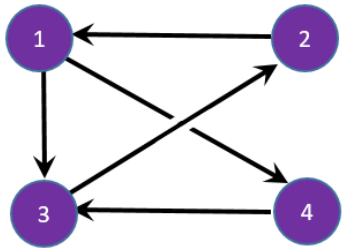
Silnie spójne składowe

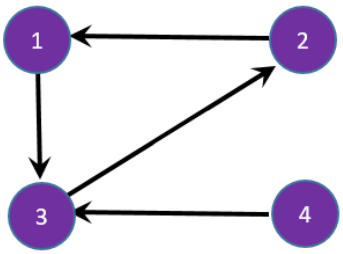
#graf_transponowany
#relacja_zwrotna #relacja_symetryczna
#relacja_przechodnia #relacja_równoważnościowa
#klasy_równoważności

Mówiliśmy już o spójnych składowych grafu nieskierowanego: tak graf lub jego fragment nazywany był spójnym, jeśli można było dostać się z każdego wierzchołka w nim do dowolnego innego w tym samym fragmencie. W przypadku grafów skierowanych mówi się raczej o *silnie spójnych składowych* (ang. **SSC**, **Strongly Connected Components**), wymawiaj: *strongly konekted komponents* z akcentem na pierwszą sylabę w ostatnim słowie).

Generalnie spójność grafu skierowanego (silną spójność) jako całości określa się tak samo, jak grafu nieskierowanego. Oto przykład grafu silnie spójnego:

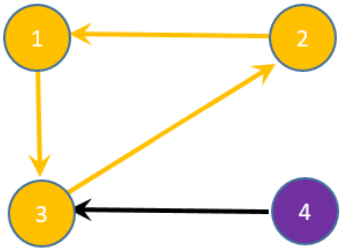


Da się przejść z każdego do każdego wierzchołka. Jeśli natomiast usuniemy na przykład krawędź $1 \to 4$, wtedy graf już nie ma tej własności:



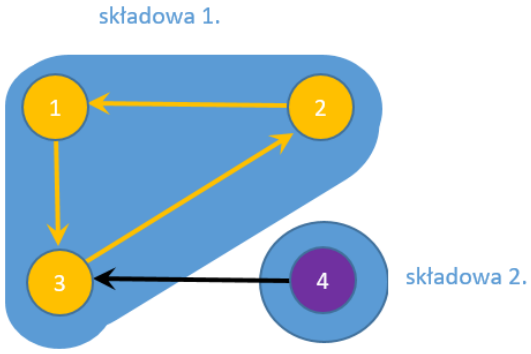
Na ogół jednak graf jako całość nie wykazuje cech silnej spójności, natomiast zawsze można w nim wyróżnić pewne fragmenty, które – każdy z osobna – są silnie spójne.* Na przykładzie powyższego grafu widać, że wierzchołki $\{1, 2, 3\}$ tworzą silnie spójną strukturę:

*Niech Czytelnik doczyta jeszcze kilka najbliższych akapitów i zastanowi się, dlaczego tak jest.



Ważną cechą silnie spójnej składowej jest fakt, że nie da się już dołączyć żadnego wierzchołka grafu, aby nie zepsuć silnej spójności. Jest to więc struktura o maksymalnej możliwej wielkości.

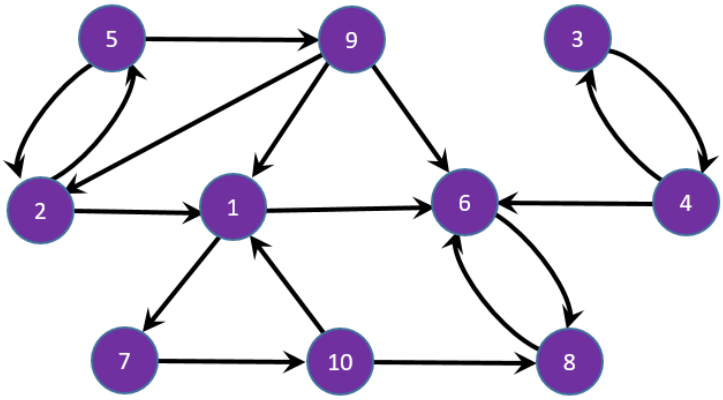
Zaraz, zaraz, a co z wierzchołkiem 4? Taki samotny wierzchołek może być samodzielną silnie spójną składową – i tak jest w tym przypadku, gdyż dołączenie jakiegokolwiek innego wierzchołka zaburzy spójność tej składowej. Tak więc nasz graf możemy podzielić na dwie rozłączne silnie spójne składowe.[†]:



Pomiędzy poszczególnymi silnie spójnymi składowymi mogą istnieć krawędzie (u nas jest jedna), ale do tego jeszcze wrócimy.

