Maximum load w modelu kul i urn oraz insertion sort

Ewa Kasprzak

31 grudnia 2023

1. Wstęp

W niniejszym sprawozdaniu skoncentrujemy się na dwóch istotnych zagadnieniach w teorii algorytmów: maximum load w modelu kul i urn oraz analizie złożoności algorytmu Insertion Sort. Obejmują one obszary, w których ilość danych oraz skomplikowanie operacji mają kluczowe znaczenie, zarówno w teorii, jak i praktyce.

Maximum load to pojęcie szczególnie istotne w analizie efektywności rozproszonych systemów, gdzie kulki (reprezentujące zadania, dane czy obciążenie) są umieszczane w urnach (symbolizujących zasoby, procesory czy inne jednostki). Naszym celem jest zrozumienie, jak równomiernie rozłożone jest obciążenie pomiędzy urnami oraz jakie są skutki jego maksymalnego nagromadzenia w jednej z nich. W toku analizy będziemy badać, jakie strategie umieszczania kul w urnach prowadzą do optymalnego rozłożenia obciążenia, minimalizując jednocześnie ryzyko maksymalnego obciążenia jednej urny.

Algorytm Insertion Sort to jedna z fundamentalnych metod sortowania, która operuje na zbiorze danych poprzez sekwencyjne wstawianie kolejnych elementów na właściwe pozycje. W kontekście analizy algorytmicznej zastanowimy się nad złożonością tego procesu, skupiając się na liczbie porównań oraz przestawień, co pozwoli nam ocenić efektywność sortowania. Przyjrzymy się, jak algorytm radzi sobie z różnymi rodzajami danych wejściowych oraz jakie czynniki wpływają na jego wydajność.

Zrozumienie tych dwóch zagadnień pozwoli na lepsze projektowanie algorytmów i optymalne zarządzanie zasobami w systemach informatycznych.

2. Maximum Load

2.1 Opis modelu

W modelu tym, m kul jest wrzucane kolejno do n ponumerowanych urn. Każda kula jest wrzucana niezależnie z jednakowym prawdopodobieństwem równym $\frac{1}{n}$ do jednej z urn. Wrzucenie kuli do urny możemy utożsamiać z losową funkcją f ze zbioru $\{1, \ldots, m\}$ do zbioru $\{1, \ldots, n\}$. Formalnie, przestrzeń zdarzeń elementarnych jest wówczas zbiorem $\Omega_{n,m} = \{1, \ldots, n\}^{1,\ldots,m}$.

Celem tego zadania jest eksperymentalne wyznaczenie następujących wielkości:

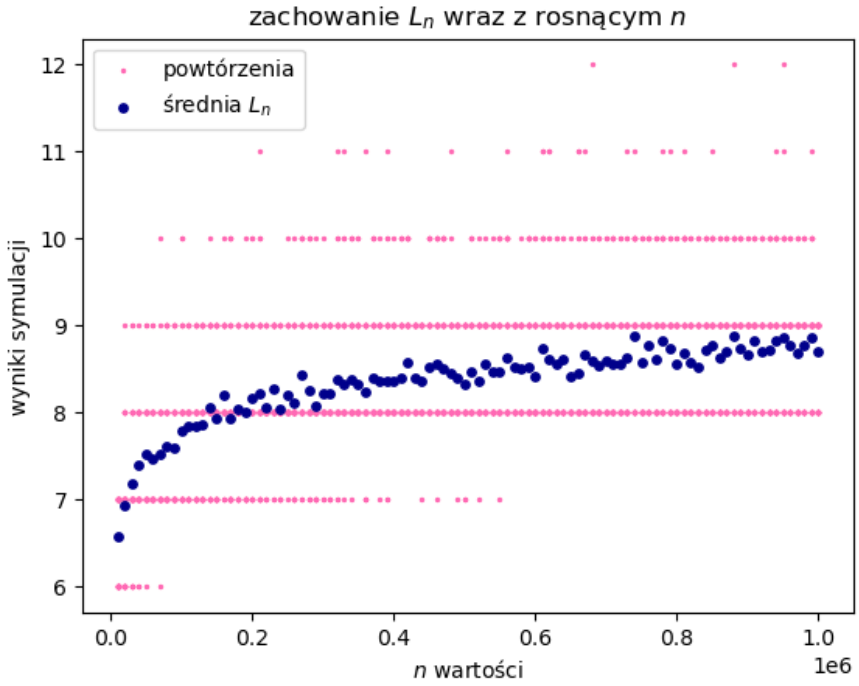
1. $L_n^{(1)}$ - maksymalne zapełnienie pojedynczej urny po wrzuceniu n kul, gdy dla każdej kuli niezależnie i jednostajnie losujemy jedną urnę, do której umieszczamy kulkę.
2. $L_n^{(2)}$ - maksymalne zapełnienie pojedynczej urny po wrzuceniu n kul, gdy dla każdej kuli niezależnie i jednostajnie losujemy urnę z powtórzeniami, a kulkę umieszczamy w najmniej zapełnionej z wybranych urn.

