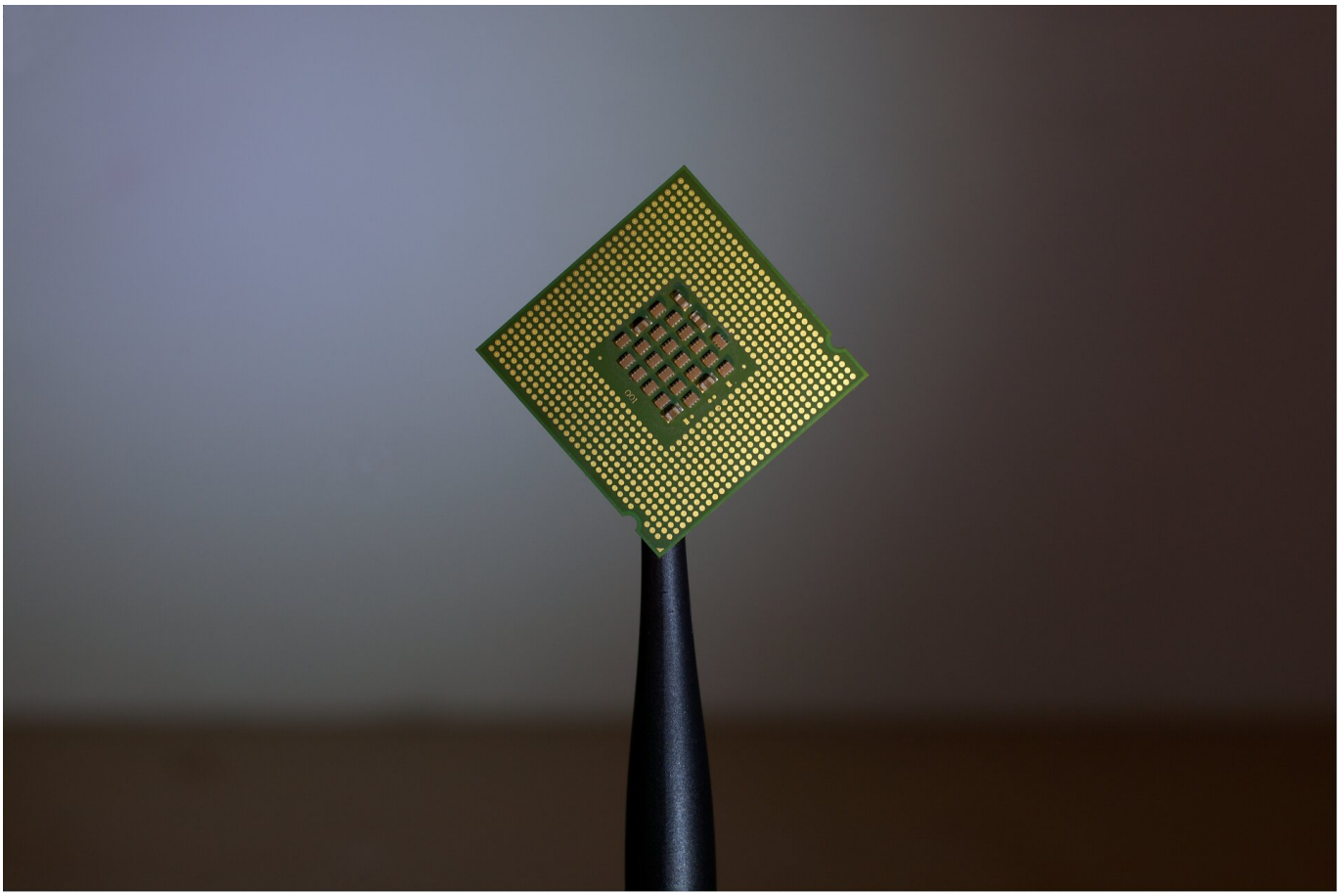


Złożoność obliczeniowa algorytmów

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Brian Kostiuk, domena publiczna.

Niezależnie od tego, z jakiego języka programowania się korzysta, należy zwracać uwagę na liczbę operacji wykonywanych przez program. Jest to ważne zagadnienie, związane bezpośrednio z zastosowanymi algorytmami. Od ich wydajności zależy szybkość działania programu – a to kryterium w dużym stopniu decyduje o użyteczności aplikacji.

