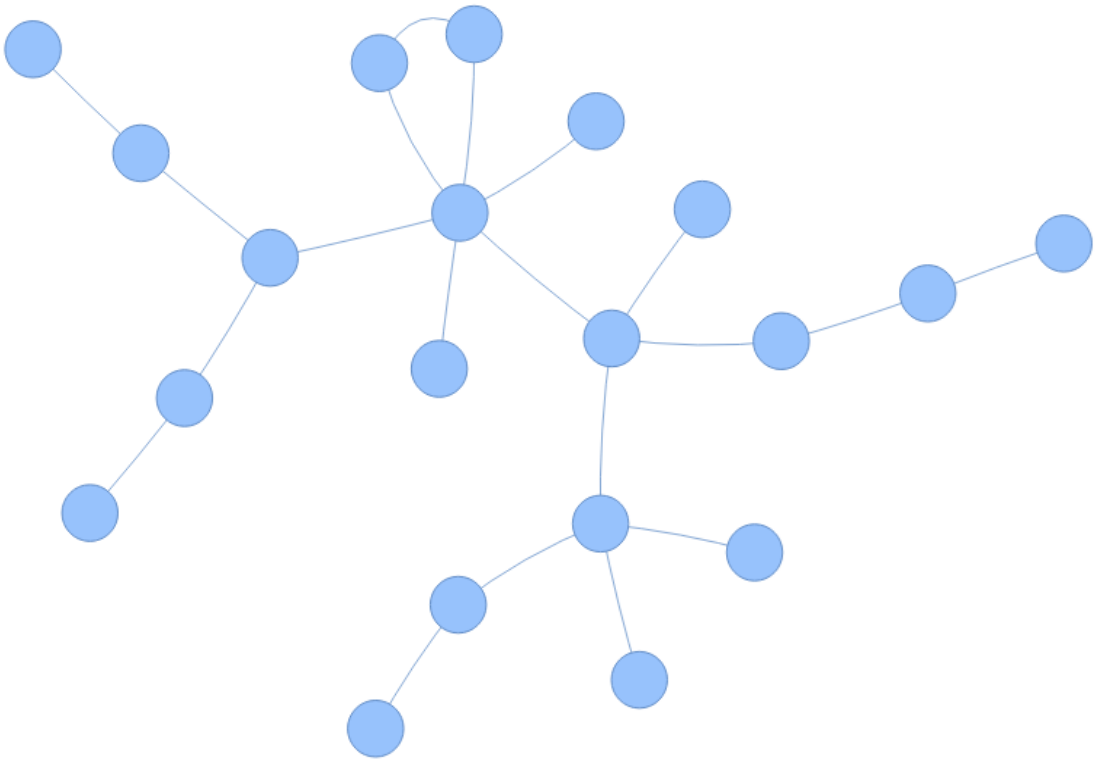
Lista 2

1. Stworzyłem graf, który zwizualizowałem przy użyciu pakietu pyvis.network



I wygenerowałem nastepującą,losową macierz natężeń z uwzględnieniem tego, że wierzchołek grafu nie może łączyć się z samym sobą:

```
my_N = [[0, 9, 6, 10, 5, 4, 5, 6, 4, 2, 4, 1, 8, 2, 8, 5, 8, 2, 4, 6],
        [6, 0, 10, 3, 4, 4, 7, 10, 7, 2, 7, 6, 2, 3, 4, 8, 2, 5, 2, 10],
        [1, 1, 0, 4, 8, 4, 6, 1, 7, 10, 5, 2, 7, 2, 5, 8, 9, 7, 9, 9],
        [1, 4, 1, 0, 10, 6, 3, 10, 2, 10, 3, 1, 3, 10, 1, 3, 2, 1, 6, 3],
        [2, 6, 8, 3, 0, 2, 5, 5, 3, 1, 7, 4, 4, 2, 4, 1, 8, 5, 5, 10],
        [3, 2, 1, 7, 10, 0, 1, 3, 6, 3, 6, 3, 4, 8, 1, 4, 8, 4, 4, 4],
        [10, 1, 2, 9, 3, 6, 0, 10, 2, 1, 10, 4, 9, 2, 6, 5, 3, 9, 7, 5],
        [6, 7, 7, 7, 9, 5, 4, 0, 6, 4, 10, 1, 7, 1, 3, 7, 4, 3, 6, 9],
        [9, 9, 1, 9, 4, 6, 9, 2, 0, 2, 10, 9, 1, 7, 2, 6, 1, 3, 4, 6],
        [3, 5, 5, 7, 10, 3, 9, 6, 10, 0, 7, 10, 10, 1, 8, 10, 3, 1, 2, 2],
        [4, 7, 8, 2, 10, 7, 1, 2, 7, 7, 0, 6, 6, 3, 8, 5, 5, 3, 5, 6],
        [1, 7, 6, 3, 6, 1, 2, 2, 6, 2, 4, 0, 1, 6, 5, 4, 8, 10, 10, 10],
        [4, 2, 2, 1, 4, 4, 7, 1, 10, 8, 10, 9, 0, 3, 3, 8, 4, 8, 3, 8],
        [8, 2, 6, 1, 4, 9, 3, 4, 7, 10, 1, 6, 1, 0, 10, 10, 9, 2, 3, 8],
        [1, 8, 9, 6, 7, 6, 8, 4, 6, 2, 5, 3, 4, 5, 0, 6, 4, 3, 5, 4],
        [1, 5, 10, 4, 3, 3, 2, 5, 2, 2, 7, 3, 8, 3, 8, 0, 5, 7, 10, 2],
        [4, 9, 6, 6, 1, 6, 4, 6, 6, 8, 5, 5, 5, 3, 7, 5, 0, 2, 9, 1],
        [3, 7, 6, 3, 5, 5, 5, 8, 6, 9, 8, 9, 4, 10, 2, 8, 1, 0, 6, 5],
        [4, 6, 3, 6, 3, 9, 1, 6, 5, 7, 7, 5, 4, 10, 6, 9, 6, 8, 0, 4],
        [10, 2, 1, 1, 6, 4, 7, 2, 4, 5, 1, 7, 9, 8, 7, 9, 10, 10, 9, 0]]
```