2.2 Opis symulacji

Wykonano dla każdej wartości n z zakresu $\{10000, 20000, \ldots, 1000000\}$ 50 niezależnych powtórzeń eksperymentu wrzucania kul do urn. Każde pojedyncze powtórzenie polegało na wrzuceniu n kul i ustaleniu maksymalnej liczby kul w urnie.

Do generacji liczb pseudolosowych został użyty generator SecureRandom z języka Java, co zapewniło dobre własności statystyczne. W celu prezentacji wyników użyto narzędzia NumPy z języka Python. Na wykresach przedstawiono wyniki poszczególnych powtórzeń, zaznaczając punkty danych dla każdego n, oraz dodano średnie wartości, co umożliwiło łatwą ocenę koncentracji wyników wokół wartości średniej.

2.3 Wyniki symulacji i asymptotyka wartości oczekiwanych zmiennych losowych



Rysunek 1: Wyniki eksperymentów dla wartości $L_n^{(1)}$.

Powyższy wykres ilustruje zmiany wartości $L_n^{(1)}$ w miarę wzrostu parametru n podczas przeprowadzanej symulacji. Wzrost wartości parametru n nie wpływa na znaczące odchylenia od wartości oczekiwanej, co wskazuje na fakt, że współczynnik koncentracji, tj. $\frac{\sigma(X)}{E(X)}$, jest niski.

Aby wskazać asymptotyczne tempo wzrostu powyższej funkcji, udowodnimy hipotezę, że $\forall n \in \mathbb{N}$, $\forall c \in \mathbb{R}$

$$L_n^{(1)} \leq \frac{c \ln(n)}{\ln(\ln(n))}$$

z prawdopodobieństwem $1 - \frac{1}{n}$.

Zmienna losowa X oznaczająca liczbę kul w urnie to zmienna losowa o rozkładzie Poissona. Prawdopodobieństwo, że dana urna ma dokładnie r kul jest zatem równe, mając m liczba kul, n liczba urn:

$$\binom{m}{r}\left(\frac{1}{n}\right)^r\left(1-\frac{1}{n}\right)^{m-r} = \frac{1}{r!}\cdot\frac{m\cdot(m-1)\cdot\ldots\cdot(m-r+1)}{n^r}\cdot\left(1-\frac{1}{n}\right)^{m-r}$$

W przypadku, gdy m i n są duże w porównaniu do r

$$(1-\frac{1}{n})^{(n-r)} \approx e^{-\frac{m}{n}}$$

$$\frac{m\cdot(m-1)\cdot\ldots\cdot(m-r+1)}{n^r} \approx \left(\frac{m}{n}\right)^r$$

Z tego wynika natomiast, że $X \sim \mathrm{Po}\left(\frac{m}{n}\right)$, a więc

$$\binom{m}{r}\left(\frac{1}{n}\right)^r\left(1-\frac{1}{n}\right)^{m-r} \approx e^{-\frac{m}{n}}\frac{\left(\frac{m}{n}\right)^r}{r!}$$

Na tej podstawie można również wywnioskować, że średnia liczba kul w urnie wynosi

$$E(X) = \frac{m}{n}$$

Prawdopodobieństwo tego, że w dowolnej urnie jest co najmniej r kul wynosi

$$P(X = r) = \frac{1}{er!}$$

Każdą urnę możemy traktować jako niezależną zmienną losową o rozkładzie Poissona z parametrem $\frac{m}{n}$, a więc prawdopodobieństwo tego, że w żadnej urnie nie będzie r kul wynosi

$$(1-\frac{1}{er!})^n$$

Na potrzeby dalszych obliczeń udowodnimy teraz krótki lemat.

