Space, rocket and aerospace technologies in science and programming.

Author, MSc: Adrian Szklarski: 10.2023

Subject: A probabilistic approach to the algorithm in the context of the Python language based on the example of the control problem of the missile.

Temat: Ujęcie probabilistyczne algorytmu w kontekście języka Python w oparciu o przykład zagadnienia sterowania pociskiem rakietowym.

Introduction: Good design of a conceptual algorithm greatly influences its effectiveness, more so than how that algorithm will be programmed, what hardwere it will be put on and run. To be able to work well with algorithms, it is necessary to be able to navigate such branches of mathematics as probability calculus, combinatorics, graph/optimization/algorithm/automata theory, operations research or computational complexity.

The paper presents an approach to algorithm analysis with an example of an approach to the problem based on a ground-to-air missile.

Wstęp: *Dobre zaprojektowanie algorytmu koncepcyjnego bardzo wpływa na jego efektywność, bardziej niż sposób w jaki ten algorytm zostanie zaprogramowany, na jakim hardwere zostanie postawiony i uruchomiony. Aby móc pracować dobrze z algorytmami koniecznym jest umiejętność poruszania się w takich działach matematyki jak: rachunek prawdopodobieństwa, kombinatoryka, teoria grafów/ optymalizacji/ algorytmów/ automatów, badania operacyjne czy złożoność obliczeniowa.*

W pracy zostało zaprezentowane podejście do analizy algorytmów wraz z przykładem podejścia do problemu w oparciu o pocisk rakietowy klasy ziemia – powietrze.

Space, rocket and aerospace technologies in science and programming.

Algorithm analysis: is the study of the cost of an algorithm treated as a random variable. When studying an algorithm, one always analyzes the number of steps performed by the algorithm, in other words, the time cost.

Analiza algorytmów: to badanie kosztu algorytmu traktowanego jako zmienna losowa. Podczas badań algorytmu zawsze analizuje się liczbę kroków wykonanych przez algorytm czyli inaczej mówiąc koszt czasowy.

The first step of the analysis: the probabilistic space is the set of data accepted by the algorithm, so it is necessary to define these data (precise determination of the probabilistic model) and probability distribution

Pierwszy krok analizy: przestrzenią probabilistyczną jest zbiór danych przyjmowanych przez algorytm, zatem koniecznym jest określenie tych danych (precyzyjne ustalenie modelu probabilistycznego) oraz rozkładu prawdopodobieństwa.

Universality: the analysis carried out is based on the data of an algorithm of any size and various data so the cost of operation will depend on this data size and the specific layout of this data.

- n natural number (data size),
- d data layout, for a given algorithm there may be many,
- $|d|$ data layout size.

Uniwersalizm: Przeprowadzona analiza oparta jest o dane algorytmu dowolnego rozmiaru i różnych danych zatem koszt działania będzie zależał od tego rozmiaru danych i konkretnego układu tych danych.

- n liczba naturalna (rozmiar danych),
- d układ danych, dla danego algorytmu może istnieć wiele,
- $|d|$ rozmiar układu danych.

The data space for a fixed n is a set :

Przestrzeń danych dla ustalonego n to zbiór:

$$D_n = \{d : |d| = n\}$$

Probability of a data system d of size n in D_n :

Prawdopodobieństwo wystąpienia układu danych d rozmiaru n w :

$$\sum_{D_n} P_{D_n} = 1$$

Cost of an algorithm: if given a one-element set then the cost will be a function of one variable n , since it depends only on n . However, if this set is multi-element and given a finite probability P_{D_n} distribution then the cost for a given n is a random variable defined for D_n .

Koszt algorytmu: jeżeli dany jest zbiór jednoelementowy to koszt będzie funkcją jednej zmiennej n , ponieważ zależy on tylko do n . Jeżeli jednak ten zbiór jest wieloelementowy i ma dany określony rozkład prawdopodobieństwa P_{D_n} wtedy koszt dla danego n jest zmienną losową określoną dla D_n .

The cost is determined by the number of steps performed by the algorithm for a given d, so for a fixed n we work with a random variable T_n [1] that takes integer values greater than zero (positive).
Thus, we have $T_n \geqslant 0$ for $|d|=n$.

Koszt jest określony przez liczbę kroków wykonanych przez algorytm dla danego d, zatem dla ustalonego n pracujemy ze zmienną losową T_n która przyjmuje wartości całkowite większe od zera (dodatnie).
Zatem mamy $T_n \geqslant 0$ dla $|d|=n$.

The study of the probability of an algorithm boils down to the study of the characteristic quantities of a variable T_n, namely:
- expected value
- variance
- standard deviation

When analyzing an algorithm, one usually works with a large number of random variables indexed by size n.