Graf transponowany

Zobaczymy, jak to wszystko wygląda w przypadku jakiegoś bardziej wypasionego grafu, na przykład:



[†]Z samej definicji silnie spójnych składowych – tutaj przedstawionej w formie opisowej – wynika, że muszą być one rozłączne (dlaczego?).

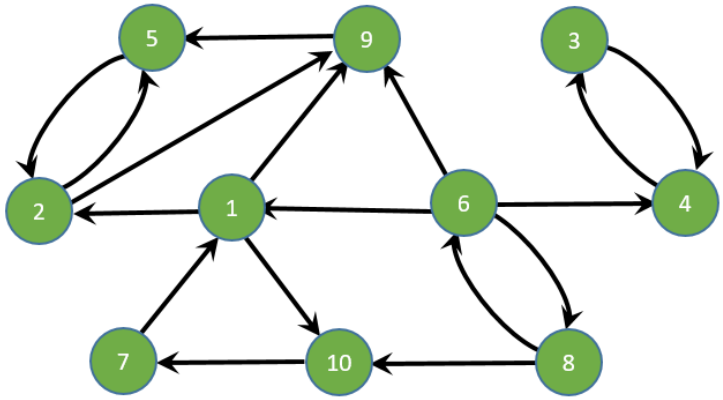
Przygotujemy dane wejściowe dla tego grafu:

```
10 17
2 5
5 2
1 7
2 1
10 1
9 2
7 10
5 9
9 1
9 6
10 8
6 8
8 6
1 6
4 6
3 4
4 3
```

Do ich wczytania wykorzystamy naszą standardową funkcję `read_directed_graph()`, a otrzymane listy sąsiedztwa wypisze nam funkcja `print_graph()`:

```
1: 7 6
2: 5 1
3: 4
4: 6 3
5: 2 9
6: 8
7: 10
8: 6
9: 2 1 6
10: 1 8
```

Ale teraz będzie nowość: oprócz „normalnego” grafu G zbudujemy również *graf transponowany* G^T (ang. **transposed graph**, wymawiaj: *transpołzd graf*, z akcentem na *o*). Jest to graf otrzymany z grafu G przez odwrócenie zwrotów wszystkich krawędzi, czyli w tym przypadku:



Ten graf będzie nam potrzebny do ostatecznej identyfikacji silnie spójnych składowych, co przedstawimy w następnej sekcji. Na razie wygenerujemy ten transponowany graf przy pomocy funkcji `transpose_graph()`:[‡]

```
void transpose_graph(vector<vector<int>> &G, int V,
                     vector<vector<int>> &GT)
{
  GT.resize(V + 1);
  for(int v = 1; v <= V; v++)
    for(int x : G[v])
      GT[x].push_back(v);
}
```

Tym samym krystalizuje się nam początkowa część funkcji `main()`:

```
int main()
{
  int V, E;
  vector<vector<int>> G;
  read_directed_graph(G, V, E);
  print_graph(G, V);
  transpose_graph(G, V, GT);
  print_graph(GT, V);
  . . .
}
```

Funkcja `print_graph()` dla grafu transponowanego G^T pokaże następujące listy sąsiedztwa:

```
1: 2 9 10
2: 5 9
3: 4
4: 3
5: 2
6: 1 4 8 9
7: 1
8: 6 10
9: 5
10: 7
```

Na razie tyle, jeśli chodzi o graf transponowany – przechodzimy do następnego etapu.

Pierwsze przeszukiwanie w głąb

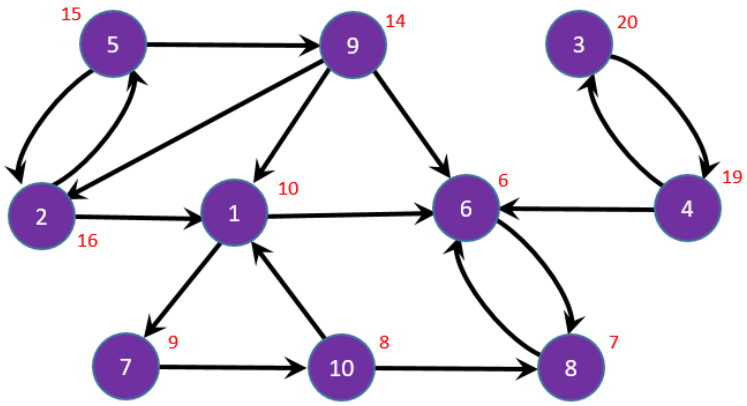
Najpierw przeszukamy w głąb graf G odkładając na stosie S numery wierzchołków w kolejności zakończenia ich przetwarzania – tak samo, jak robiliśmy podczas sortowania topologicznego.

[‡]Można to zrobić już na etapie wczytywania danych, ale wtedy musielibyśmy zmodyfikować – jakże zasłużoną – funkcję `read_connected_graph()`.