Przyjąłem następujące definicje funkcji:

- a(n) (funkcja przepływu) – to wszystkie pakiety, które mogą przepłynąć przez daną krawędź. Pakiety do dwóch konkretnych wierzchołków przesyłane są najkrótszą ścieżką, znalezioną przy użyciu nx.shortest_path. Po jej ustaleniu, do wartości funkcji a każdego wierzchołka na jej trasie dodaję odpowiadającą mu wartość z macierzy natężeń.

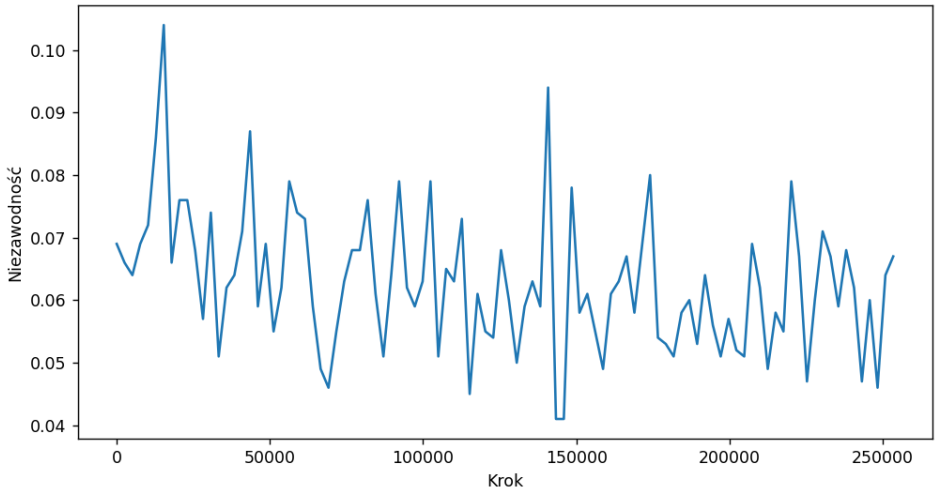
- c(n) (funkcja przepustowości) – ustalam ją na podstawie wartości funkcji a(n) dla danego wierzchołka. Jest to dziesięciokrotność wartości funkcji a(n), do której dodaję określoną, stałą wartość SIZE, reprezentującą standardową wielkość pakietu. W razie potrzeby, można przyjąć odmienną wartość SIZE, jednakże ja ustawiłem ją na 256. Taka definicja funkcji c(n) zapewnia, że będzie zachodziło c(n) > a(n).

- T (funkcja niezawodności) – w każdym z testów jest obliczana wartość funkcji T w celu ustalenia niezawodności sieci po wykonaniu na niej jakiejś operacji. T = 1/G * SUM_e(a(e)/(c(e)/m - a(e)))
  Sumuję po wszystkich krawędziach grafu funkcję a(e)/(c(e)/m – a(e)). G to suma wszystkich elementów z macierzy natężeń N. M to średnia wielkość pakietu w bitach.
  Ponadto, do badania niezawodności, używana jest wartośc p określająca prawdopodobieństwo nieprzerwania ścieżki pomiędzy dwoma wierzchołkami.
  Jeśli podczas testu sieć nie została przerwana oraz zachowane jest T < T_MAX, to powyższa funkcja T jest miarą jej niezawodności.
  Testy niezawodności przeprowadzam dla 100 iteracji, gdzie w każdej kilkukrotnie sprawdzam, czy jakaś ścieżka zostanie przerwana. Jeśli tak, usuwam ją z grafu, przy pomocy nx.is_connected upewniam się, czy graf nadal jest spójny, obliczam na nowo wartość funkcji przepływu oraz funkcję T. Jeśli T spełnia założenia (T < T_MAX), to inkrementuję liczbę sukcesów. Na końcu wykonywania funkcji zwracam wartośc sukcesów podzieloną przez całościową liczbę testów.

