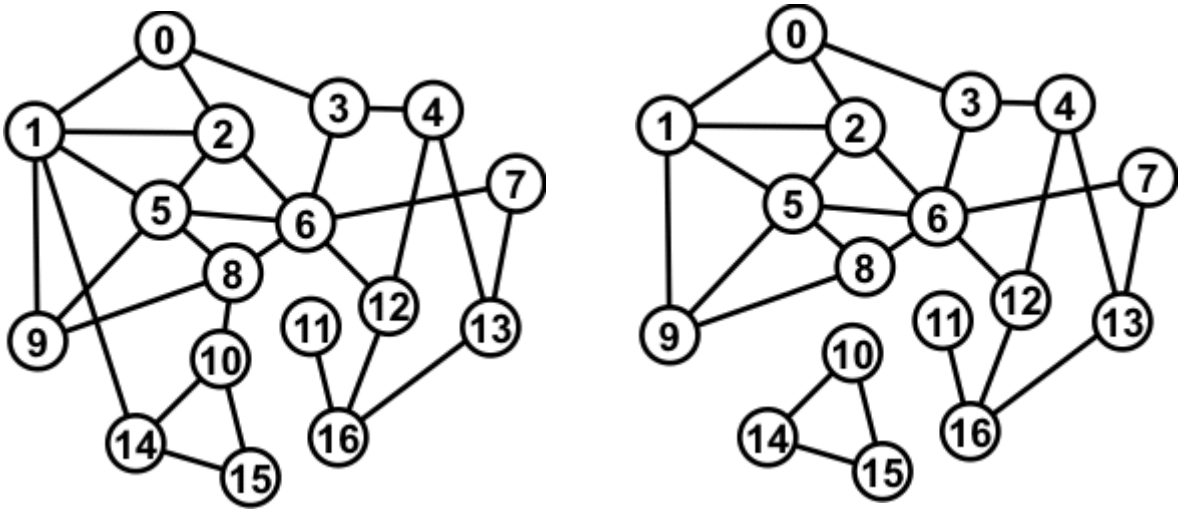
Projektowanie i Eksploatacja Sieci Komputerowych
Badanie spójności grafów i stopnia wypełnienia krawędziami

Jako podstawę do tego ćwiczenia wykorzystaj albo program z laboratorium 6. W klasie graf pozostaw jedynie funkcję dodającą wierzchołki (addVertex), countSteps, generateHistogram i wyświetlającą graf – pozostałe możesz usunąc. W trakcie zajęć spróbujemy zająć się problemem spójności w grafie.

Mówimy że graf jest spójny, jeśli z każdego wierzchołka można przejść do każdego innego wierzchołka. Przykłady grafu spójnego i niespójnego przedstawiają odpowiednio rysunki poniżej (po lewej graf spójny)



Spójność w grafie wyznaczamy, przechodząc po wszystkich wierzchołkach od wybranego i sprawdzając, czy utworzona podczas przechodzenia lista jest tak samo długa jak lista wszystkich wierzchołków.

Wobec powyższego wykonaj następujące zadania:

1. Dodaj funkcję, która obliczy maksymalną ilość krawędzi w grafie o aktualnej ilości wierzchołków: (n*(n-1))/2, gdzie n jest liczbą wierzchołków grafu
2. Dodaj funkcję, która:
    1. Jeśli graf ma już wszystkie możliwe krawędzie, nie zrobi nic lub wypisze komunikat o błędzie
    2. Jeśli nie, doda krawędź pomiędzy dwoma losowymi wierzchołkami już istniejącymi w grafie
3. Dodaj funkcję, która obliczy współczynnik wypełnienia grafu (podany w procentach), zdefiniowany następująco: maksymalna_liczba_krawędzi / faktyczna_ilość_krawędzi. Faktyczną ilość krawędzi obliczysz, sprawdzając listę wierzchołków (suma ilości sąsiadów podzielona przez 2 to ilość krawędzi w grafie)
4. Do badania spójności użyj następującej funkcji:

```
def spojnosc(self):
    visited = []
    for i in self.g:
        visited.append(False)
    visited.append(False)
    stack = []
    vVertex = 0
```

```
stack.append(1)
visited[1] = True
while len(stack) > 0:
    actual = stack.pop()
    vVertex = vVertex + 1
    for u in self.g[actual]:
        if visited[u] == True:
            continue

        visited[u] = True
        stack.append(u)

if vVertex == len(self.g):
    return True
else:
    return False
```

5. Korzystając z powyższej funkcji przygotuj następujący test:
    1. Dla grafu o podanej liczbie wierzchołków (dodaj je funkcją addVertex, tak żeby nie były połączone) sprawdź ile losowych krawędzi potrzeba, aby graf stał się spójny
    2. Do sprawozdania załącz tabelę z wynikami o następujących kolumnach: ilość wierzchołków, ilość krawędzi, współczynnik wypełnienia grafu
    3. Test wykonaj po 10 razy dla 10, 20 i 50 wierzchołków
6. Do klasy graf dodaj funkcję countIsolated, która policzy ilość wierzchołków izolowanych (do których nie jest dołączona żadna krawędź).
7. Zmodyfikuj działanie programu tak, aby jeśli są jakieś wierzchołki izolowane, funkcja dodająca losowe krawędzie dodawała je w pierwszej kolejności do wierzchołków izolowanych. W tym przypadku funkcja nie musi robić tego losowo, możesz iterować przez self.g (tablicę wierzchołków) szukając pierwszej pary wierzchołków izolowanych
8. Powtórz testy z punktu 5 dla tak zmodyfikowanego programu, wyniki załącz do sprawozdania.
9. Czy modyfikacje z punktu 7 znacząco poprawiają szybkość osiągnięcia spójności w grafie?