Badanie prawdopodobieństwa algorytmu sprowadza się do badania charakterystycznych wielkości zmiennej T_n czyli:
- wartości oczekiwanej
- wariancji
- odchylenia standardowego

Analizując algorytm pracuje się najczęściej z dużą ilością zmiennych losowych indeksowanych rozmiarem n.

At a fixed value n characteristic values are numbers T_n from which it is difficult to extract information about the analyzed algorithm and its characteristics. In contrast, when working with algorithms, one looks for "how the algorithm behaves as the size of the data increases n, that is, its asymptotic behavior.

Przy ustalonej wartości n charakterystyczne wartości T_n to są liczby, z których trudno jest pozyskać informacje na temat analizowanego algorytmu i jego cech. Natomiast pracując z algorytmami szuka się „jak algorytm zachowuje się wraz ze wzrostem rozmiaru danych n czyli jego asymptotyczne zachowanie”.

$(T_n)_\Theta$ - expected value[2]
$(T_n)_V$ - variance
$(T_n)_\Gamma$ - standard deviation

$(T_n)_\Theta$ - wartość oczekiwana[3]
$(T_n)_V$ - wariancja
$(T_n)_\Gamma$ - odchylenie standardowe

P_{nj} - the probability T_n of it taking the value of $j \geqslant 0$

P_{nj} - prawdopodobieństwo tego, że T_n przyjmie wartość $j \geqslant 0$

$P_n(p)$ function that builds for T_n a series of the form:

$P_n(p)$ funkcja która buduje dla T_n szereg postaci:

$$P_n(p)=\sum_{j\geqslant 0} P_{nj}\, p^j$$

1 Random discrete variable / Losowa zmienna dyskretna.
2 Random variable T_n and are functions of the natural variable n, the asymptotic properties of these functions are studied.
3 Zmiennej losowej T_n i są one funkcjami zmiennej naturalnej n, badane są własności asymptotyczne tych funkcji.

Space, rocket and aerospace technologies in science and programming.

MATHEMATICAL THEOREM 1: **TWIERDZENIE MATEMATYCZNE 1:**

$$\sum_{j\geqslant 0} P_{nj}=1 \quad \rightarrow \quad P_n(1)=1$$

$$P_n(p) \quad / \frac{d}{dt} \quad \rightarrow \quad P_n^{'}(p)=\sum_{j\geqslant 0} jP_{nj}\,p^{j-1}$$

$$(T_n)_{\Theta}=\sum_{j\geqslant 0} jP_{nj} \quad \rightarrow \quad (T_n)_{\Theta}=P_n^{'}(1)$$

$$P_n(p) \quad / \frac{d^2}{d^2t} \quad \rightarrow \quad P_n^{''}(p)=\sum_{j\geqslant 0} j(j-1)\,P_{nj}\,p^{j-2}$$

Thus, from the variation we get: Z wariacji otrzymujemy zatem:

$$(T_n)_V=\sum_{j\geqslant 0}(j-(T_n)_{\Theta})^2 P_{nj}=\sum_{j\geqslant 0}(j^2-2\,j(T_n)_{\Theta}+(T_n)_{\Theta}^2)P_{nj}=\sum_{j\geqslant 0}(j^2-2\,jP_n^{'}(1)+P_n^{'}(1)^2)\,P_{nj}$$

$$(T_n)_V=\sum_{j\geqslant 0}(j^2P_{nj}-2\,jP_n^{'}(1)P_{nj}+P_n^{'}(1)^2P_{nj})=\sum_{j\geqslant 0} j^2P_{nj}-2\,P_n^{'}(1)\sum_{j\geqslant 0} jP_{nj}+P_n^{'}(1)^2\sum_{j\geqslant 0} P_{nj}$$

$$(T_n)_V=\sum_{j\geqslant 0} j^2P_{nj}-2\,P_n^{'}(1)\sum_{j\geqslant 0} jP_{nj}+P_n^{'}(1)^2\sum_{j\geqslant 0} P_{nj}$$

$$(T_n)_V=\sum_{j\geqslant 0} j(j-1)P_{nj}+\sum_{j\geqslant 0} jP_{nj}-P_n^{'}(1)^2 \quad \rightarrow \text{we get , } \textit{otrzymujemy}:$$

$$\boxed{(T_n)_V=P_n^{''}(1)+P_n^{'}(1)-(P_n^{'}(1))^2}$$

$$\boxed{P_n(1)=1} \quad , \quad \boxed{(T_n)_{\Theta}=P_n^{'}(1)} \quad , \quad \boxed{(T_n)_V=P_n^{''}(1)+P_n^{'}(1)-(P_n^{'}(1))^2}$$

Using these formulas we determine $(T_n)_{\Theta}$, $(T_n)_V$ without determining the probabilities P_{nj} but we need to know the formation function $P_n(p)$ given not in the form of a series.

