в която всеки връх в компонентата е достижим от всеки друг връх в нея. На фигурата горе имаме 4 различни компоненти на свързаност. Например $\{1,2,3\}$:

- ➔ 1 е достижим връх от 2 и 3
- ➔ 2 е достижим връх от 3 и 1
- ➔ 3 е достижим връх от 1 и 2

Аналогично $\{4,5,6\}$, $\{7,8,9,10\}$ и $\{11\}$ са силно свързани компоненти.

Приложения: например в инстаграм - социална мрежа наподобяваща фейсбук с фундаменталната разлика, че всеки потребител е представен като връх в насочен граф (докато във фейсбук е ненасочен). Инстаграм може да използва този алгоритъм, за да намери всички силно свързани групи (компоненти) от потребители и да търси в тях общ интерес или каквото и да е в тази компонента. Друго приложение е това което сме използвали в задачата за летищните връчки - за компресация на граф.

Има няколко алгоритъма, които намират силно свързани компоненти и един от тях е този на Kosaraju-Sharir. Този алгоритъм минава през графа 2 пъти.

► Първо обхождане:

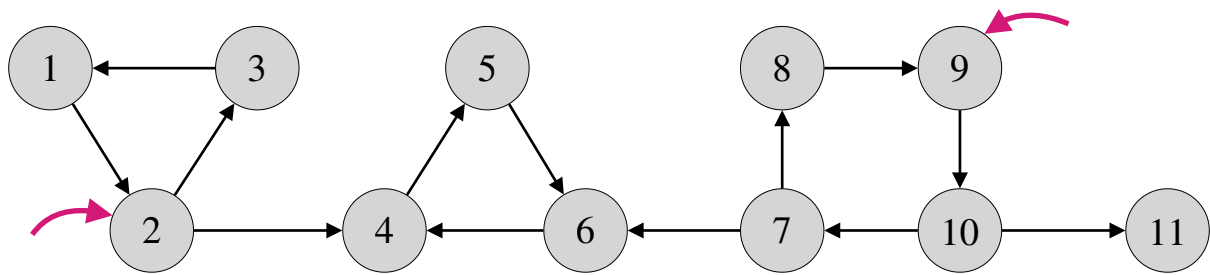
Обхождаме графа в дълбоична (dfs) започвайки от непосетен връх и нареждаме върховете на графа според момента на последното им посещаване в намаляващ ред (в момента, в който всички съседни на текущия връх са посетени - излизаме от него и го добавяме в стека). Ще имаме нужда и от множество visited, до което ще се допитваме, когато искаме да разберем дали даден връх е посетен.

► Обръщане на посоката на ребрата:

Обръщаме ребрата на насочения граф и създаваме нов списък на съседства. Това може да стане успоредно с действията в първото обхождане (но може и да е като отделно обхождане - повишаването на скритата константа от 2 към 3 не оказва особено влияние на общата времева сложност на алгоритъма)

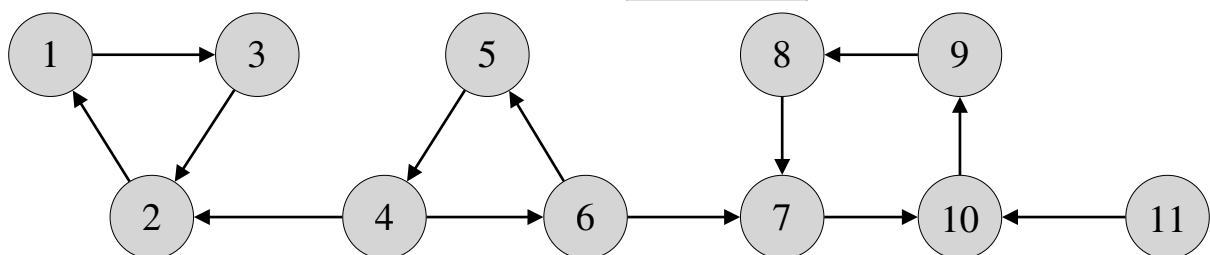
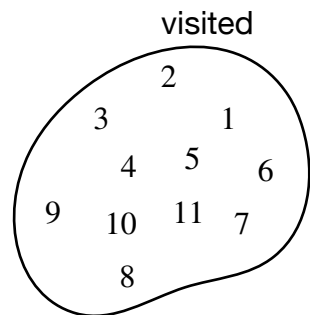
► Второ обхождане:

Докато има връх на графа в стека - взимаме го и ако не е посетен пускаме dfs от него и на всеки връх обходен от това dfs ще му бъде прикачен представител от съответната компонента на свързаност (именно текущия връх).