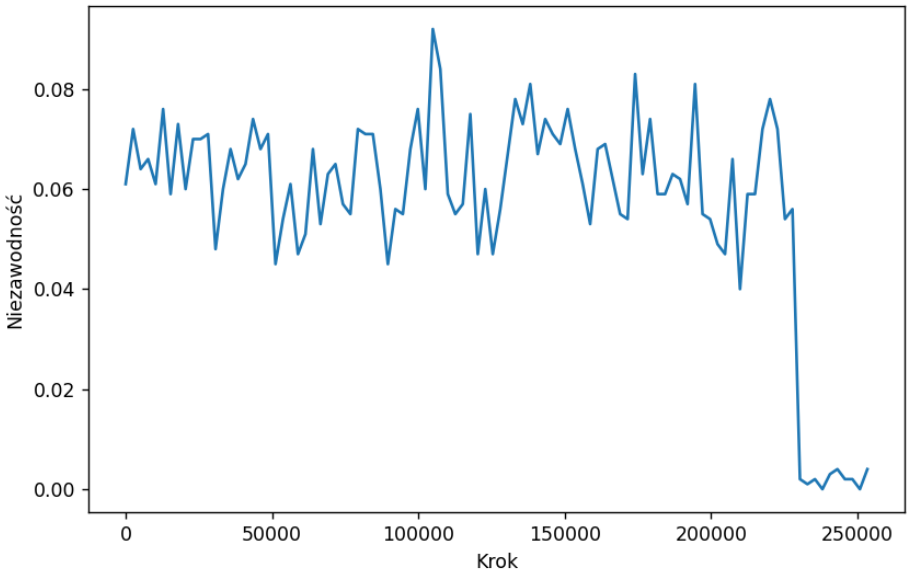
Test 1 – zwiększanie wartości w macierzy natężeń N

W tym teście zwiększam wartości w macieży N o konkretny krok, obliczany na podstawie wartości SIZE (SIZE * 10). W każdej iteracji testu losowo wybieram dwa wierzchołki I oraz j, które są od siebie różne. Następnie zwiększam wartość w macierzy N[i][j] o podany krok, jak I wartości funkcji a na całej trasie między tymi dwoma wierzchołkami. Następnie obliczam niezawodność.

Dla m = 5 i p = 95:



Po zwiększeniu m do 6, można zauważyć nagły krach w niezawodności. Im większe m, tym szybciej on następuje