Korzystając z tych wzorów wyznaczamy $(T_n)_{\Theta}$, $(T_n)_V$ bez wyznaczania prawdopodobieństw P_{nj} ale musimy znać funkcję tworzącą $P_n(p)$ daną nie w postaci szeregu. Ponieważ $(T_n)_{\Gamma}=\sqrt{(T_n)_V}$ zatem jest to bardzo potężne narzędzie do probabilistycznej analizy algorytmów.

Example:	**Przykład:**

Image source:https://www.youtube.com/watch?v=CrOCN1-4Ozg, 8.10.2023

Space, rocket and aerospace technologies in science and programming.

Let's consider the rocket missile as a material point with variable mass in three-dimensional motion. Its state during flight is determined by seven variables:

- three components of position x, y, z [4]
- three components of velocity v_x, v_y, v_z
- mass m

Therefore, the system is described by a finite number of real variables $k_1, \ldots, k_n$, whose values at each moment of time determine the state of the system t.

Treating a missile as a material point, also the state of the system of this missile can be treated as a point in Euclidean space E^n that is $k = (k_1 \ldots, k_n)$ - this space is called *state space* or *phase space*.

The dynamic behavior of the system (its time course of states) is governed by some rule from a set of different algorithms (usually described by ordinary differential equations). For the rocket described here, it's a system of ordinary differential equations of which three are of order two and one is of order one. These equations are the three components of acceleration and velocity as functions of position, velocity, mass, time and thrust. By changing the time course of thrust, we change the rules governing the motion of the missile, that is, we control the motion of the missile.

To a particular algorithm (rule) from a set of algorithms corresponds a certain time course of the states of the system of the time function: $k(t)$ that is, for a given algorithm, r the state (rocket) moves along a certain track $path$ in the state space.

Pocisk rakietowy rozpatrzmy jako punkt materialny o zmiennej masie w ruchu trójwymiarowym. Jego stan podczas lotu określa siedem zmiennych:

- trzy składowe położenia x, y, z
- trzy składowe prędkości v_x, v_y, v_z
- masa m

W związku z tym układ opisany jest przez skończoną liczbę zmiennych rzeczywistych $k_1, \ldots, k_n$, których wartości w każdej chwili czasu t określają stan układu.

Traktując pocisk rakietowy jako punkt materialny, również stan układu tego pocisku można potraktować jako punkt w przestrzeni Euklidesowej E^n czyli $k = (k_1 \ldots, k_n)$ - przestrzeń tą nazywa się przestrzenią stanu lub przestrzenią fazową.

Dynamiczne zachowanie się układu (jego przebieg czasowy stanów) rządzi pewna reguła ze zbioru różnych algorytmów (zazwyczaj opisanych zwyczajnymi równaniami różniczkowymi). Dla opisywanej rakiety to układ równań różniczkowych zwyczajnych z którego trzy to równania rzędu drugiego, jedno zaś rzędu pierwszego. Równania te to trzy składowe przyspieszenia i prędkości jako funkcje położenia, prędkości, masy, czasu i ciągu. Zmieniając przebieg czasowy ciągu zmieniamy reguły rządzące ruchem pocisku czyli sterujemy ruchem rakiety.

Konkretnemu algorytmowi (regule) ze zbiorów algorytmów odpowiada pewien przebieg czasowy stanów układu funkcji czasu: $k(t)$ czyli dla danego algorytmu r stan (rakieta) porusza się wzdłuż pewnego toru $path$ w przestrzeni stanu.

4 $x, y, z\,[m]$, $v_x, v_y, v_z\,[m/s]$, $m\,[kg]$

Analyzing the flight path according to the algorithm from the set of algorithms leading to the meeting of the missile with the target, in particular, it is important to bring a given state $k(t_0)=k^0$ to any other state Θ^1 in finite time that is, in general:

- there are flight paths carrying the missile from a state k^0 to $k^1 \in \Theta^1$
- there are flight paths carrying a missile from a state k^0 to $\bar{k}^1 \notin \Theta^1$

Having now a set of algorithms for controlling the missile (governing the behavior of the missile during flight), if having now a particular algorithm in the above set, which will uniquely assign a numerical value (the so-called function or quality indicator) to each of its transitions, this value is called the function value or cost. Such a transition cost $k^0 \rightarrow k^1$ (or $k^0 \rightarrow \bar{k}^1$) depends on the rule r and its track $path$. Such a function is described as follows:

Analizując tor lotu według algorytmu ze zbioru algorytmów doprowadzających do spotkania rakiety z celem, w szczególności ważnym jest doprowadzanie danego stanu $k(t_0)=k^0$ do dowolnego innego stanu Θ^1 w skończonym czasie czyli ogólnie rzecz biorąc:

- istnieją tory lotu przeprowadzające pocisk ze stanu k^0 do $k^1 \in \Theta^1$
- istnieją tory lotu przeprowadzające pocisk ze stanu k^0 do $\bar{k}^1 \notin \Theta^1$

Mając teraz zbiór algorytmów sterowania rakietą (rządzących zachowaniem pocisku podczas lotu), jeżeli mamy teraz konkretny algorytm w powyższego zbioru, który każdemu jego przejściu przyporządkuje jednoznacznie wartość liczbową (tak zwany funkcjonał lub wskaźnik jakości), to tą wartość określa się wartością funkcjonału lub kosztem. Taki koszt przejścia $k^0 \rightarrow k^1$ (lub $k^0 \rightarrow \bar{k}^1$) zależy od reguły r oraz jej toru $path$. Funkcjonał taki opisuje się następująco:

$$V(k^0, k^f, r, path); k^f = k^f \vee \bar{k}^f$$

This function can be the integral of the rate of change of the rocket mass during flight. The mass of the rocket changes primarily during the combustion of the engine propellant and the propellant drive of the pressure accumulator. Thus, the value of this integral is the mass of "combustible chemical materials" consumed during flight to carry out the rocket from the starting point (starting point: position, velocity, mass) to the target according to the specified law of changes in thrust and control vector of the steering machine. Since the rocket projectile consumes all the propellant during launch therefore the optimization should take place at the level of programming changes in the rocket's control vector produced on the rocket body, and resulting from its rotation and the frequency of the flipped rudders. The flip rudder is gas-dynamic, so one of the elements of optimization should be to optimally program the rudder machine's sinics.

Funkcjonałem tym może być całka szybkości zmian masy rakiety podczas lotu. Masa rakiety zmienia się przede wszystkim podczas spalania się materiału pędnego silnika oraz napędu prochowego akumulatora ciśnienia. Zatem wartość tej całki to masa „palnych materiałów chemicznych” zużytych podczas lotu na przeprowadzanie rakiety od punktu startu (punkt początkowy: położenie, prędkość, masa) do celu zgodnie z określonym prawem zmian ciągu i wektora sterowania maszynki sterowej. Ponieważ pocisk rakietowy zużywa cały materiał pędny podczas startu zatem optymalizacja powinna się odbyć na poziomie programowania zmian wektora sterowania rakietą wytwarzanego na korpusie rakiety, a wynikającego z jej obrotu i częstotliwości pracy przerzucanych sterów. Stery przerzucane są gazodynamiczne, zatem jednym z elementów optymalizacji powinno być optymalnie programowanie silnikami maszynki sterowej.

The guided missile algorithm is optimal r^* for $k^0 \to \Theta^1$ if the track reaches a certain state at minimum cost:

Algorytm kierowania pociskiem jest optymalny r^* dla $k^0 \to \Theta^1$ jeżeli tor osiąga pewien stan przy minimalnym koszcie:

$$V(k^0, k^{f*}, r*, path*) \leqslant V(k^0, k^f, r, path)$$

The above condition applies to all rules governing tracks ending in Θ^1 .

Powyższy warunek dotyczy wszystkich reguł rządzących torami kończącymi się w Θ^1 .

There may be more than one optimal flight algorithm that will guide the missile from the starting point to the target (initial state, final state) but the minimum cost is only one:

Może istnieć więcej niż jeden algorytm lotu optymalny który poprowadzi pocisk od punktu startu do celu (stan początkowy, stan końcowy) ale koszt minimalny jest tylko jeden:

$$V(k^0, k^{f*}, r*, path*) = V(k^*, \Theta^1), \quad \forall r^*$$

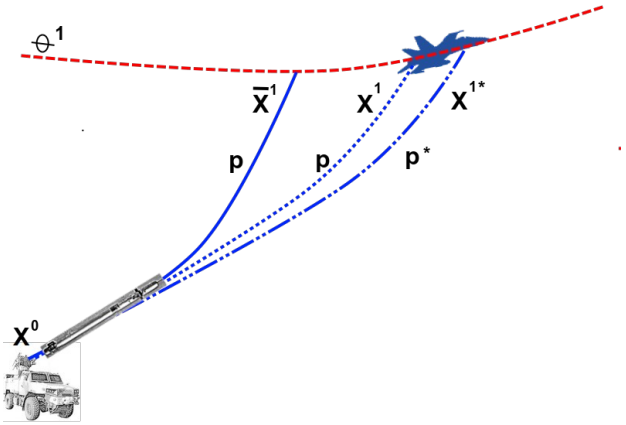


Fig.1: The transition $k^0 \to \Theta^1$ itself can follow an optimal or non-optimal algorithm:

Rys.1: Samo przejście $k^0 \to \Theta^1$ może odbywać według algorytmu optymalnego lub nie optymalnego:

When working with algorithms, there are two basic cases:

1. a set of such points from which a track to the target is formed: $E = \{k^0 : path \quad \exists k^0 \to \Theta^1\}$
2. a set of such points whose track is governed by optimal control rules: $E = \{k^0 : path^* \quad \exists k^0 \to \Theta^1\}$

Podczas pracy z algorytmami można mieć do czynienia z dwoma podstawowymi przypadkami:

1. zbiór takich punktów z których powstaje tor do celu: $E = \{k^0 : path \quad \exists k^0 \to \Theta^1\}$
2. zbiór takich punktów których tor rządzony jest przez optymalne reguły sterowania: $E = \{k^0 : path^* \quad \exists k^0 \to \Theta^1\}$

Both sets may or may not fill E^n that is, from the set of algorithms there may not be an optimal control rule or a classical rule for $k^0 \to \Theta^1$. If there is no such filling then there must be boundary points.

Oba zbiory mogą lecz nie muszą wypełniać E^n czyli z zbiorze algorytmów może nie być optymalnej reguły sterowania lub klasyczna reguła dla $k^0 \to \Theta^1$. Jeżeli nie ma takiego wypełnienia wtedy muszą istnieć punkty brzegowe.

So let's assume that we have a rocket control algorithm in which we have managed to determine functions $(T_n)_\Theta$ and $(T_n)_\Gamma$ or know their asymptotic properties, which characterize the behavior of the algorithm in the case of random data for very large sizes of these data.

Załóżmy zatem, że mamy algorytm sterowania rakietą w którym udało się wyznaczyć funkcje $(T_n)_\Theta$ oraz $(T_n)_\Gamma$ lub znamy ich własności asymptotyczne, które charakteryzują zachowanie się algorytmu w przypadku danych losowych dla bardzo dużych rozmiarów tych danych.

In order to accurately determine the asymptotic properties of the function, we use the following computational complexity:

W celu precyzyjnego określenia własności asymptotycznych funkcji posługujemy się następującymi złożonościami obliczeniowymi:

$f = O(g) \Leftrightarrow$ there is a real constant $c>0$ and a natural number m such that for all $n>m$ there is a $f(n) \leqslant c\,g(n)$

$f = O(g) \Leftrightarrow$ istnieje stała rzeczywista $c>0$ oraz liczba naturalna m taka, że dla wszystkich $n>m$ zachodzi $f(n) \leqslant c\,g(n)$

$f = \Omega(g) \Leftrightarrow$ there is a real constant $c>0$ and a natural number m such that for all $n>m$ there is a $f(n) \geqslant c\,g(n)$

$f = \Omega(g) \Leftrightarrow$ istnieje stała rzeczywista $c>0$ oraz liczba naturalna m taka, że dla wszystkich $n>m$ zachodzi $f(n) \geqslant c\,g(n)$

$$f = \Theta(g) \Leftrightarrow f = O(g) \wedge f = \Omega(g)$$

Working with the Python language in rocket technology (not only) based on the above analysis, one uses the "Big O" notation, which, as I mentioned above, tells how many operations a program/computer must perform in order for a given algorithm to complete its calculations.

Pracując z językiem Python w technice rakietowej (nie tylko) opierając się na powyższej analizie, korzysta się z notacji „Big O”, która jak wyżej wspomniałem mówi o tym ile operacji musi wykonać program/ komputer aby dany algorytm zakończył swoje obliczenia.

In order to illustrate the analysis of algorithms, let's simplify the task as much as possible (now the purpose of the work is not strictly to describe the flight algorithm of the above-mentioned rocket), but to illustrate the operation of the mechanism using the Python language.

W celu zobrazowania analizy algorytmów uprośćmy maksymalnie zadanie (obecnie celem pracy nie jest stricte opisanie algorytmu lotu wyżej wspomnianej rakiety), a jednanie zobrazowanie działania mechanizmu przy użyciu języka Python.

