Algorytmicznie optymalne wybory życiowe

Dominik Kozimor

27 kwietnia 2025

1 Wprowadzenie

Proszę o wykonanie podanych zadań, a następnie wrzucenie rozwiązań na platformie Github lub dowolnej innej wspierającej system kontroli wersji. Rozwiązanie zadania pierwszego to skrypt w języku Python - natomiast proszę, aby drugie i trzecie były zawarte w pliku formatu Jupyter - możesz użyć środowiska Google Colab albo innego, dowolnego rozwiązania (polecam środowisko Anaconda).

Ostatnie, bonusowe zadanie możesz przesłać w dowolnym formacie, ale proszę, aby znalazło się w tym samym repozytorium co pozostałe. Zamieść również plik `requirements.txt` z użytymi bibliotekami oraz ich wersjami, w szczególności jeśli będziesz używać niewymienionych w opisie kursu.

Oceniane będą przede wszystkim:

- działanie rozwiązania (trochę słabo, jeśli nie działa)
- przejrzystość i czytelność kodu
- optymalizacja kodu - w szczególności unikanie pętli `for` tam, gdzie można użyć operacji z biblioteki `numpy`
- zadania 1-3 posiadają również dodatkowe kryteria oceniania „z gwiazdką“, za które można otrzymać ekstra punkty

2 Zadania

Każde z zadań od 1 do 3 oceniane jest na 10+3 punkty (gdzie można otrzymać 3 dodatkowe punkty). Zadanie bonusowe oceniane jest na 3 punkty maksimum.

2.1 Zadanie 1 - Sudoku

Sudoku składa się kwadratu o wymiarach 9×9 podzielonego na 9 małych kwadratów o wymiarach 3×3. Na potrzeby zadania, kolejne małe kwadraty numerujemy wierszami od 1 do 9. W poprawnym rozwiązaniu: w każdym wierszu, każdej kolumnie i każdym małym kwadracie znajdują się cyfry 1–9. W poprawnym rozwiązaniu ktoś zamienił miejscami dwa małe kwadraty. Proszę napisać program, który wskaże zamienione kwadraty.

W pierwszym i jedynym wierszu standardowego wyjścia program powinien wypisać dwa numery małych kwadratów, które zostały zamienione. Numery te należy wypisać w porządku rosnącym.

Przykład
Dla danych wejściowych:
8 1 2 7 5 3 6 4 9
9 4 3 6 8 2 1 7 5
6 7 5 4 9 1 2 8 3
1 5 4 3 6 8 9 2 6
3 6 9 1 7 7 2 1 1
2 7 8 4 5 2 5 3 4
5 2 1 9 7 4 2 3 7
4 3 8 5 2 6 8 4 5
7 9 6 3 1 8 1 6 9

Poprawną odpowiedzią jest:
5 9

Zostały zamienione miejscami kwadrat środkowy (5) z prawym dolnym kwadratem (9).

Bonus
Oszacuj na podstawie wiedzy własnej oraz zaczerpniętej z sieci złożoność obliczeniową swojego algorytmu (zastosuj notację „big O“, gdzie np. złożoność O(n) oznacza liniową.). Uzasadnij swój wybór krótkim komentarzem w kodzie.

2.2 Zadanie 2 - wyznaczanie π metodą Monte Carlo

Wyobraźmy sobie największe koło, które można wpisać w kwadrat o boku długości 2 z bokami od -1 do 1 na obu osiach układu kartezjańskiego (ma on środek w punkcie $(0, 0)$). Koło ma promień 1 i pole powierzchni π, natomiast kwadrat ma pole powierzchni $2 \cdot 2 = 4$.

Stosunek ich pól wynosi więc $\frac{\pi}{4}$.

Możemy przybliżyć wartość liczby π metodą Monte Carlo za pomocą następującej procedury:

- Rysujemy kwadrat oraz wpisane w niego największe możliwe koło.
- Losowo rozrzucamy dużą liczbę (n) punktów wewnątrz kwadratu (próbki generujemy z rozkładu jednostajnego na przedziale $[-1, 1]$ dla obu współrzędnych x oraz y).
- Zliczamy, ile punktów trafiło do wnętrza koła.
- Liczbę punktów wewnątrz koła dzielimy przez n i wynik mnożymy przez 4 — otrzymujemy przybliżenie π.

Zadanie
Twoim zadaniem jest napisać program w Pythonie, który:
- korzysta z biblioteki `numpy` do generowania próbek,
- przybliża wartość liczby π zgodnie z powyższą metodą,
- wyświetla przybliżoną wartość liczby π.

Bonus
Jeśli chcesz zdobyć dodatkowe punkty - stwórz wizualizację punktów za pomocą biblioteki `matplotlib`:
- Punkty trafiające do wnętrza koła oznacz innym kolorem niż punkty znajdujące się na zewnątrz
- Dodaj tytuł i legendę do wykresu

2.3 Zadanie 3 - problem Monty'ego Halla

Problem Monty'ego Halla to klasyczny problem probabilistyczny, który pochodzi z amerykańskiego teleturnieju „Let's Make a Deal". Uczestnik staje przed trzema drzwiami:
- Za jednymi drzwiami znajduje się nagroda (samochód), a za pozostałymi dwoma – kozy.
- Gracz wybiera jedne drzwi.
- Prowadzący (Monty Hall), który zna zawartość za drzwiami, otwiera jedne z dwóch pozostałych drzwi, zawsze pokazując kozę.
- Następnie gracz dostaje szansę na zmianę swojej pierwotnej decyzji — może zostać przy swoim wyborze lub zmienić drzwi na drugie nieotwarte.

Intuicyjnie może się wydawać, że zmiana lub pozostanie przy pierwotnym wyborze nie ma znaczenia, jednak matematyka mówi coś innego: Zawsze warto zmienić drzwi, bo wtedy prawdopodobieństwo wygranej wzrasta z $\frac{1}{3}$ do $\frac{2}{3}$ (!)

Nie wierzysz? Sprawdź sam!

Zadanie
Twoim zadaniem jest napisać program w Pythonie, który:
- korzysta z biblioteki `numpy` do przeprowadzenia dużej liczby symulacji (np. $n = 10000$),
- symuluje rozgrywki problemu Monty'ego Halla,
- wyznacza empiryczne prawdopodobieństwo wygranej w dwóch strategiach, pozostania przy pierwotnym wyborze i zmiany drzwi po odsłonięciu kozy.

Bonus
Jeśli chcesz zdobyć dodatkowe punkty, stwórz wykres za pomocą `matplotlib`, który:
- porównuje wyniki obu strategii (np. w formie wykresu słupkowego lub liniowego).
- poprawnie opisuje osie, dodaje legendę oraz tytuł wykresu.

Bonus 2
Opisz w komentarzu dlaczego zmiana drzwi jest lepszą strategią.

Podpowiedź
Polecam ten materiał opowiadający o problemie: https://www.youtube.com/watch?v=9vRUxbzJZ9Y

2.4 Zadanie bonusowe

Opisz własnymi słowami, stosując rysunki lub nie w jaki sposób rozumiesz pojęcie całki matematycznej. Twój opis powinien być zrozumiały dla przeciętnego licealisty z 2-3 klasy, który nie miał wprowadzonego pojęcia pochodnej. Zachęcam do opowiedzenia o nim w bardziej intuicyjny sposób oraz korzystania z rysunków, ewentualnie wizuzalicji ;)

3 Źródła

Zadanie 1 jest oparte na zadaniu o tej samej nazwie z III etapu Olimpiady o Diamentowy Indeks AGH (Informatyka) z roku 2022/2023.