Lemat

$$(1+\frac{1}{b})^a \leq e^{\frac{a}{b}}$$

Dowód

$$(1+\frac{1}{b})^a = e^{a\ln(1+\frac{1}{b})} = \sum_{k=0}^{\infty}\frac{(a\ln(1+\frac{1}{b}))^k}{k!} \leq \sum_{k=0}^{\infty}\frac{(a/b)^k}{k!} = e^{\frac{a}{b}}$$

Wynika z tego, że

$$(1-\frac{1}{e\cdot r!})^n \leq e^{-\frac{n}{e\cdot r!}}$$

Jak podano w w książce *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis* (Cambridge University Press, USA, 2nd edition, 2017) autorstwa Michaela Mitzenmachera i Eli Upfala, traktując liczbę kul w urnie jako zmienną losową o rozkładzie Poissona zdarzenie, które zachodzi z prawdopodobieństwem p zajdzie z prawdopodobieństwem maksymalnie $pe\sqrt{m}$.

Naszym celem będzie zatem znalezienie, takiego r, dla którego zdarzenie polegające na tym, że w żadnej urnie nie będzie r kul zajdzie z prawdopodobieństwem masymalnie $\frac{e\sqrt{n}}{n^2} < \frac{1}{n}$. Jest to równoznaczne ze znalezieniem takiego r, dla którego prawdopodobieństwo tego zdarzenia będzie równe $\frac{1}{n^2}$. Przeprowadźmy zatem obliczenia

$$\begin{aligned}
e^{-\frac{n}{e\cdot r!}} &\leq \frac{1}{n^2} \\
\frac{1}{e^{\frac{n}{e\cdot r!}}} &\leq \frac{1}{n^2} \\
n^2 &\leq e^{\frac{n}{e\cdot r!}} \\
2\ln(n) &\leq \frac{n}{e\cdot r!} \\
r! &\leq \frac{n}{2e\cdot \ln(n)} \\
\ln(r!) &\leq \ln(n) - \ln(\ln(n)) - \ln(2e)
\end{aligned}$$

Mając

$$n! \leq e\sqrt{n}\left(\frac{n}{e}\right)^n$$

dostajemy

$$r! \leq e\sqrt{r}\left(\frac{r}{e}\right)^r \leq r\left(\frac{r}{e}\right)^r$$

Prowadzi to do dalszych obliczeń

$$\ln(r!) \leq r\ln(r) - r + \ln(r)$$

dla $r = \frac{\ln(n)}{\ln(\ln(n))}$

$$\begin{aligned}
\ln(r!) &= \frac{\ln(n)}{\ln(\ln(n))}\left(\ln(\ln n) - \ln(\ln(\ln n))\right) - \frac{\ln(n)}{\ln(\ln(n))} + \left(\ln(\ln(n)) - \ln(\ln(\ln(n)))\right) \\
&\leq \ln(n) - \frac{\ln(n)}{\ln(\ln(n))} \\
&\leq \ln(n) - \ln(\ln(n)) - \ln(2e)
\end{aligned}$$

Dowiedliśmy zatem, że prawdopodobieństwo t. że w żadnej urnie nie ma $r = \frac{\ln(n)}{\ln(\ln(n))}$ jest maksymalnie $\frac{1}{n}$, a więc pokazaliśmy, że $\forall n \in \mathbb{N}$, $\forall c \in \mathbb{R}$