Gdy zastanawiamy się więc nad rozwiązaniem określonego problemu, musimy odpowiedzieć sobie na dwa pytania: czy zastosowany algorytm jest poprawny i jaka jest jego złożoność obliczeniowa?

Więcej informacji o złożoności obliczeniowej znajdziesz w e-materiałach:

- Algorytmy o złożoności kwadratowej,
- Algorytmy o złożoności logarytmicznej i liniowo-logarytmicznej,
- Algorytmy o złożoności wykładniczej,

- Jak porównywać złożoność obliczeniową?.

> **Twoje cele**
>
> - Wyjaśnisz, czym jest złożoność obliczeniowa algorytmu.
> - Scharakteryzujesz, jak określać złożoność obliczeniową algorytmów.
> - Przeanalizujesz wybrane rzędy złożoności czasowej.
> - Wykonasz ćwiczenia sprawdzające twoją wiedzę z zakresu złożoności obliczeniowej algorytmów.

Przeczytaj

Jak zdefiniować złożoność obliczeniową?

Złożoność obliczeniowa algorytmu pozwala określić ilość zasobów komputerowych niezbędnych do wykonania czynności opisanych w algorytmie w zależności od rozmiaru danych wejściowych.

Wyróżnia się złożoność:

- czasową;
- pamięciową.

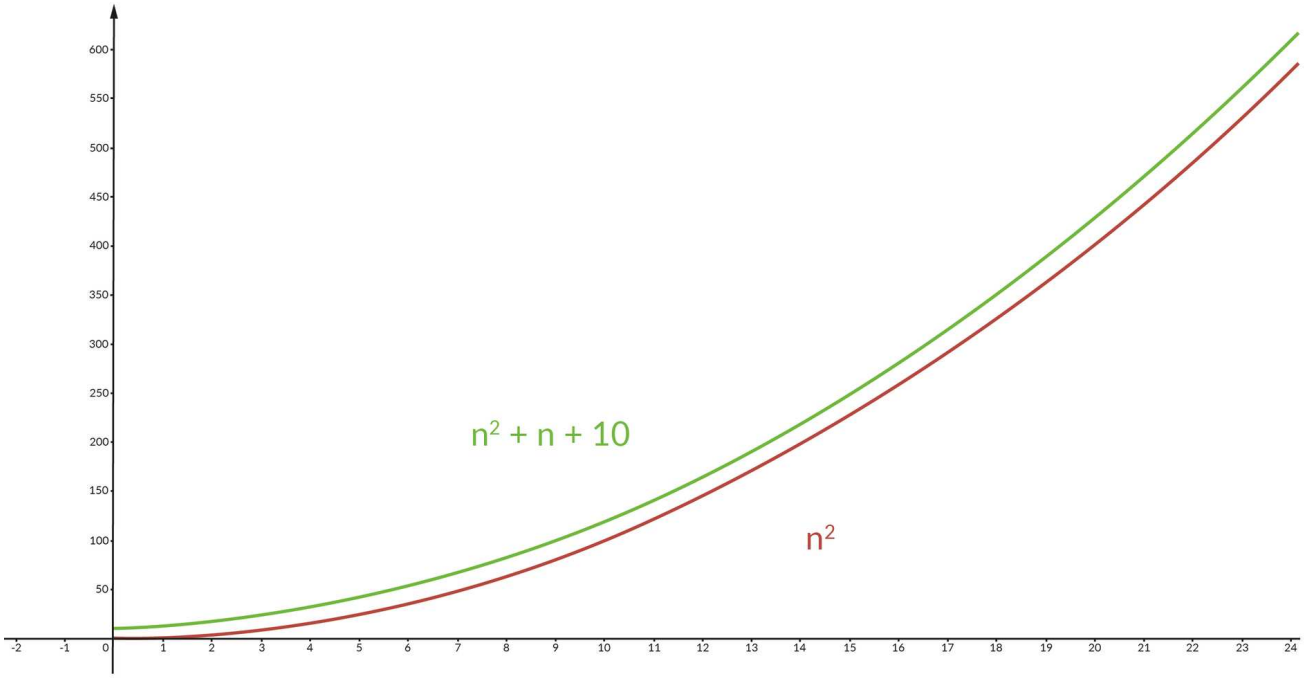
Pierwsza z nich odnosi się do liczby wykonywanych operacji i jest niezależna od komputera oraz języka programowania. Natomiast druga odnosi się do rozmiaru wykorzystywanej pamięci.