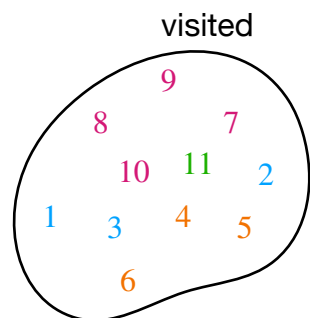
original graph

stack by finish time
9
10
7
8
11
2
4
5
6
3
1



reversed graph

stack by finish time
9
10
7
8
11
2
4
5
6
3
1



Силно свързани компоненти:

$\oplus \{9,8,7,10\}$ – първа

$\oplus \{11\}$ – втора

$\oplus \{2,1,3\}$ – трета

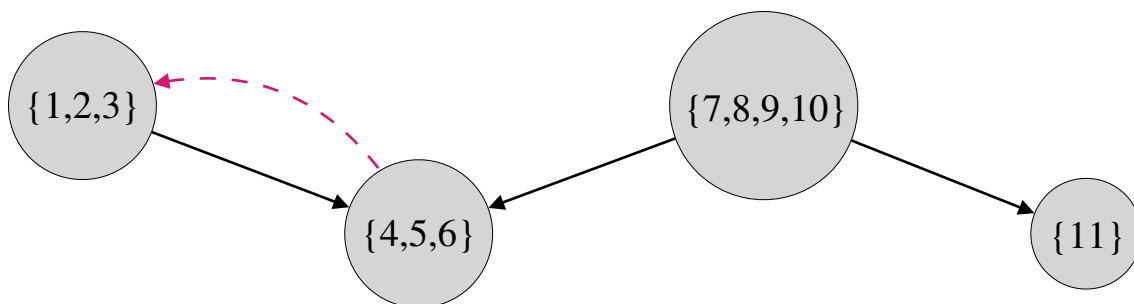
$\oplus \{4,5,6\}$ – четвърта

Нека анализираме времевата сложност и сложността по памет. Първото обхождане в дълбочина отнема $O(|V| + |E|)$, колкото отнема и обръщането на посоките на ребрата на графа. Сложността по памет е от порядъка на $O(|V|)$.

Защо този алгоритъм работи?

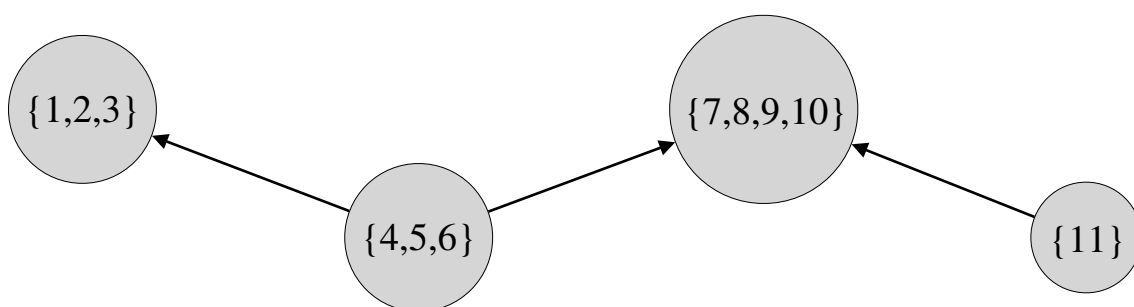
Ще разгледаме само интуицията зад него вместо чистото математическото доказателство.

Нека компресираме силно свързаните компоненти по следния начин: всяка компонента ще заместим с отделен връх (представител). Нека разгледаме новополучения граф:



Резултатния граф след компресацията е гарантирано, че ще бъде ацикличен, защото, ако допуснем например, че има ребро от $\{4,5,6\}$ към $\{1,2,3\}$, то ще направи така, че двата върха да участват в една компонента на свързаност, което е противоречие с допускането, че всяка силно свързана компонента е компресирана.

В dfs-то правим следното: когато всички деца на текущия връх са обходени, тогава добавяме текущия връх в стека. Тогава наличието на ребро от $\{1,2,3\}$ към $\{4,5,6\}$ ще гарантира, че поне един връх от първата компонента на свързаност ще бъде добавен в стека след като всички върхове от втората компонента на свързаност вече са добавени в стека.



Обръщането на ребрата не променя абсолютно нищо в силно свързаните компоненти, което да касае алгоритъма. В тях всеки връх ще продължава да е достижим от всички останали върхове в компонентата. **Но мостовете между различните компоненти ще са сменили посоките си сега.** Тоест благодарение на стека ще си гарантираме, че поне по един връх от двете горни компоненти на фигурата ще е избран преди всички върхове от двете долни компоненти и след избирането му ще обходим цялата компонента.