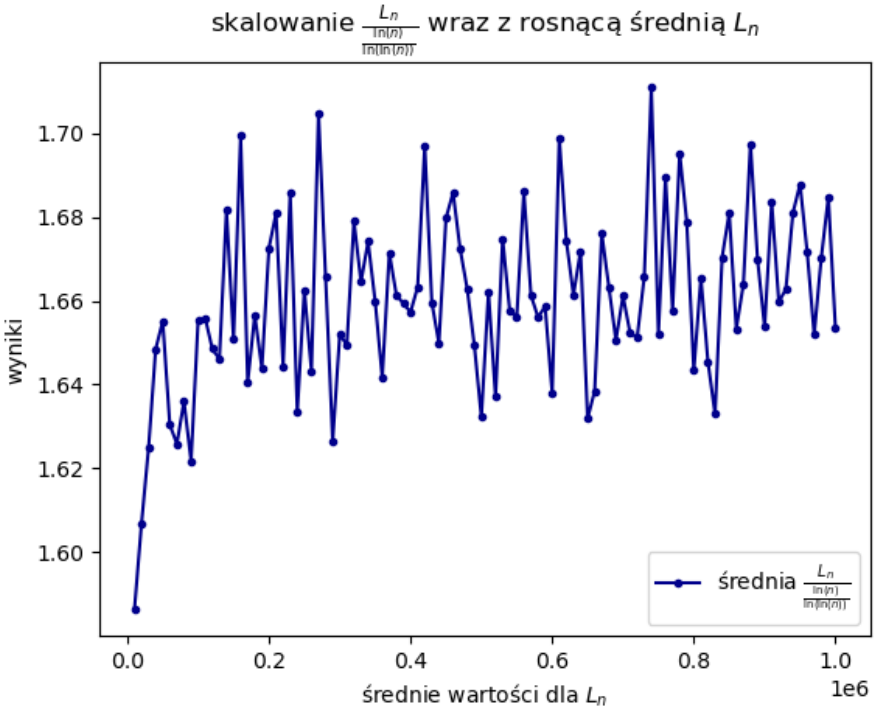
$$L_n^{(1)} \leq \frac{c\cdot\ln(n)}{\ln(\ln(n))}$$

z prawdopodobieństwem $1 - \frac{1}{n}$.

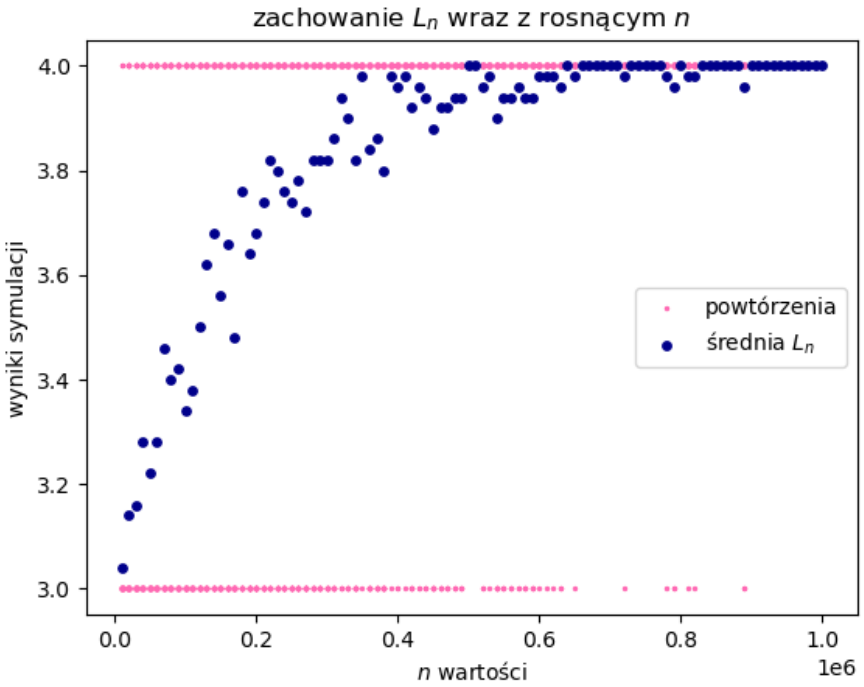
Prawdopodobieństwo tego zdarzenia dąży w granicy do 1, zatem

$$L_n^{(1)} = \frac{\ln(n)}{\ln(\ln(n))}(1 + o(1))$$

Potwierdzeniem tego stwierdzenia jest zamieszczony poniżej wykres, przedstawiający średnie wartości zmiennej losowej $L_n^{(1)}$ przeskalowane przez $\frac{\ln(n)}{\ln(\ln(n))}$. Jak można zauważyć, wartości te wydają się koncentrować w wąskim przedziale. W rezultacie otrzymujemy funkcję, która przybliża się do funkcji stałej.



Rysunek 2: Wyniki eksperymentów dla średniej wartości $L_n^{(1)}$ przeskalowanej przez $\frac{\ln(n)}{\ln(\ln(n))}$.