Załóżmy, że napisaliśmy program, który przyjmuje od użytkownika n słów, a następnie modyfikuje w pewien sposób wszystkie otrzymane wyrazy i wypisuje ostateczny rezultat przekształceń.

Na każdym słowie wykonywana jest pewna liczba operacji, a zatem szybkość działania programu zależy głównie od rozmiaru danych wejściowych. Opiszmy liczbę wykonywanych operacji. Posłużymy się w tym celu pewną funkcją f. Przyjmijmy, że ma ona następującą postać:

$$f(n) = n^2 + n + 10$$

Dla dużych argumentów wartość n^2 jest o wiele większa niż $n + 10$. Wynika z tego, że wartość funkcji f zależy głównie od składnika n^2.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Aby przedstawić powyższy wniosek w sposób matematyczny, posłużymy się notacją duże O:

$$f(n) = O(n^2)$$

W taki właśnie sposób wyznaczyliśmy **złożoność czasową** przykładowego algorytmu; jest to złożoność **kwadratowa**.

> **Ważne!**
>
> Liczba operacji wykonywanych przez program może być zależna nie tylko od wielkości danych wejściowych, ale także od nich samych. W takiej sytuacji podczas wyznaczania złożoności obliczeniowej bierzemy pod uwagę najbardziej pesymistyczny przypadek (zakładamy, że realizacja algorytmu pochłonie największą ilość zasobów systemu komputerowego), co znaczy, że dla określania złożoności obliczeniowej będziemy używać złożoności pesymistycznej.
>
> Dla dokładniejszej analizy złożoności uwzględniana jest również złożoność oczekiwana – istnieje wiele przypadków, w których porównywane algorytmy mają tę samą złożoność pesymistyczną, jednak w przypadku średnim nie działają tak samo szybko. Wyróżniamy również złożoność optymistyczną, która określa

> złożoność dla najlepszego przypadku danych. W praktyce jednak często nie jest ona brana pod uwagę.

Operacje dominujące

Określając czasową złożoność obliczeniową, bierzemy pod uwagę tzw. **operacje dominujące**. Są to operacje charakterystyczne dla danego algorytmu, a więc mające największy wpływ na czas potrzebny do wykonania algorytmu. Przeanalizujmy przykładowy pseudokod prostego programu:

```
n ← wprowadź dodatnią liczbę całkowitą
suma ← 0
i ← 1
j ← 1

dopóki i <= n wykonuj:
    suma ← suma + i
    i ← i + 1

dopóki j <= n wykonuj:
    suma ← suma + j
    j ← j + 2

wypisz(suma)
```

Program ten wykonuje m.in. operacje odczytu i zapisu danych oraz dodawania.
O czasie działania programu decyduje jednak tak naprawdę liczba wykonywanych dodawań. Jeżeli program wykona ich ponad milion, to czas potrzebny na zapis czy odczyt danych staje się mało istotny. Operacje dodawania są zatem operacjami dominującymi w przedstawionym algorytmie.

Zastanówmy się nad kolejnym przykładem:

```
a ← 5
b ← 10
```

```
c ← a + b

wypisz(c)
```

W powyższym algorytmie nie ma operacji dominujących, czyli takich, które zajmowałyby większość czasu potrzebnego do wykonania w zależności od danych wejściowych. Nasz program zawsze wykona niezmienną liczbę operacji.