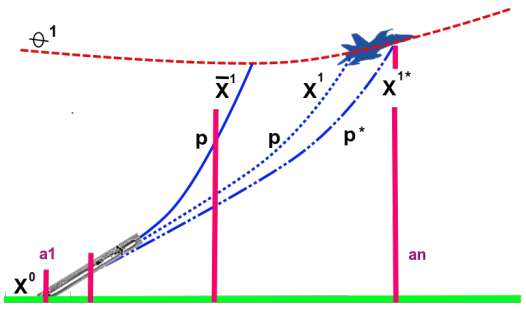


Fig.2: Trajectory points on the x-axis from, for example, 1 to 101

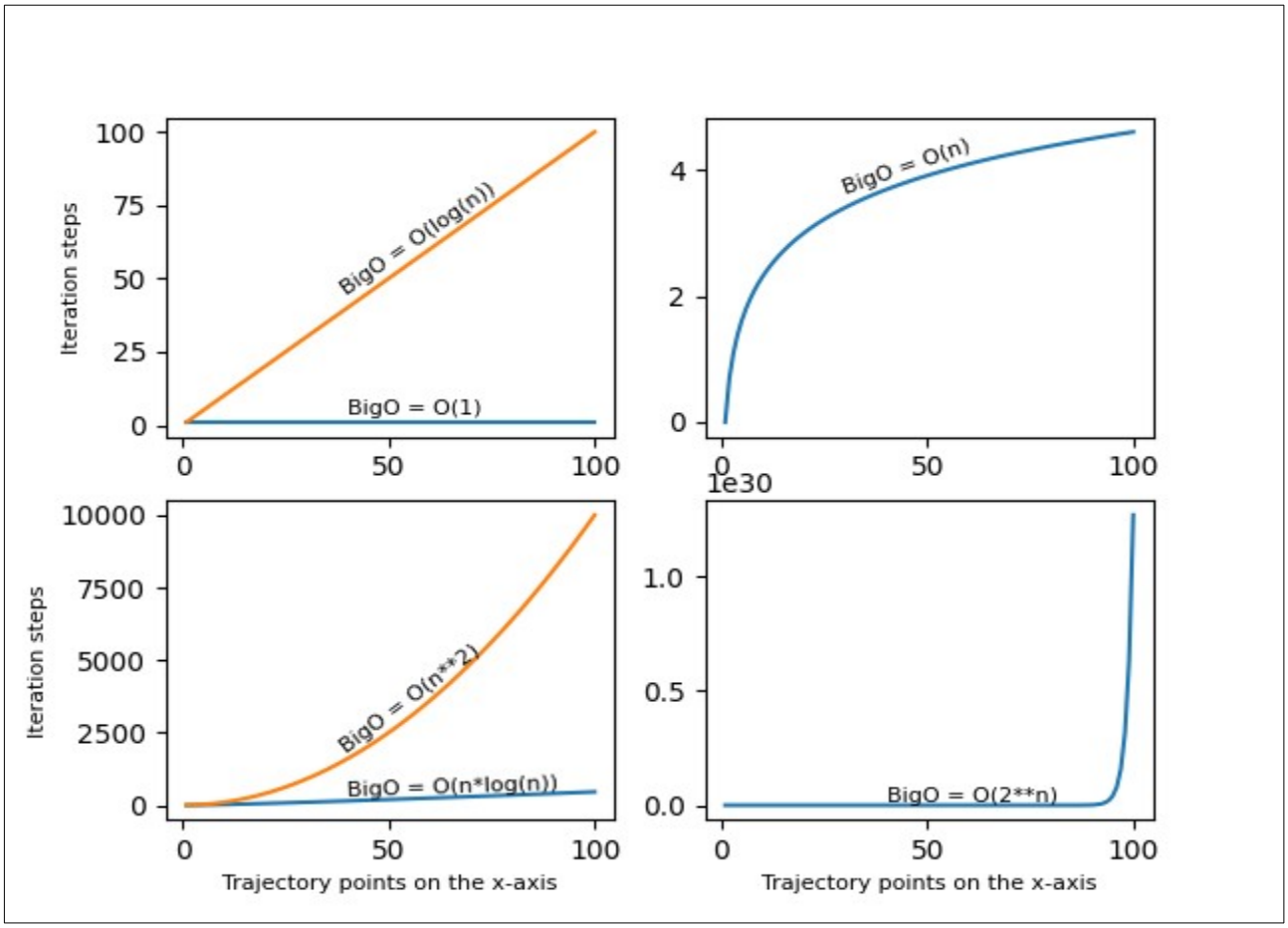
Rys.2: Punkty trajektorii lotu na osi x od przykładowo 1 do 101

Suppose we consider a portion of a flight and we have given 100 trajectory points on the X axis. (Fig. 2). $a_1 \div a_n, \quad n=1,\ldots,101$

Thus, we can encapsulate such data in a list, while the computational complexity will mean:

Załóżmy, że rozważmy fragment lotu i mamy dane 100 punktów trajektorii na osi X. (Rys. 2). $a_1 \div a_n, \quad n=1,\ldots,101$