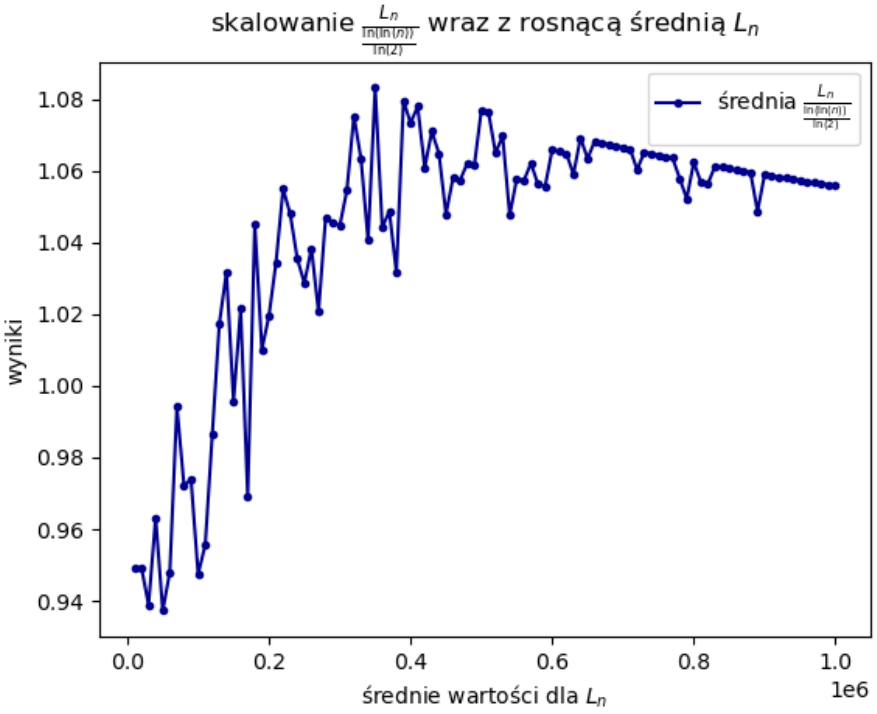


Rysunek 3: Wyniki eksperymentów dla $L_n^{(2)}$.

Powyższy wykres ilustruje zmiany wartości $L_n^{(2)}$ w miarę wzrostu parametru n podczas przeprowadzanej symulacji. Wzrost wartości parametru n wpływa na odchylenia od wartości oczekiwanej jeszcze łagodniej niż w przypadku $L_n^{(1)}$, co wskazuje na silne skoncentrowanie tej wielkości. Niższe wartości wskazują na korzystne właściwości algorytmu przy minimalizowaniu obciążenia pojedynczej urny.

Aby wskazać asymptotyczne tempo wzrostu powołamy się na pracę autorstwa Michaela Mitzenmachera i Eli Upfala *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis* (Cambridge University Press, USA, 2nd edition, 2017), rozdział 17, gdzie zostało wykazane, że losowanie urny d-krotnie i umieszczanie kuli w najmniej zapełnionej z nich można opisać funkcją o złożoności $\frac{\ln(\ln(n))}{\ln(d)} + \Theta(1)$, w szczególności dla $d = 2$ otrzymujemy $\frac{\ln(\ln(n))}{\ln(2)} + \Theta(1)$.

Obliczenia te potwierdza poniższy wykres, przedstawiający średnie wartości zmiennej losowej $L_n^{(2)}$ przeskalowane przez $\frac{\ln(\ln(n))}{\ln(2)}$. Jak można zauważyć, wartości te wydają się koncentrować w wąskim przedziale. W rezultacie otrzymujemy funkcję, która przybliża się do funkcji stałej.



Rysunek 4: Wyniki eksperymentów dla średnich wartości $L_n^{(2)}$ przeskalowanych przez $\frac{\ln(\ln(n))}{\ln(2)}$.

