# Zadanie

Numer indeksu autora sprawozdania to 234844. Reszta z jego dzielenia przez 13 to 12, a z dzielenia przez 3 to 1. W związku z tym wykonano następujące zadania:

* Pobrano graf zależności pomiędzy artefaktami Mavena ( <https://github.com/ogirardot/meta-deps/blob/master/mvn-deps.csv.lzma> )

**NetworkX:**

* Wczytano pobrany graf.
* Wyznaczono liczbę składowych spójnych oraz rząd i rozmiar największej z nich.
* Dla największej składowej spójnej wyznaczono asortatyność średnią dla różnych stopni wierzchołków oraz współczynnik asortatywności Pearsona (oraz wykres prezentujący dane na podstawie których był wyliczony współczynnik).

**Pajek:**

* Wczytano graf, który wyeksportowano wcześniej z NetworkX
* Wyznaczono liczbę składowych spójnych oraz rząd i rozmiar największej z nich.
* Porównano czasy obliczeń wyznaczania liczby składowych spójnych oraz rzędu i rozmiaru największej z nich w NetworkX oraz Pajek

# Przygotowanie danych

Dane pobrano z <https://github.com/ogirardot/meta-deps/blob/master/mvn-deps.csv.lzma> , a następnie rozpakowano do formatu CSV. Lista zależności dla każdego artefaktu została zakodowana w base64, więc zdekodowano je. Z pliku usunięto za pomocą Notepad++ wiele błędów – głównie dużo znaków | wraz z tabulatorami i nowymi liniami w różnych nieprawidłowych miejscach (dane opisujące jeden artefakt powinny się znajdować w jednej linii, a bywały rozdzielone na wiele). Poprawione dane znajdują się w załączonym archiwum mvn-deps-fixed.zip.

W trakcie tworzenia obiektu Graph w NetworkX okazało się, że wiele artefaktów (234877 z 340913) powtarza się w pliku wejściowym. Zgodnie z dokumentacją (np. <https://networkx.github.io/documentation/networkx-2.0/reference/classes/generated/networkx.Graph.add_edge.html#networkx.Graph.add_edge> ) oraz własnym sprawdzeniem, próba dodania istniejącej krawędzi lub wierzchołka do obiektu klasy Graph, spowoduje co najwyżej zaktualizowanie atrybutów wierzchołka. Krawędzie ani wierzchołki nie zostaną zduplikowane, ani nie zmieni się w żaden sposób istniejąca struktura grafu.

Przy dodawaniu zależności między artefaktami, czyli krawędzi, okazało się, że wiele artefaktów nie zostało zdefiniowanych w pliku źródłowym, a więc przy tworzeniu krawędzi do nich, automatycznie brakujące wierzchołki zostały dodane. Takich brakujących wierzchołków było 97593.

Podczas tworzenia oraz po utworzeniu obiektu Graph sprawdzono zgodność ilości wierzchołków i krawędzi w utworzonym grafie z danymi z pliku źródłowego (włączając liczby powtórzeń oraz brakujące artefakty występujące w zależnościach) i stwierdzono, że wszystko się zgadza. Obliczenia zostały udokumentowane w pliku maven-artifacts-dependencies.ipynb.

# Składowe spójne – NetworkX

Na początku sprawdzono, czy graf jest spójny. Funkcja zwróciła wartość False, więc stwierdzono, że graf posiada więcej niż jedną składową spójną. Stwierdzono, że graf posiada **21495 składowych spójnych**. Ten rząd wielkości nie został uznany za zbyt duży, ponieważ w pliku źródłowym już można bardzo łatwo znaleźć całkiem dużo artefaktów nie posiadających żadnych zależności od innych (czyli występujących jako niepołączone wierzchołki, o ile inny artefakt nie zależy od nich). Następnie wyznaczono największą składową spójną. Opisują ją następujące parametry:

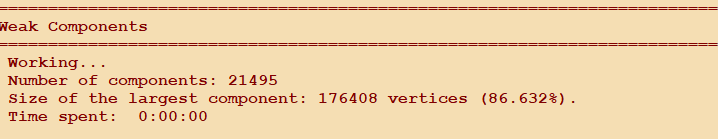
* **Rząd: 176408** (86,6% wszystkich wierzchołków w grafie)
* **Rozmiar: 1219622** (99,2% wszystkich krawędzi w grafie)

# Składowe spójne – Pajek

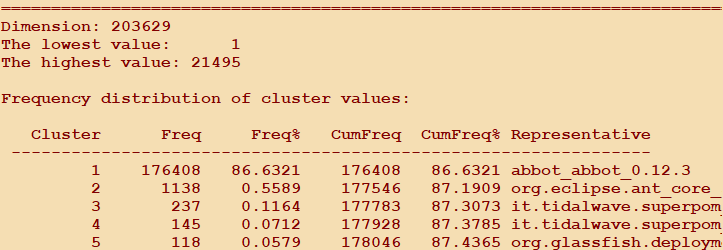
Dane w formacie programu Pajek pozyskano poprzez eksport przygotowanego wcześniej obiektu klasy Graph z NetworkX (przygotowanego do analizy dla poprzedniego punktu). Wszystkie węzły zostały opisane jako Vertices, a wszystkie krawędzie jako Edges (nieskierowane). Niestety przy próbie wczytania danych okazało się, że Pajek odpowiada błędem, ponieważ w pliku znajdowało się sporo spacji, tabulatorów i nowych linii w różnych, niepożądanych miejscach. Po naprawieniu pliku przy pomocy Notepad++, Pajek wczytał sieć bez problemu.

