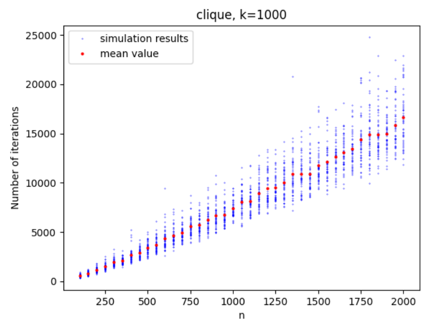
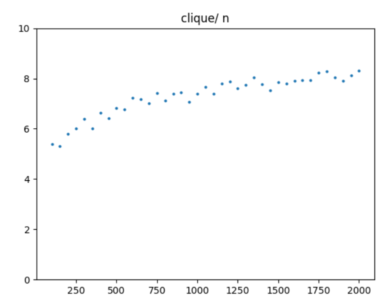
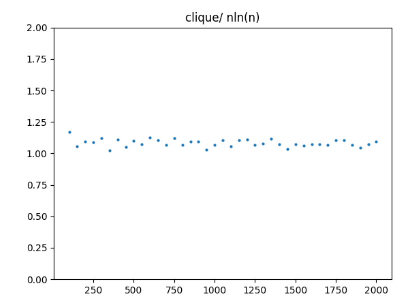
1 Clique

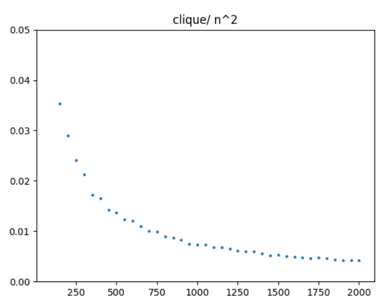




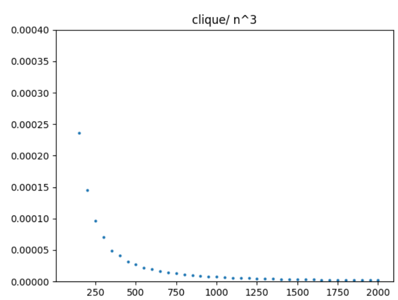
(a) $\frac{\text{clique}(n)}{n}$



(b) $\frac{\text{clique}(n)}{nln(n)}$



(c) $\frac{\text{clique}(n)}{n^2}$



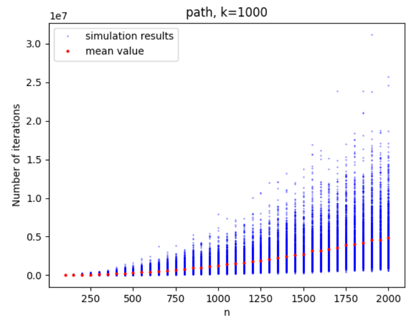
(d) $\frac{\text{clique}(n)}{n^3}$

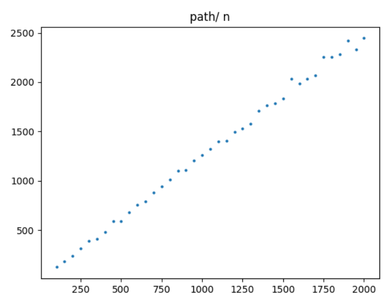
Rysunek 1: Wykresy pomagające wyznaczyć asymptotykę funkcji.

Hipoteza:

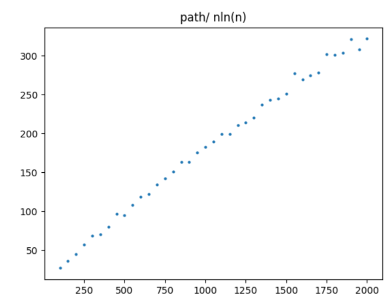
$$\text{Clique}(n) = O(nln(n))$$

2 Path

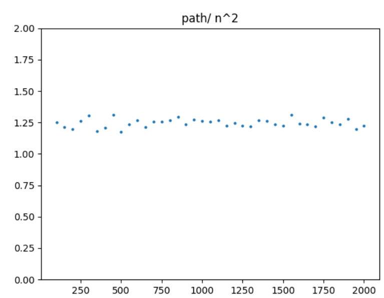




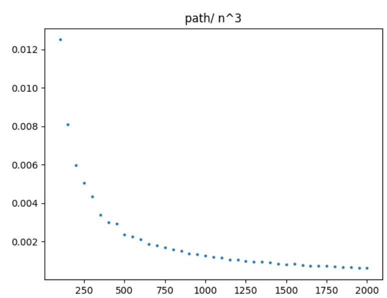
(a) $\frac{\text{path}(n)}{n}$



(b) $\frac{\text{path}(n)}{nln(n)}$



(c) $\frac{\text{path}(n)}{n^2}$



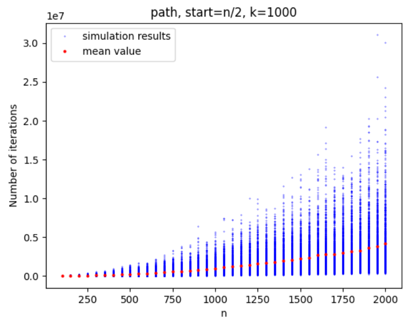
(d) $\frac{\text{path}(n)}{n^3}$

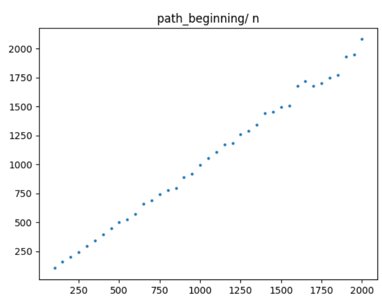
Rysunek 2: Wykresy pomagające wyznaczyć asymptotykę funkcji.

Hipoteza:

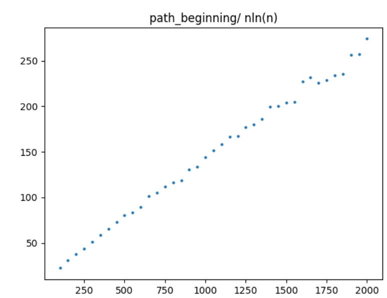
$$\text{path}(n) = O(n^2)$$

3 Path, start in the beginning node

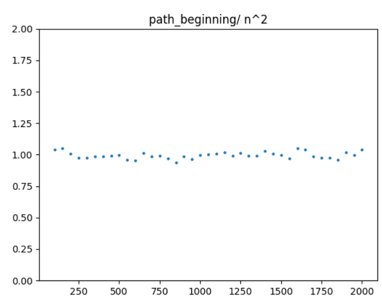




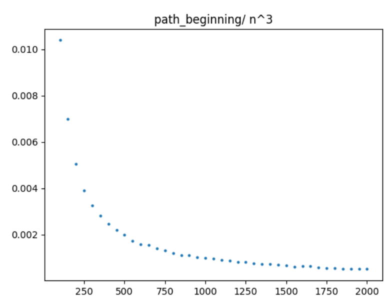
(a) $\frac{\text{path}_0(n)}{n}$



(b) $\frac{\text{path}_0(n)}{n ln(n)}$



(c) $\frac{\text{path}_0(n)}{n^2}$



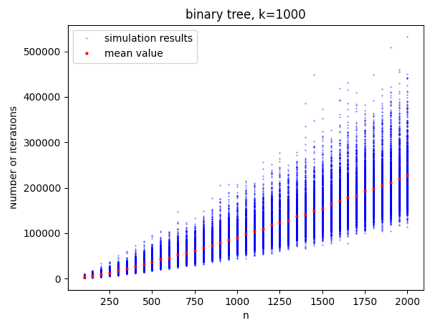
(d) $\frac{\text{path}_0(n)}{n^3}$

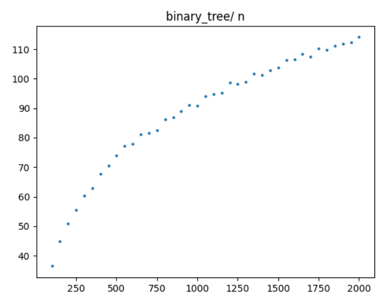
Rysunek 3: Wykresy pomagające wyznaczyć asymptotykę funkcji.

Hipoteza:

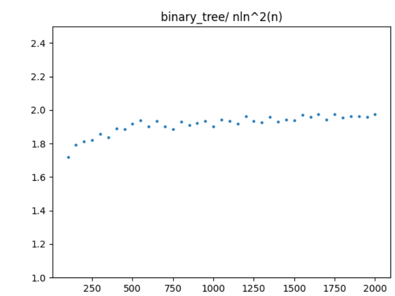
$$\text{path}_0(n) = O(n^2)$$

4 Binary Tree

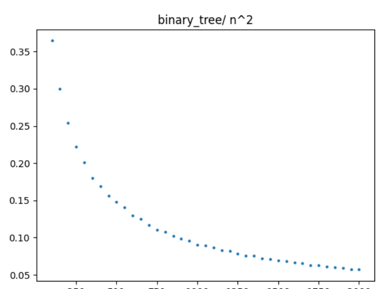




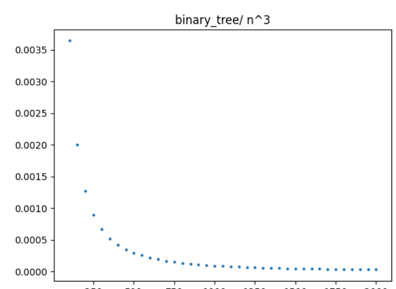
(a) $\frac{\text{binarytree}(n)}{n}$



(b) $\frac{\text{binarytree}(n)}{nln^2(n)}$



(c) $\frac{\text{binarytree}(n)}{n^2}$



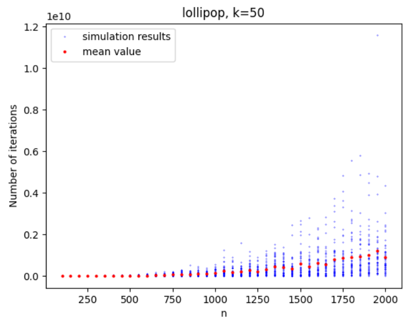
(d) $\frac{\text{binarytree}(n)}{n^3}$

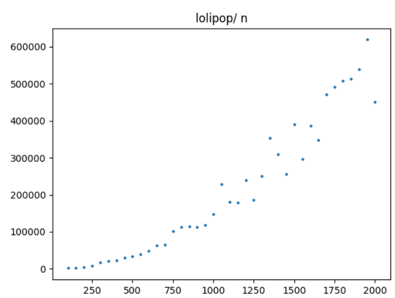
Rysunek 4: Wykresy pomagające wyznaczyć asymptotykę funkcji.

Hipoteza:

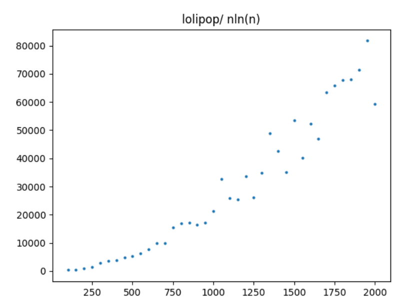
$$\text{binarytree}(n) = O(nln^2(n))$$

5 Lollipop

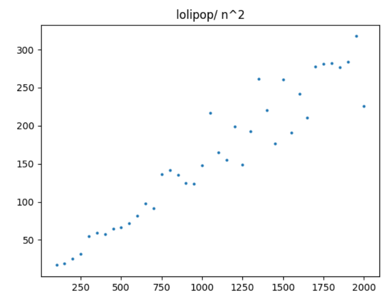




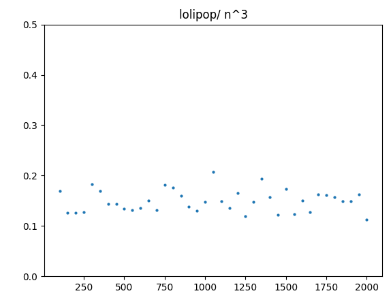
(a) $\frac{\text{lollipop}(n)}{n}$



(b) $\frac{\text{lollipop}(n)}{nln(n)}$



(c) $\frac{\text{lollipop}(n)}{n^2}$



(d) $\frac{\text{lollipop}(n)}{n^3}$

Rysunek 5: Wykresy pomagające wyznaczyć asymptotykę funkcji.

Hipoteza:

$$\text{lollipop}(n) = O(n^3)$$

6 Wnioski

Z ciekawych rzeczy, zauważyć możemy, że przy symulacji ścieżki, niezależnie od miejsca startu (początek / środek) mamy i tak tę samą złożoność. Poza tym możemy zaobserwować bardzo dużą wariancję, przykładowo w jednej symulacji lollipopa przy $n = 1950$ wykonane zostało aż $11,578,127,039$ iteracji, podczas gdy średnia wynosiła około 200 mln.

symulacja

Jeśli chodzi o rzeczy ściśle powiązane z częścią programistyczną tego zadania, to przeprowadziłem eksperyment porównujący wydajność symulacji przy korzystaniu z listy sąsiedztwa w grafie, a funkcją losową, która zwraca następny wierzchołek do odwiedzenia (bez tworzenia żadnych struktur w pamięci, poza tablicą odwiedzonych wierzchołków). Okazuje się, że drugie rozwiązanie jest znacznie wydajniejsze w grafach, w których listy sąsiedztwa są na tyle długie, że nie mieszczą się w pamięci cache. Tutaj były to klika oraz lizak. Dla kliki wydajność wzrosła około ośmiokrotnie, a dla lizaka dwukrotnie. Dla reszty grafów, gdzie każdy wierzchołek ma niewielu sąsiadów, wydajniejsze było stworzenie grafu i losowanie wierzchołka z listy sąsiedztwa.