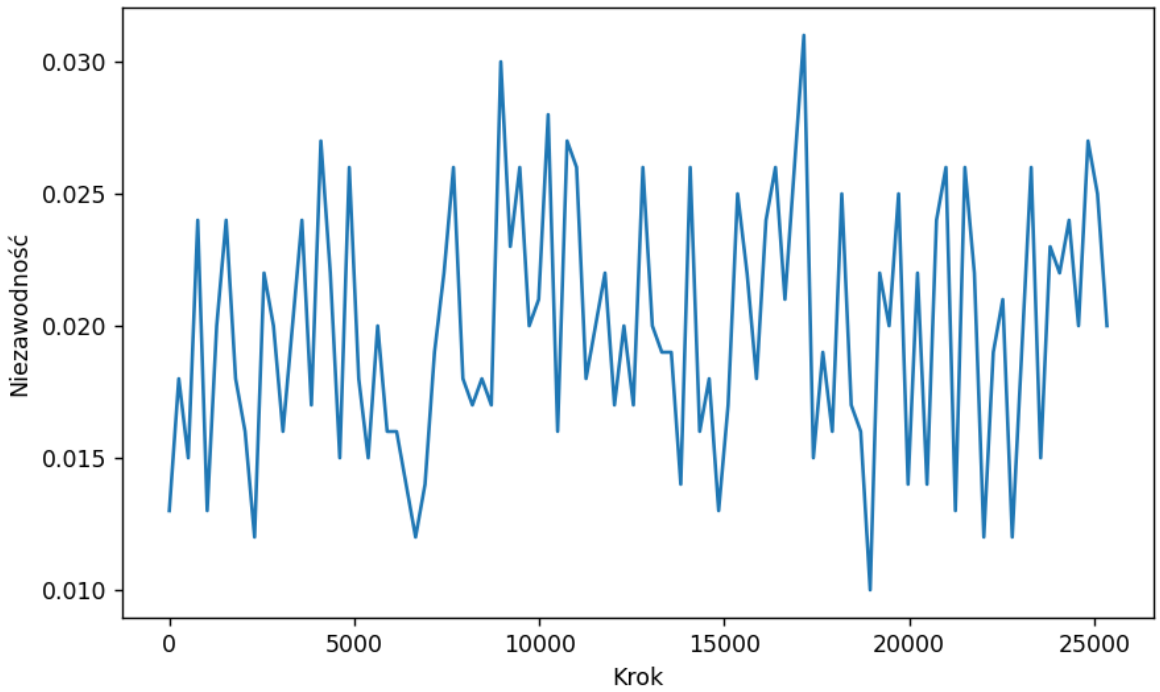


Obserwacje:
1. Wzrost T_MAX pozwala na dodanie większej liczby pakietów
2. Wzrost p również
3. Wzrost m wykazuje tendencję odwrotną

Wnioski: wraz ze wzrostem wartości w macierzy natężeń N, spada niezawodność całej sieci.

Test 2 – zwiększanie wartości funkcji przepustowości

W tym teście zwiększam wartości funkcji ‘c’ dla każdego z wierzchołków o wartość SIZE, a następnie obliczam niezawodność.

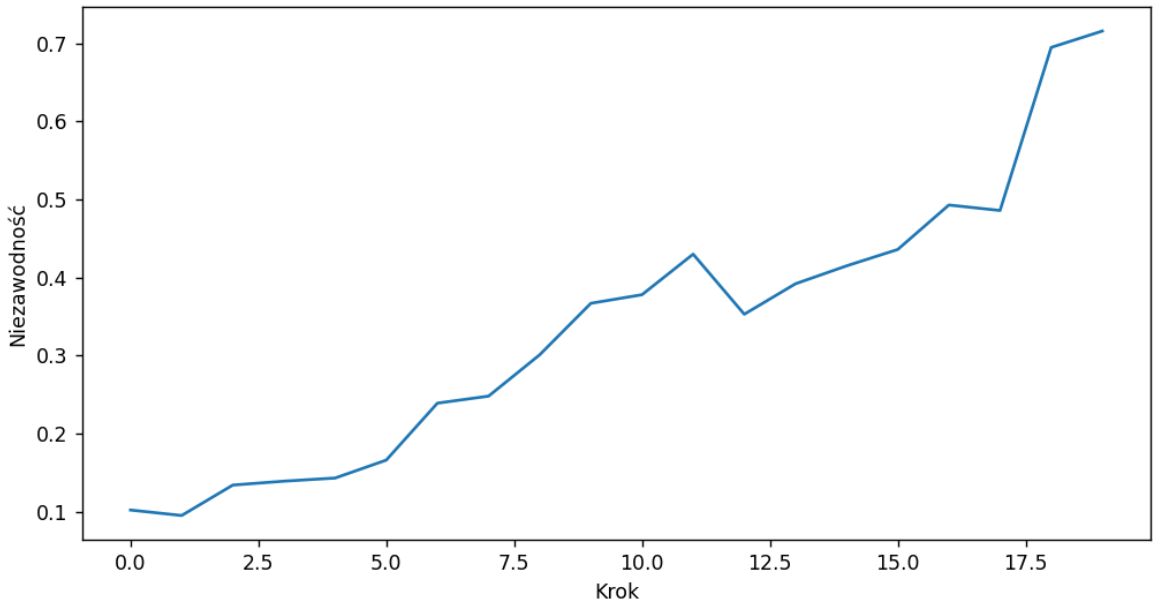


Przeprowadzone testy wykazały poprawę stabilności sieci, jednakże nie aż tak, jak dodawanie nowych krawędzi (w teście poniżej). Mniejsze wartości m mają zdecydowanie wpływ na widoczność pozytywnego trendu.

Wnioski: zwiększanie przepustowości zwiększa niezawodność całej sieci

Test 3 – dodawanie nowych krawędzi

W tym teście losowo wybieram dwa wierzchołki, które nie mają połączenia. Następnie dodaję je oraz dostosowuję odpowiednio wartości funkcji przepustowości (średnia wartość sieci początkowej). Obliczam ponownie funkcję a(n), a na końcu niezawodność.



Wnioski: dodawanie nowych krawędzi wpływa bardzo pozytywnie na niezawodność.

Wnioski ogólne: wyniki przeprowadzonych eksperymentów pokrywają się z ogólną intuicją na temat działania sieci.