Kombinatoryka Analityczna 2

Bohdan Tkachenko 256630

November 17, 2023

1 Zadanie48

W tym sprawozdaniu przedstawiono analize funkcji

$$s(n) = \sum_{k=1}^{n} sd(k),$$

, która jest zdefiniowana jako suma cyfr binarnych przedstawienia wszystkich liczb od 1 do n. Celem analizy jest zbadanie asymptotycznego zachowania tej funkcji, a także próba znalezienia i porównania jej przybliżeń za pomoca prostszych funkcji matematycznych.

1.1 Metodologia

Analiza została przeprowadzona w kilku krokach:

1. Zdefiniowanie i obliczenie wartości funkcji $s(n)$ dla n od 1 do 1024.
2. Porównanie funkcji $s(n)$ z funkcjami n, oraz $n \log n$, w celu zbadania jej wzrostu.
3. Zastosowanie metody najmniejszych kwadratów do znalezienia optymalnych współczynników skalujacych dla porównywanych funkcji.
4. Analiza różnic miedzy funkcja $s(n)$ a przybliżajacymi ja funkcjami.

1.2 Proces Znalezienia Współczynników Skalujacych

W celu znalezienia optymalnych współczynników skalujacych dla funkcji porównawczych zastosowano metode najmniejszych kwadratów. Jest to standardowa technika w analizie regresji, służaca do minimalizacji sumy kwadratów różnic miedzy obserwowanymi wartościami a wartościami przewidywanymi przez model.

1.2.1 Metoda Najmniejszych Kwadratów

Metoda najmniejszych kwadratów polega na znalezieniu takich wartości parametrów modelu, które minimalizuja sume kwadratów odchyleń wartości przewidywanych od wartości rzeczywistych. W naszym przypadku, dla każdej z funkcji porównawczych (liniowej i $n \log n$), szukamy współczynnika c, który minimalizuje wyrażenie:

$$S(c) = \sum_{i=1}^{1024} \left(s(n_i) - c \cdot f(n_i)\right)^2 ,$$

gdzie $s(n_i)$ to wartość funkcji $s(n)$ dla danego n_i, a $f(n_i)$ to wartość odpowiedniej funkcji porównawczej (liniowej lub $n \log n$) dla danego n_i.

1.2.2 Implementacja w Pythonie

W naszym przypadku użyto funkcji `curve_fit` z biblioteki SciPy w jezyku Python, która implementuje metode najmniejszych kwadratów. Funkcja ta automatycznie dostosowuje parametry modelu (w naszym przypadku, współczynnik skalujacy c), aby znaleźć najlepsze dopasowanie do danych. Dla funkcji liniowej model ma postać $c \cdot n$, a dla funkcji $n \log n$ - $c \cdot n \log n$.

Wynikiem działania tej funkcji sa optymalne wartości współczynników c, które nastepnie wykorzystano do porównania z funkcja $s(n)$ i oceny dopasowania.

1.3 Definicja Funkcji $s(n)$

Funkcja $s(n)$ jest zdefiniowana jako suma cyfr binarnych wszystkich liczb od 1 do n. Może być wyrażona jako:

$$s(n) = \sum_{k=1}^{n} sd(k),$$

gdzie $sd(k)$ oznacza sume cyfr binarnych liczby k.

1.3.1 Porównanie z Funkcjami n i $n \log n$

Do porównania z funkcja $s(n)$ wybrano dwie funkcje: liniowa n oraz $n \log n$. W celu uzyskania najlepszego dopasowania, dla każdej z nich wyznaczono współczynnik skalujacy metoda najmniejszych kwadratów.