Zatem kolejna instrukcja w funkcji `main()` to wywołanie funkcji realizującej algorytm DFS (poprzedzone deklaracjami niezbędnych zmiennych):

```
vector<bool> visited(V + 1);
vector<int> parent(V + 1), start(V + 1), finish(V + 1);
int t = 0;
stack<int> S;
DFS_topological(G, V, visited, parent, t, start, finish, S);
```

Na rysunku zaznaczymy tylko wartości czasów przetworzenia (*finish* – na czerwono, bez żadnej adnotacji):



Oznacza to, że na stosie odłożyły się numery wierzchołków w następującej kolejności:

3 (wierzch stosu)
4
2
5
9
1
7
10
8
6

Drugie przeszukiwanie w głąb

Następny etap znajdowania silnie spójnych składowych polega na uruchomieniu przeszukiwania w głąb dla grafu G^T, przy czym kolejne wierzchołki startowe pobierać będziemy ze stosu S. W tym celu napiszemy nieco zmodyfikowane wersje funkcji `DFS_topological()` oraz `DFS_visit_topological()`: odpowiednio `DFS_SCC()` oraz `DFS_SCC_visit()`.

Ta pierwsza funkcja będzie zawierać petlę `while` operującą na stosie S i jej zadaniem będzie ustalenie numeru silnie spójnej składowej (zmienna c), do której należy dany wierzchołek (przechowamy to w wektorze *component*; numeracja składowych zacznie się od 1):

```
void DFS_SCC(vector<vector<int>> &GT, int V, vector<bool> &visited,
             stack<int> &S, vector<int> &component)
{
  int c = 0;
  while(!S.empty())
  {
    int v = S.top();
    S.pop();
    if(!visited[v])
      DFS_visit_SCC(GT, V, v, visited, component, ++c);
  }
}
```

Użycie operatora zwiększania w ostatnim argumencie wywołania funkcji `DFS_visit_SCC()` chyba nie speszyło Czytelnika?

Jednym słowem, DFS uruchamiamy (tak już na serio) dla kolejnych wierzchołków grafu G^T – o ile są nieodwiedzone.

Teraz funkcja odwiedzająca:

```
void DFS_visit_SCC(vector<vector<int>> &GT, int V, int s,
                   vector<bool> &visited, vector<int> &component, int c)
{
  visited[s] = true;
  component[s] = c;
  for(int x : GT[s])
    if(!visited[x])
      DFS_visit_SCC(GT, V, x, visited, component, c);
}
```

Jedyną nowinką jest przypisanie odwiedzanego wierzchołka do konkretnej silnie spójnej składowej (o numerze c).

Tak powinna wyglądać końcowa część funkcji `main()` (pamiętamy o wyczyszczeniu wektora *visited*):

```
fill(visited.begin(), visited.end(), false);
vector<int> component(V + 1);
DFS_SCC(GT, V, visited, S, component);
for(int v = 1; v <= V; v++)
  cout << v << ": " << component[v] << endl;
```

Na końcu dodaliśmy wypisanie zawartości wektora *component* – zobaczymy następujący wydruk:

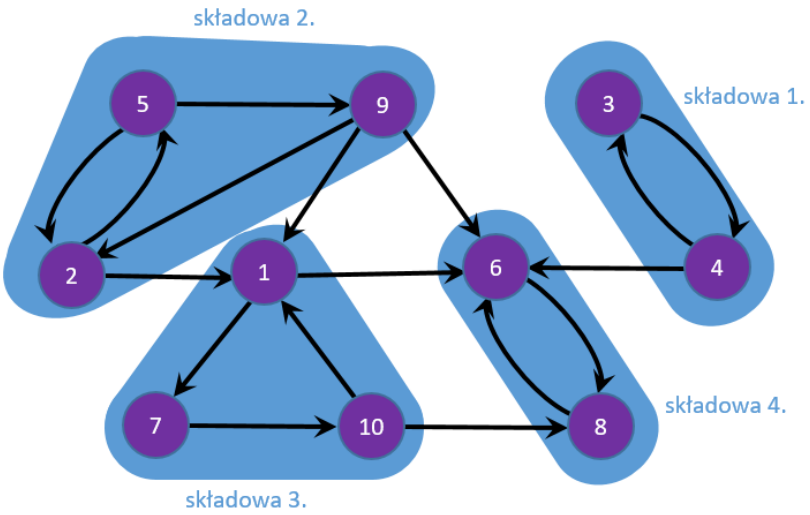
```
1: 3
2: 2
3: 1
```

```
4: 1
5: 2
6: 4
7: 3
8: 4
9: 2
10: 3
```

Widzimy tu cztery silnie spójne składowe:

1. wierzchołki 3, 4,
2. wierzchołki 2, 5, 9,
3. wierzchołki 1, 7, 10,
4. wierzchołki 6, 8.