Zatem dane takie możemy zamknąć w liście, zaś złożoność obliczeniowa będzie oznaczać:



Space, rocket and aerospace technologies in science and programming.

```
import math
import matplotlib.pyplot as plt
from abc import abstractmethod


class BigO:

    def __init__(self, axisX):
        self.axisX = list(range(1, 101, 1))
        self.link = r'/home/.../fig'
        self.FONT_SIZE = 8
        self.get_plot_first()
        self.get_plot_second()
        self.get_plot_third()
        self.get_plot_fourth()
        self.get_show()

    @abstractmethod
    def get_data(ABC):
        #  The abstract class was added for possible further code extension and data reuse.
        x_ver1 = [1] * 100
        x_ver2 = list(range(1, 101, 1))
        x_ver3 = [math.log(x) for x in x_ver2]
        x_ver4 = [x * math.log(x) for x in x_ver2]
        x_ver5 = [math.pow(x, 2) for x in x_ver2]

        x_ver6 = [math.pow(2, x) for x in x_ver2]
        return x_ver1, x_ver2, x_ver3, x_ver4, x_ver5, x_ver6

    def get_plot_first(self):
        plt.subplot(2, 2, 1)
        plt.text(40, 3.8, 'BigO = O(1)', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[0])

        plt.text(40, 44, 'BigO = O(log(n))', rotation=36,
                 rotation_mode='anchor', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[1])
        plt.ylabel("Iteration steps", fontsize=self.FONT_SIZE)

    def get_plot_second(self):
        plt.subplot(2, 2, 2)
        plt.text(30, 3.6, 'BigO = O(n)', rotation=20,
                 rotation_mode='anchor', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[2])

    def get_plot_third(self):
        plt.subplot(2, 2, 3)
        plt.text(40, 300, 'BigO = O(n*log(n))', rotation=2,
                 rotation_mode='anchor', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[3])

        plt.text(40, 1785, 'BigO = O(n**2)', rotation=38,
                 rotation_mode='anchor', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[4])
        plt.xlabel("Trajectory points on the x-axis", fontsize=self.FONT_SIZE)
        plt.ylabel("Iteration steps", fontsize=self.FONT_SIZE)
```

```
    def get_plot_fourth(self):
        plt.subplot(2, 2, 4)
        plt.text(40, 1 * 1e28, 'BigO = O(2**n)', rotation=0,
                 rotation_mode='anchor', fontsize=self.FONT_SIZE)
        plt.plot(self.axisX, self.get_data()[5])
        plt.xlabel("Trajectory points on the x-axis", fontsize=self.FONT_SIZE)

    def get_show(self):
        # creating a dictionary
        font = {'size': 12}
        # using rc function
        plt.rc('font', **font)

        plt.savefig(self.link + '/figure1.png')
        plt.show()

if __name__ == '__main__':
    BigO(100)
```

Lowest computational complexity / Najmniejsza złożoność obliczeniowa:

$O(1)$ - On this complexity of the algorithm we work (no matter how many elements there 10 will be 10^{10} or more) when we have a certain minimum number of operations or only one. Hashing arrays, that is, enumerating elements in a set or dictionary, works on this complexity.

$O(1)$ - na tej złożoności algorytmu pracujemy (nie zależnie od ilości elementów czy będzie ich 10 czy 10^{10} czy więcej) gdy mamy określoną minimalną ilość operacji lub tylko jedną. Na takiej złożoności pracują tablice hashujące czyli wyliczanie elementów w set lub słowniku.

Very good computational complexity / Bardzo dobra złożoność obliczeniowa:

$O(\log(n))$ - is the complexity that is the logarithm of the size of the set, that is, the operations are less than the size of the set, such as binary search.

$O(\log(n))$ - jest to złożoność będąca logarytmem z rozmiaru zbioru, czyli operacji jest mniej niż rozmiar zbioru, np. wyszukiwanie binarne.

Further complexity is considered good / Dalsze złożoności uważane są za dobre:

$O(n)$ - we perform as many operations as there are in the set

$O(n)$ - wykonujemy tyle operacji ile jest w zbiorze

$O(n*\log(n))$ - is a combination of two complexities, having the size of the data we perform the operation data size times the logarithm of the data. Very efficient sorting algorithms such as quicksort or teamsort.

$O(n*\log(n))$ - to połączenie dwóch złożoności, mając rozmiar danych wykonujemy operacji rozmiar danych razy logarytm z tych danych. Bardzo wydajne algorytmy sortowania np. quicksort czy teamsort.