3. Insertion Sort

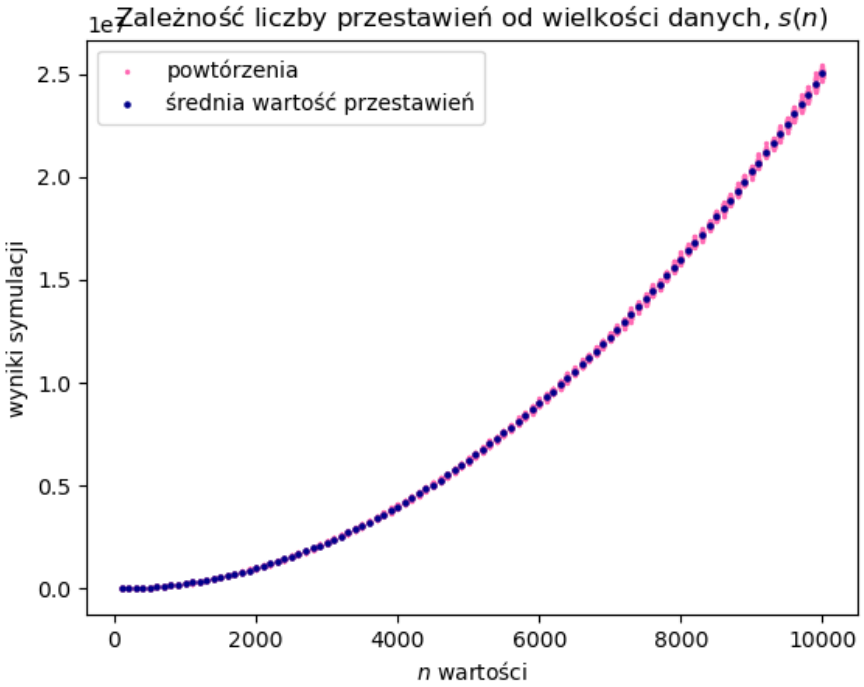
3.1 Opis symulacji

Zaimplementowano oraz przetestowano działanie algorytmu sortowania przez wstawianie (Insertion Sort). W tym celu dla każdego $n \in \{100, 200, \ldots, 10,000\}$ wykonano po $k = 50$ niezależnych powtórzeń:

1. Generowania tablicy $A[1..n]$ będącej losową permutacją zbioru liczb {1, ..., n}. Do tego celu użyto generatora liczb losowych SecureRandom z języka Java.
2. Sortowania wygenerowanej tablicy A.
3. Zapisywania statystyk obejmujących rozmiar danych, liczbę wykonanych porównań między kluczami (elementami tablicy A) oraz liczbę przestawień kluczy.

3.2 Wyniki symulacji i asymptotyka wartości oczekiwanych zmiennych losowych

Podczas analizy asymptotycznej oraz badania problemu skoncentrujemy się na kwestiach porównań i zamian w kontekście algorytmu sortowania przez wstawianie. To właśnie te aspekty istotnie wpływają na złożoność tego algorytmu.



Rysunek 5: Wyniki eksperymentów dla liczby przestawień.

Powyższy wykres ilustruje zmianę liczby przestawień w miarę wzrostu parametru n podczas przeprowadzanej symulacji. Wyniki te mają niski współczynnik koncentracji, gdyż wartości we wszystkich powtórzeniach nieznacznie odchylają się od wartości średniej.

Liczba przestawień w permutacji jest równa liczbie inwersji, ponieważ każde przestawienie skutkuje zmniejszeniem liczby inwersji o jeden.

W celu ustalenia maksymalnej liczby przestawień, a co za tym idzie maksymalny czas działania algorytmu, należey wyznaczyć maksymalną liczbę inwersji w permutacji n-elementowej.

Teza

Maksymalna liczba inwersji w permutacji n-elementowej jest asymptotycznie równa $O(n^2)$.

Dowód

Permutacja charakteryzująca się największą liczbą inwersji to taka, w której każda z dowolnie wybranych par elementów jest w inwersji. Maksymalna liczba inwersji w takiej permutacji jest równa liczbie możliwych par, które można utworzyć ze zbioru n-elementowego.

$$\binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$$

co dowodzi wcześniej postawioną tezę.