Wygląda to następująco:



Nie będziemy tutaj roztrząsać, dlaczego ten algorytm działa,[§] ważne, że przy jego pomocy możemy rzeczywiście znaleźć silnie spójne składowe każdego grafu.

Klasy równoważności

Wyobraźmy sobie zbiór A par wszystkich wierzchołków grafu G:

$$A = \{(a, b) : 1 \leqslant a \leqslant V, 1 \leqslant b \leqslant V\},$$

gdzie wierzchołki reprezentowane są (wzajemnie jednoznacznie) przez liczby całkowite z zakresu od 1 do V. Ilość elementów zbioru A wynosi V^2, gdyż dopuszczamy sytuację, aby $a = b$.

W tym zbiorze wyodrębnimy pewien podzbiór $R \subseteq A$. Do tego podzbioru zaliczymy pary wierzchołków, które należą do tej samej silnie spójnej składowej (obojętnie której, byle tej

[§]Dowód jego poprawności można znaleźć na przykład w podręczniku Cormena i spółki.

samej). Na przykład w zbiorze R znajdą się pary $(3, 4)$, $(2, 5)$ czy $(7, 10)$, natomiast nie ma w nim na przykład pary $(1, 9)$ ani $(9, 1)$.

Dowolny podzbiór zbioru A nazywany jest *relacją*, zatem zbiór R jest przykładem relacji – i to relacji o bardzo szczególnych cechach.

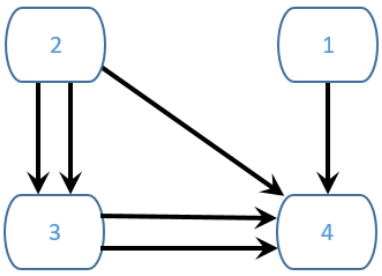
Zauważmy, że wszystkie pary postaci (a, a), $a = 1, 2, \ldots, V$ należą do zbioru R (relacji) (dlaczego?). Jeśli relacja spełnia ten warunek, wtedy nazywana jest *relacją zwrotną* (to znaczy, że każdy wierzchołek jest w relacji z samym sobą).

Ponadto, jeśli para (a, b) należy do relacji, wtedy na pewno para (b, a) także do relacji należy.[¶] Na przykład, skoro para $(6, 8)$ należy do relacji, wtedy para $(8, 6)$ również należy do R. Taka relacja nazywa się *relacją symetryczną*.

I jeszcze trzecia własność: jeśli dla dowolnych par (a, b) oraz (b, c) ze zbioru A z tego, że obie te pary należą do relacji, wynika że para (a, c) również należy do relacji.[‖] Taka relacja nazywana jest *relacją przechodnią*.

Jeśli relacja jest zwrotna, symetryczna i przechodnia, wtedy nazywana jest *relacją równoważnościową*. Taki typ relacji zachodzi właśnie w przypadku przynależności par wierzchołków do tej samej silnie spójnej składowej. Cały zbiór wierzchołków grafu można podzielić na rozłączne podzbiory – każdy z nich odpowiada oddzielnej składowej. Dowolna para wierzchołków pochodzących z jednego takiego podzbioru jest ze sobą w relacji (czyli należy do R) – i na odwrót: jeśli dwa wierzchołki pochodzą z dwóch odrębnych podzbiorów, wtedy na bank nie są w relacji. Takie podzbiory pełnią ważną rolę w teorii grup i noszą nazwę *klas równoważności*.

Zauważmy, że w przypadku składowych grafu wewnętrzny układ krawędzi w danej składowej jest drugorzędny. W wielu zastosowaniach silnie spójną składową traktuje się jako jedną całość, a poszczególne składowe są połączone krawędziami, które nie są wewnętrzne. Jeśli silnie spójne składowe potraktujemy jako wierzchołki nowego grafu, wtedy w naszym przypadku otrzymamy coś takiego (numery 1–4 to numery silnie spójnych składowych):



Otrzymaliśmy specyficzny graf: pomiędzy poszczególnymi jego wierzchołkami może wystąpić wiecej niż jedna krawędź (czyli jest to tak zwany *multigraf*), ale daleko ważniejszy jest fakt, że tak utworzony graf musi być acykliczny (!). Pozostawiamy dociekliwemu Czytelnikowi zastanowienie się, dlaczego tak musi być.

[¶] Warunek ten musi zachodzić dla wszystkich par należących do zbioru A.

[‖] Skoro $(a, b) \in R$, więc a i b mają ten sam numer silnie spójnej składowej. Również (b, c) mają ten sam numer – i musi to być ten sam numer, co przy poprzedniej parze, gdyż w obydwu parach powtarza się wierzchołek b, zaś przydział wierzchołków do składowych jest jednoznaczny.