1.4 Wyniki

Wyniki analizy prezentuja sie nastepujaco:

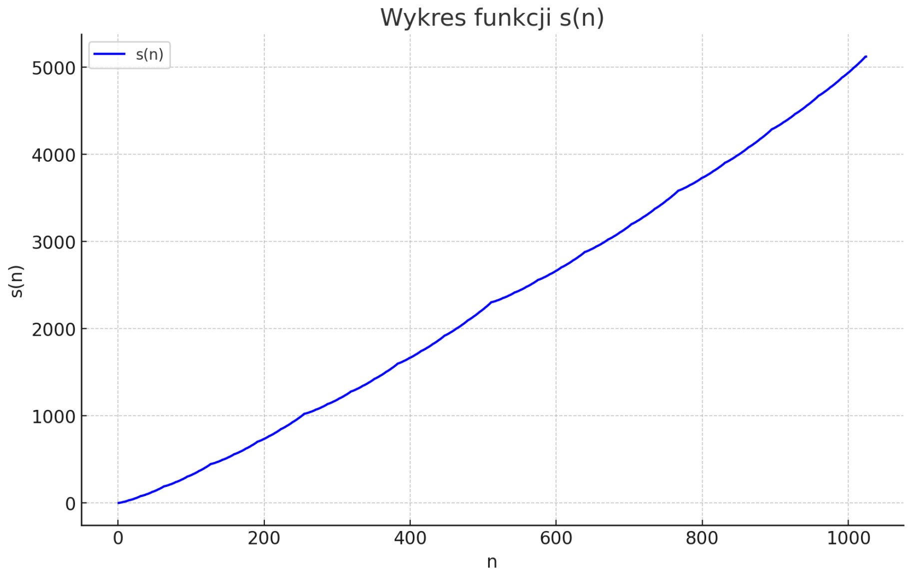

Figure 1: Wykres funkcji $s(n)$

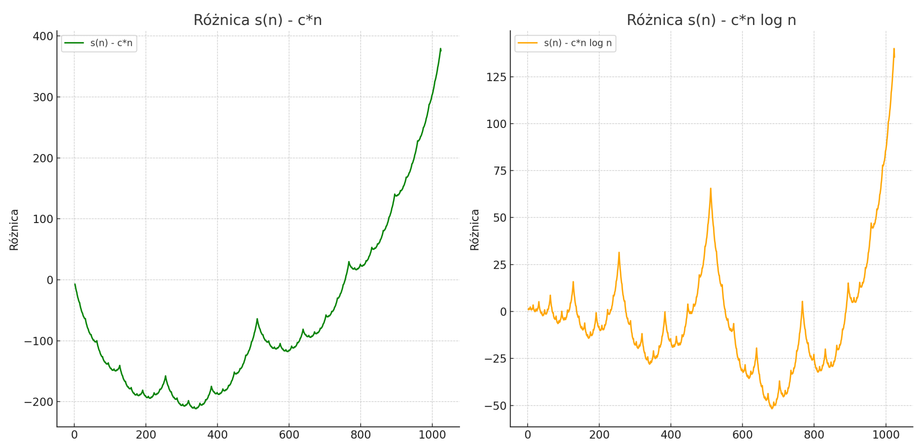

Figure 2: Porównanie funkcji $s(n)$, n, i $n \log n$

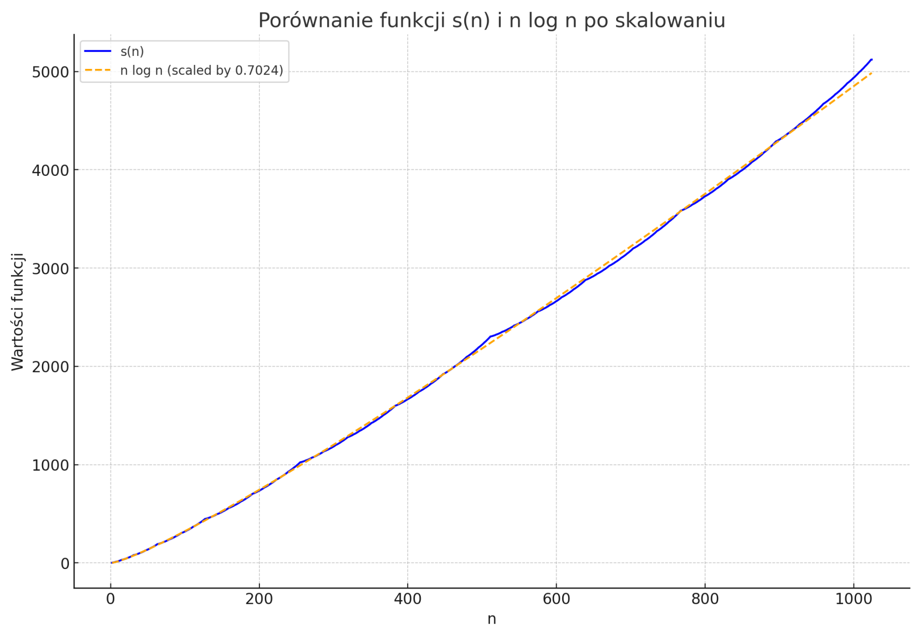

Figure 3: Porównanie funkcji $s(n)$, $n \log n$

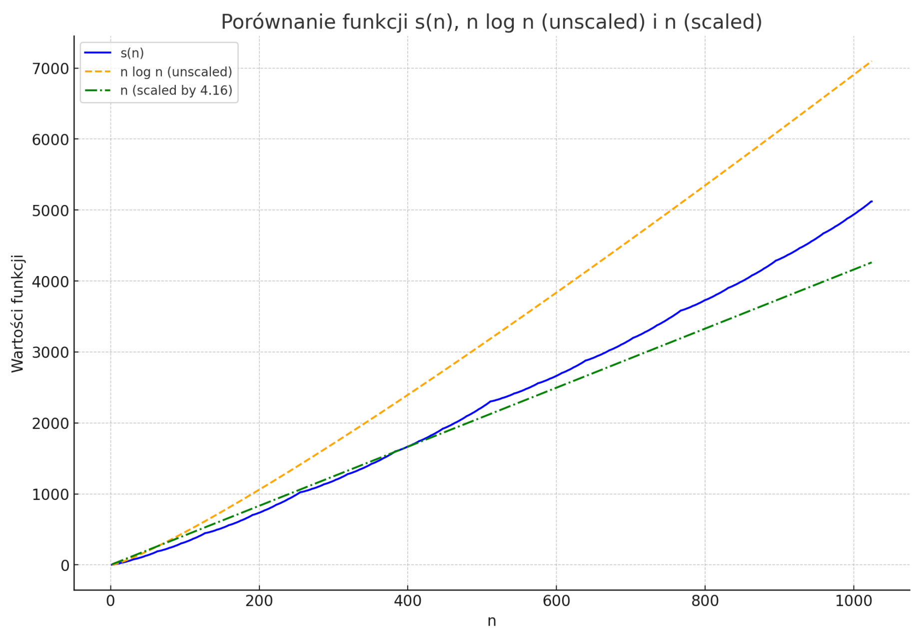

Figure 4: Porównanie 3 funkcji

1.5 Dopasowanie Funkcji

Znaleziono, że funkcja $n \log n$ z optymalnym współczynnikiem skalujacym $c \approx 0.7024$ daje najlepsze dopasowanie do funkcji $s(n)$.

1.6 Podsumowanie

Analiza wykazała, że funkcja $s(n)$ rośnie szybciej niż liniowo, ale wolniej niż kwadratowo. Najlepsze dopasowanie zapewnia funkcja $n \log n$ z odpowiednio dobranym współczynnikiem skalujacym. Wynik ten jest zgodny z intuicja, że liczba cyfr binarnych rośnie logarytmicznie w stosunku do rozmiaru liczby.

2 Zadanie 49

W niniejszej sekcji przedstawiamy szczegółowe przekształcenia matematyczne na podstawie zadanych wzorów. Naszym celem jest wyprowadzenie i analiza wzoru na np_n w kontekście funkcji generujacej $P(x)$.

2.1 Definicja i Właściwości Funkcji $P(x)$

Rozpoczynamy od zdefiniowania funkcji generujacej:

$$P(x) = \prod_{k=1}^{\infty} \frac{1}{1 - x^k}$$

Współczynnik p_n w rozwinieciu tej funkcji w szereg potegowy jest zdefiniowany jako:

$$p_n = [x^n]P(x)$$

co oznacza, że p_n to współczynnik przy x^n w rozwinieciu $P(x)$.