Poor complexity / Słaba złożoność:

$O(n^2)$ - we perform n-square operations from the set data

$O(n^2)$ - wykonujemy n-kwadrat operacji z danych zbioru

Very bad complexity / Bardzo zła złożoność:

$O(2^n)$ - Complexity grows very quickly, it should be **avoided**!, necessary optimization of the algorithm.

$O(2^n)$ - złożoność bardzo szybko rośnie, należy jej **unikać**!, konieczna optymalizacja algorytmu.

Attention should still be paid to metamathematical formulas that facilitate the calculation of the expected value and variation of a random variable T_n for the case when the generating function is the product of the generating functions. For the proof, I refer you to the relevant literature.

Należy zwrócić jeszcze uwagę na formuły metamatematyczne ułatwiające obliczenie wartości oczekiwanej i wariacji zmiennej losowej T_n dla przypadku kiedy funkcja tworząca jest iloczynem funkcji tworzących. W celu zapoznaniem się z dowodem odsyłam od odpowiedniej literatury.

$$R_n(p)=\sum_{j\geqslant 0} R_{nj} z^j, \quad Q_n(p)=\sum_{j\geqslant 0} Q_{nj} z^j, \text{ and } P_n(p)=R_n(p)Q_n(p)$$

and are creation functions for random variables therefore:

i są to funkcje tworzące dla zmiennych losowych zatem:

$$(T_n)_\Theta=(U_n)_\Theta+(V_n)_\Theta, \text{ and } (T_n)_\Gamma=(U_n)_\Gamma+(V_n)_\Gamma$$

In conclusion:

Having a given $(T_n)_\Theta=O(n),\ (T_n)_\Gamma=O(1)$ algorithm runs in linear time with respect to its data, while the random variable that is the cost of running this algorithm centers around the expected value, and this means that the standard deviation is constant and does not depend on the size of the data.

Podsumowując:

Mając dany $(T_n)_\Theta=O(n),\ (T_n)_\Gamma=O(1)$ algorytm działa w czasie liniowym względem swoich danych, zaś zmienna losowa będąca kosztem działania tego algorytmu skupia się wokół wartości oczekiwanej, a to oznacza, że odchylenie standardowe jest stałe i nie zależy od rozmiaru danych.

Having a given $(T_n)_\Theta=O(nlog(n)),\ (T_n)_\Gamma=O(n)$ this standard deviation is not constant, but with the increase of the variable n the cost of running the algorithm better and better clusters around the expected value of the order $nlog(n)$, that is, with the increase n increases the probability of running the algorithm over time $nlog(n)$.

Mając dany algorytm $(T_n)_\Theta=O(nlog(n)),\ (T_n)_\Gamma=O(n)$ to odchylenie standardowe nie jest stałe, ale ze wzrostem zmiennej n koszt działania algorytmu coraz lepiej się skupia wokół wartości oczekiwanej rzędu $nlog(n)$ czyli ze wzrostem n rośnie prawdopodobieństwo działania algorytmu w czasie $nlog(n)$.

Given an algorithm $(T_n)_\Theta = O(n)$, $(T_n)_\Gamma = O(n)$, this running time is linear, but the random variable does not center around its expected value.

Mając dany algorytm $(T_n)_\Theta = O(n)$, $(T_n)_\Gamma = O(n)$ to czas działania jest liniowy, ale zmienna losowa nie skupia się wokół swojej wartości oczekiwanej.

To sum up:
Algorithm design and analysis is not an easy matter, especially when it comes to flying objects, especially at high speeds like missiles. It should be noted that algorithms may not behave according to our expectations, but for random data it will most often work in linear time. The whole analysis, however, requires deeper work with functions to be able to draw appropriate conclusions about the possible behavior of the algorithm especially for large data sets.

Reasumując:
Projektowanie i analiza algorytmów nie jest sprawą łatwą, szczególnie jeżeli chodzi o obiekty latające, szczególnie z dużymi prędkościami tak jak pociski rakietowe. Należy zwrócić uwagę, że algorytmy mogą nie zachowywać się zgodnie z naszymi oczekiwaniami ale dla losowych danych najczęściej będzie działał w czasie liniowym. Cała analiza jednak wymaga głębszych prac z funkcjami $(T_n)_\Theta$, $(T_n)_\Gamma$ żeby móc wyciągnąć odpowiednie wnioski o możliwym zachowaniu się algorytmu szczególnie dla dużych zbiorów danych.

Literature:
[1]: L. Baranowski, A.Kreczmar, W.Rytter: Analiza Algorytmów, WNT
Logo: Photo from:
https://pixabay.com/pl/vectors/bomba-atomowa-rakiet-atomowych-149833/

https://pixabay.com/pl/illustrations/listy-abc-szkolenie-pyton-linia-483010/