Warto zauważyć, że dla dowolnego elementu permutacji liczba jego przestawień jest o jeden mniejsza od liczby jego porównań z innymi elementami. Niech $cmp(n)$ będzie liczbą porównań konieczną do posortowania permutacji n-elementowej z wykorzystaniem algorytmu Insertion Sort, a $cmp_i(n)$ będzie liczbą porównań konieczną do ustawienia i-tego elementu na odpowiednim miejscu, wtedy

$$cmp(n) = cmp_1(n) + cmp_2(n) + \ldots + cmp_n(n)$$

Analogicznie możemy wprowadzić oznaczenie $s(n)$ jako liczba przestawień potrzebna do posortowania permutacji n-elementowej. Dostajemy zatem

$$s(n) = s_1(n) + \ldots + s_n(n) = cmp_1(n) - 1 + \ldots + cmp_n(n) - 1 = cmp(n) - (n-1)$$

Na podstawie tych zależności możemy wyznaczyć maksymalną liczbę porównań jako

$$\frac{n(n-1)}{2} + (n-1) = \frac{n(n-1) + 2(n-1)}{2} = \frac{(n+2)(n-1)}{2} = O(n^2)$$

Dzięki wcześniej uzyskanym wielkościom możemy ustalić, że maksymalny czas działania algorytmu, w przypadku gdy tablica jest posortowana w kolejności odwrotnej, asymptotycznie wynosi $O(n^2)$.

Sytuacja, w której tablica jest już posortowana na wejściu, jest przypadkiem, w którym czas działania algorytmu będzie minimalny.

W takiej sytuacji algorytm nie dokona żadnych przestawień, a jedynie porówna ze sobą $n-1$ elementów. Dostajemy zatem

$$s(n) = O(1)$$

$$cmp(n) = O(n)$$

Czas działania algorytmu jest w tym przypadku wyznaczony przez liczbę porównań i jest asymptotycznie równy $O(n)$.

Aby ustalić średnią liczbę przestawień, a tym samym średni czas działania algorytmu, konieczne jest znalezienie średniej liczby inwersji w permutacji n-elementowej.