Po wczytaniu sieci, w celu Znalezienia składowych spójnych, wybrano z menu: Network -> Create Partition -> Components -> Weak

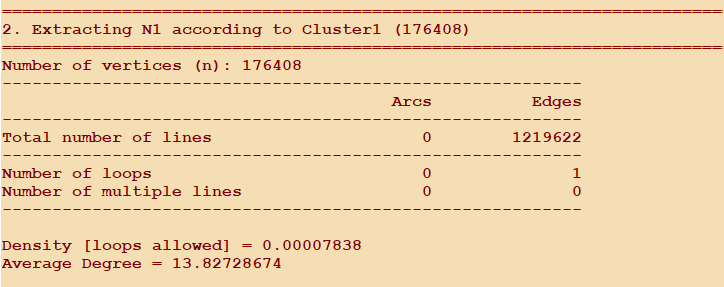
W oknie report otrzymano:



Następnie wybierając stworzoną partycję, wybrano z menu: Partition -> Canonical Partition -> with Decreasing Frequencies. Potem dla utworzonej partycji wybrano z menu: Partition -> Info. Oto początek wyniku:



Pierwsza pozycja na liście odpowiada największej składowej spójnej. Utworzono z niej cluster: Partition -> Make Cluster -> Vertices from selected Clusters. W okienku podano 1, aby wybrać tylko wierzchołki z pierwszej pozycji na powyższej liście. Nastepnie wybrano: Operations -> Network + Cluster -> Extract SubNetwork i w ten sposób utworzono sieć będącą największą składową spójną sieci, którą badamy. Dla największej składowej spójnej wybrano: Network -> Info -> General -> OK i otrzymano następującą odpowiedź:



Z powyższych danych można wyczytać, że:

W sieci jest **21495 składowych spójnych**. **Rząd** największej składowej spójnej to **176408**, a jej   
**rozmiar to 1219622**. Są to wyniki identyczne do tych otrzymanych w NetworkX.

# Porównanie NetworkX i Pajek

W Pajeku wczytywanie grafu trwało około 2 sekundy, wypisywanie rzędów stu największych składowych spójnych trwało około 1 sekundy. Wszystkie pozostałe operacje trwały poniżej   
1 sekundy.

W NetworkX Wczytywanie danych z pliku wraz z utworzeniem grafu (fragmenty kodu autora do zliczania powtarzających się krawędzi i wierzchołków zostały na czas sprawdzenia wykomentowane) trwa około 23 sekund (13 s. wczytywanie z pliku i 10 s. tworzenie grafu). Wyznaczanie ilości składowych spójnych oraz rozmiaru i rzędu największe składowej spójnej zajmuje około 18 sekund.

Z powyższych pomiarów widać bardzo wyraźnie, że Pajek jest dużo bardziej wydajny od biblioteki NetworkX.

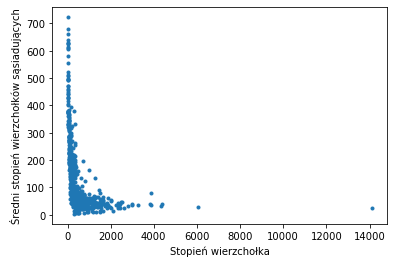
Jednak autor projektu dużo bardziej woli pracę z biblioteką NetworkX od programu Pajek, ponieważ przy wykonywaniu tego projektu, wydaje mu się ona dużo bardziej intuicyjna. Również dokumentacja dla NetworkX jest dużo przystępniejsza. Być może zdobycie większego doświadczenia w pracy z programem Pajek umożliwiłoby jeszcze lepsze zrozumienie programu, co pozwoliłoby na dużo przyjemniejsze wykorzystywanie niewątpliwie wyższej wydajności Pajeka.

# Asortatywność

Za pomocą biblioteki NetworkX przeanalizowano asortatywność największej składowej spójnej.

Na początku sprawdzono współczynnik asortatywności Pearsona dla wszystkich wierzchołków ze składowej za pomocą funkcji networkx.algorithms.assortativity.degree\_pearson\_correlation\_coefficient, która wywołuje funkcję scipy.stats.pearsonr liczącą korelację dla zbioru danych złożonego z każdej pary sąsiadujących wierzchołków. Ten współczynnik korelacji wynosi -0,060, co mogłoby świadczyć o

Najpierw obliczono średni stopień wierzchołków, z którymi łączą się wierzchołki poszczególnego stopnia. Wyniki przedstawiono na wykresie:



Współczynnik korelacji Pearsona dla tych danych wynosi