Przeanalizujmy jeszcze jeden przykład:

```
n ← wprowadź dodatnią liczbę całkowitą

jeżeli n parzyste:
    wypisz(n / 2)

w przeciwnym razie:
    i ← 0
    suma ← 0

    dopóki i < n wykonuj:
        suma ← suma + 2 * i
        i ← i + 1

    wypisz(suma)
```

Ten z kolei program może być wykonywany przez różny czas w zależności od tego, jaką wartość będzie miała zmienna `n`. W takiej sytuacji zawsze bierzemy pod uwagę najbardziej pesymistyczny przypadek. Operacjami dominującymi są więc:

`suma ← suma + 2 * i` oraz `i ← i + 1`.

Słownik

> **efektywność algorytmu**

praktyczne zastosowanie algorytmu w programie; miara tego, jak bardzo korzystna jest wybrana metoda

notacja duże O

miara określająca, w jaki sposób rośnie wartość funkcji wraz ze zwiększaniem jej argumentów; służy do wyznaczania górnych granic asymptotycznych

operacje dominujące

operacje charakterystyczne dla danego algorytmu, na ogół zajmujące w nim najwięcej czasu

pamięciowa złożoność obliczeniowa

ilość pamięci potrzebnej do wykonania zadania, wyrażona jako funkcja ilości danych

rząd złożoności

właściwość służąca do opisywania czasu działania algorytmu

złożoność czasowa

ilość czasu potrzebnego do wykonania zadania, wyrażona jako funkcja ilości danych

złożoność obliczeniowa

ilość zasobów komputerowych potrzebnych do wykonania zadania

złożoność oczekiwana

inaczej złożoność średnia; ilość zasobów potrzebna do zrealizowania zadania dla statystycznie oczekiwanych danych; zapisywana za pomocą notacji theta – Θ

złożoność optymistyczna

ilość zasobów potrzebna programowi do zrealizowania zadania dla najkorzystniejszego przypadku danych; zapisywana za pomocą notacji „małe O” – o

złożoność pesymistyczna

ilość zasobów potrzebna do zrealizowania zadania w przypadku najgorszych danych; zapisywana za pomocą notacji „duże O” – O

Prezentacja multimedialna

Polecenie 1

Zapoznaj się z prezentacją multimedialną przedstawiającą rodzaje złożoności czasowej. Omów na forum klasy jedną z nich.

Rodzaje złożoności czasowej

1

Złożoność czasowa zaproponowanego rozwiązania jest często uwzględniana w punktacji zadań maturalnych.
Źródło: Isabelle Grosjean, pl.wikipedia.org, CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Większość algorytmów ma złożoność czasową, która jest proporcjonalna do jednej z następujących funkcji:

- złożoność stała (1);
- złożoność logarytmiczna $(\log n)$;
- złożoność liniowa (n);

- złożoność liniowo-logarytmiczna $(n \log n)$;
- złożoność kwadratowa $\left(n^2\right)$;
- złożoność wykładnicza (2^n).

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Złożoność stała

W przypadku złożoności stałej problem rozwiązywany jest z wykorzystaniem **stałej** liczby operacji dominujących (**bez względu na rozmiar danych wejściowych**).

Przykład algorytmu: obliczanie wartości wyrażeń stałych, np. obliczanie liczby przekątnych wielokąta wypukłego na podstawie liczby jego wierzchołków.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Złożoność logarytmiczna

W przypadku złożoności logarytmicznej problem o danych rozmiaru n może zostać zredukowany (za pomocą stałej liczby operacji dominujących) do problemu rozmiaru $\frac{n}{2}$. Podproblemy algorytmów o złożoności logarytmicznej dzielone są na pół.

Przykład algorytmu: szybkie potęgowanie, obliczanie NWD, przeszukiwanie binarne.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Złożoność liniowa

W przypadku złożoności liniowej dla n danych algorytm wykonuje n razy stałą liczbę operacji dominujących. Czas rozwiązania problemu jest więc wprost proporcjonalny do wielkości danych — zachodzi zależność liniowa pomiędzy czasem, a danymi wejściowymi.

Przykład algorytmu: szukanie maksymalnego oraz minimalnego elementu, schemat Hornera, którego celem jest wyznaczenie wartości wielomianu stopnia n.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Złożoność liniowo-logarytmiczna

W przypadku tej złożoności problem, w którym mamy n danych, w liniowej liczbie operacji da się sprowadzić do rozwiązania dwóch problemów o rozmiarach $\frac{n}{2}$.