2.2 Logarytmowanie i Różniczkowanie $P(x)$

Nastepnie logarytmujemy funkcje $P(x)$ i różniczkujemy wynik wzgledem x:

$$\ln P(x) = \sum_{k=1}^{\infty} \ln \left(\frac{1}{1 - x^k} \right)$$

$$\frac{P'(x)}{P(x)} = \sum_{k=1}^{\infty} \frac{kx^{k-1}}{1 - x^k}$$

2.3 Wyprowadzenie Wzoru na np_n

Mnożymy obie strony ostatniego równania przez x, otrzymujac:

$$x\frac{P'(x)}{P(x)} = \sum_{k=1}^{\infty} \frac{kx^k}{1 - x^k}$$

Wykorzystujac definicje $p_n = [x^n]P(x)$, możemy wyprowadzić wzór na np_n:

$$np_n = n[x^n]P(x) = [x^n]xP'(x)$$

Oznacza to, że np_n jest współczynnikiem przy x^n w szeregu $xP'(x)$.

2.4 Zwiazek z Suma Dzielników $\sigma(n)$

Rozważmy wyrażenie $\frac{kx^k}{1-x^k}$, które można rozwinać jako szereg geometryczny:

$$\frac{kx^k}{1-x^k} = kx^k + kx^{2k} + kx^{3k} + \cdots$$

Każdy wyraz tej serii reprezentuje wkład od poteg x bedacych wielokrotnościami k. Sumujac to wyrażenie dla wszystkich k, otrzymujemy wkłady dla wszystkich dzielników każdej liczby n, co prowadzi do sumy dzielników $\sigma(n)$.

2.5 Końcowe Przekształcenia i Wzór

Ostatecznie, przekształcenie do postaci np_n wyglada nastepujaco:

$$np_n = [x^n]P(x)\sum_{k=1}^{\infty}\frac{kx^k}{1-x^k} = \sum_{j=1}^{n}\sigma(j)p_{n-j}$$

gdzie $\sigma(n)$ to suma dzielników liczby n, a p_{n-j} to współczynnik przy x^{n-j} w szeregu $P(x)$.

2.6 Złożoność Obliczeniowa Programu

2.6.1 Obliczanie Sumy Dzielników (σ)

Pierwszym krokiem algorytmu jest obliczenie sumy dzielników (σ) dla każdej liczby od 1 do n. Funkcja ta jest realizowana poprzez iteracje po wszystkich liczbach od 1 do n i dla każdej z nich sumowanie jej dzielników. Złożoność obliczeniowa tej cześci algorytmu wynosi $O(n \log n)$, co wynika z faktu, że liczba operacji potrzebna do obliczenia sumy dzielników dla każdej liczby jest proporcjonalna do liczby jej dzielników, a średnia liczba dzielników liczby rośnie logarytmicznie w stosunku do jej wartości.

2.6.2 Obliczanie Wartości p_n

Główna cześć algorytmu obejmuje obliczanie wartości p_n zgodnie z danym wzorem rekurencyjnym. Dla każdej liczby n od 1 do 100, algorytm wykonuje sumowanie produktów wartości $\sigma(j)$ i p_{n-j} dla każdego j od 1 do n. Ponieważ każde takie sumowanie wymaga wykonania n operacji, a operacje te sa wykonywane dla każdego n od 1 do 100, złożoność obliczeniowa tej cześci algorytmu wynosi $O(n^2)$.

2.6.3 Złożoność Całkowita

Biorac pod uwage obie wyżej opisane cześci algorytmu, złożoność całkowita programu jest dominowana przez złożoność obliczeniowa obliczania wartości p_n, czyli $O(n^2)$.

2.7 Wyniki

W tej sekcji prezentujemy wyniki obliczeń wartości p_n dla $n = 1, \ldots, 10$

n	p_n
1	1
2	2
3	3
4	5
5	7
6	11
7	15
8	22
9	30
10	42

Table 1: Wyniki obliczeń wartości p_n