Teza

Średnia liczba inwersji w permutacji n-elementowej jest równa

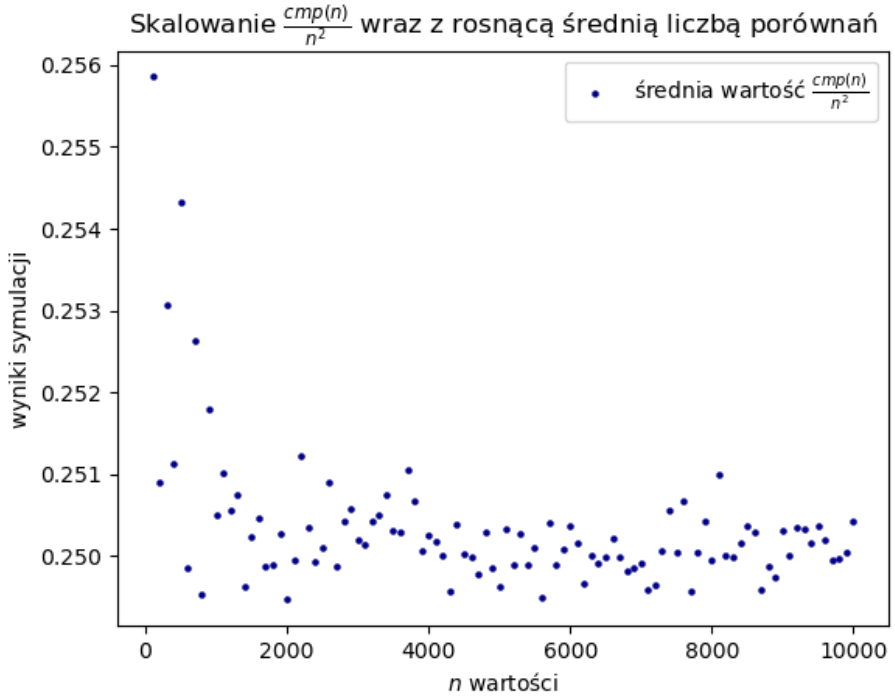
$$\frac{n(n-1)}{4}$$

Dowód

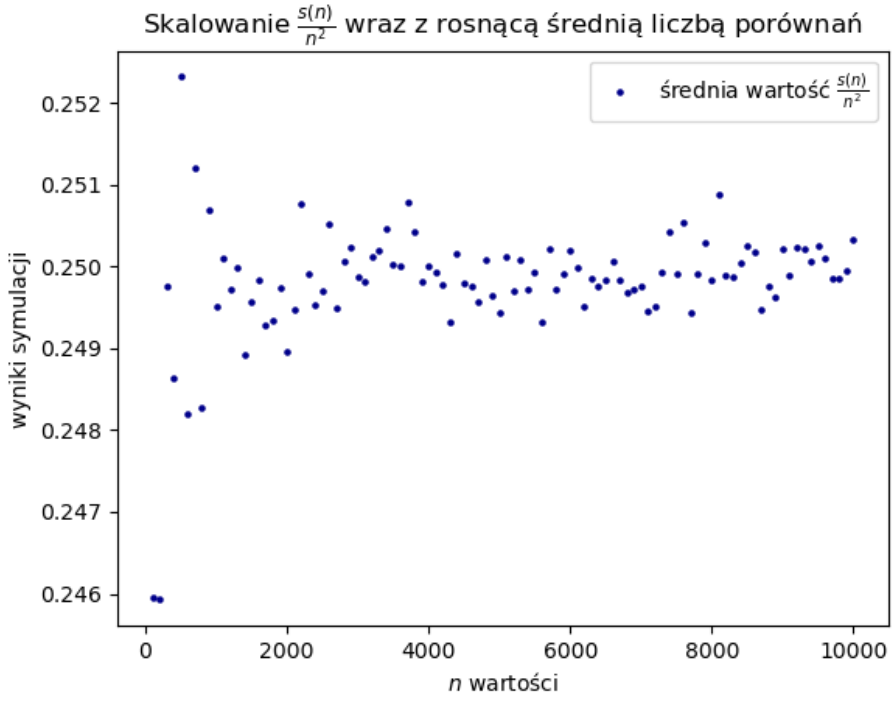
$$s(n) = \frac{s_{\max}(n) + s_{\min}(n)}{2} = \frac{\frac{(n(n-1)}{2} + 0}{2} = \frac{n(n-1)}{4}$$

Na podstawie wyżej udowodnionej tezy jesteśmy w stanie wywnioskować, że asymptotyczne tempo wzrostu średniego czasu działania algorytmu Insertion Sort wynosi $O(n^2)$.

Obliczenia te potwierdzają poniższe wykresy, przedstawiające średnie wartości liczby porównań i przestawień przeskalowane przez n^2. Jak można zauważyć, wartości te wydają się koncentrować w wąskim przedziale. W rezultacie otrzymujemy funkcję, która przybliża się do funkcji stałej.



Rysunek 6: Wyniki eksperymentów dla średniej liczby porównań przeskalowane przez n^2.



Rysunek 7: Wyniki eksperymentów dla średniej liczby przestawień przeskalowane przez n^2.

4. Wnioski

W niniejszym sprawozdaniu skoncentrowano się na dwóch kluczowych zagadnieniach w teorii algorytmów: maximum load w modelu kul i urn oraz analizie złożoności algorytmu Insertion Sort. Obejmują one obszary, w których ilość danych oraz skomplikowanie operacji mają kluczowe znaczenie.

W kontekście modelu kul i urn analizowano maksymalne zapełnienie pojedynczej urny w zależności od strategii umieszczania kul. Przeprowadzono symulacje, eksperymentalnie ustalając asymptotyczną złożoność tych procesów. Wykazano, że maksymalne zapełnienie jest asymptotycznie proporcjonalne do $\frac{\ln(n)}{\ln(\ln(n))}$ w przypadku, gdy losujemy urny z rozkładem jednostajnie oraz $\frac{\ln(\ln(n))}{\ln(2)}$, wtedy gdy losujemy dwie urny z rozkładem jednostjanym, a następnie umieszczamy kule w mniej zapełnionej urnie.

W przypadku algorytmu Insertion Sort skoncentrowano się na analizie liczby porównań i przestawień w permutacjach n-elementowych. Udowodniono, że maksymalna liczba porównań w tablicy n-elementowej jest asymptotycznie równa $O(n^2)$, co wpływa na asymptotyczną złożoność czasową algorytmu Insertion Sort. Symulacje potwierdziły, że średni czas działania tego algorytmu jest również $O(n^2)$, co było wynikiem analizy średniej liczby inwersji w permutacji.

Podsumowując, analiza asymptotyczna oraz eksperymentalne wyniki potwierdzają, że teoria algorytmów pozwala na trafne prognozowanie zachowania różnych procesów, co jest kluczowe w projektowaniu efektywnych algorytmów i optymalnym zarządzaniu zasobami w systemach informatycznych.