Przykład algorytmu: sortowanie przez scalanie,

6

Materiał audio dostępny pod adresem:

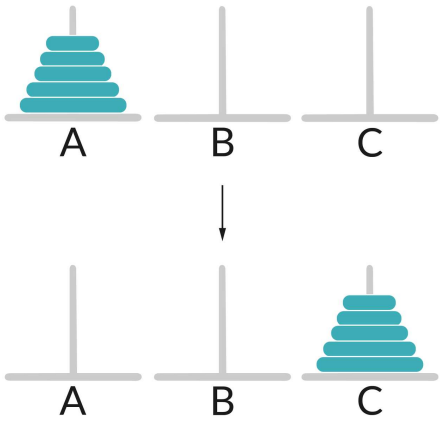
https://zpe.gov.pl/b/PLbEkkFv

Złożoność kwadratowa

Liczba instrukcji algorytmu rośnie proporcjonalnie do kwadratu rozmiaru danych

wejściowych. Złożoność dotyczy tych zestawów danych, w przypadku których dla każdej pary danych wejściowych realizuje się określone operacje dominujące.

Przykład algorytmu: sortowanie przez wstawianie, sortowanie bąbelkowe.



Wykładniczą złożonością czasową charakteryzuje się algorytm rozwiązujący zagadkę Wież Hanoi dla n krążków.
Źródło: Contentplus.pl sp. z o.o., CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PLbEkkFv

Złożoność wykładnicza

Czas wykonania rośnie wykładniczo względem rozmiaru danych. W przypadku algorytmów, których złożoność jest złożonością wykładniczą określona liczba operacji dominujących wykonywana jest dla każdego podzbioru danych wejściowych rozmiaru n.

Przykład algorytmu: rozwiązywanie zagadki Wież Hanoi dla n krążków.

Ze złożoności algorytmu wynika jego **efektywność**. Im korzystniejsza jest złożoność czasowa danego algorytmu, tym jest on efektywniejszy.

> **Polecenie 2**
>
> Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w prezentacji multimedialnej.

> **Ciekawostka**
>
> Istnieją algorytmy, których realizacja w pesymistycznym przypadku będzie trwać w nieskończoność. Przykładem jest algorytm sortowania *Bogosort*. Jego istotą jest losowe ustawianie podanych liczb do momentu, w którym będą one uporządkowane rosnąco.

> **Ważne!**
>
> Więcej informacji na temat wspomnianych w prezentacji algorytmów znajdziesz w e-materiałach:
>
> - szybkie potęgowanie liczb: Algorytmy iteracyjne i liczbowe – potęgowanie liczb;
> - obliczanie NWD: Algorytm Euklidesa;
> - wyszukiwanie binarne: Znajdowanie określonego elementu w zbiorze oraz Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”;
> - szukanie maksymalnego oraz minimalnego elementu: Instrukcja warunkowa – ćwiczenia;
> - obliczanie wartości wielomianu za pomocą schematu Hornera: Schemat Hornera;
> - sortowanie przez scalanie: Sortowanie przez scalanie;
> - sortowanie przez wstawianie: Sortowanie przez wstawianie;
> - sortowanie bąbelkowe: Sortowanie bąbelkowe;

- rozwiązywanie zagadki wież Hanoi: Zagadka Wież Hanoi.

Polecenie 3

Podaj przykłady, nie wymienione w tym materiale, algorytmów o złożoności linowej, logarytmicznej i wykładniczej.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Załóżmy, że dwa algorytmy o różnej złożoności czasowej i pamięciowej pozwalają uzyskać poprawne wyniki. Wskaż, który z nich jest lepszym wyborem.

- [ ] Ten o większej złożoności czasowej.
- [ ] Ten o mniejszej złożoności pamięciowej, jeżeli oba mają taką samą złożoność czasową.
- [ ] Ten o mniejszej złożoności czasowej.
- [ ] Ten o większej złożoności pamięciowej, jeżeli oba mają taką samą złożoność czasową.
- [ ] Złożoność czasowa nie ma większego znaczenia.

Ćwiczenie 2

Wskaż, czy kwadratowa złożoność czasowa jest złożonością wielomianową.

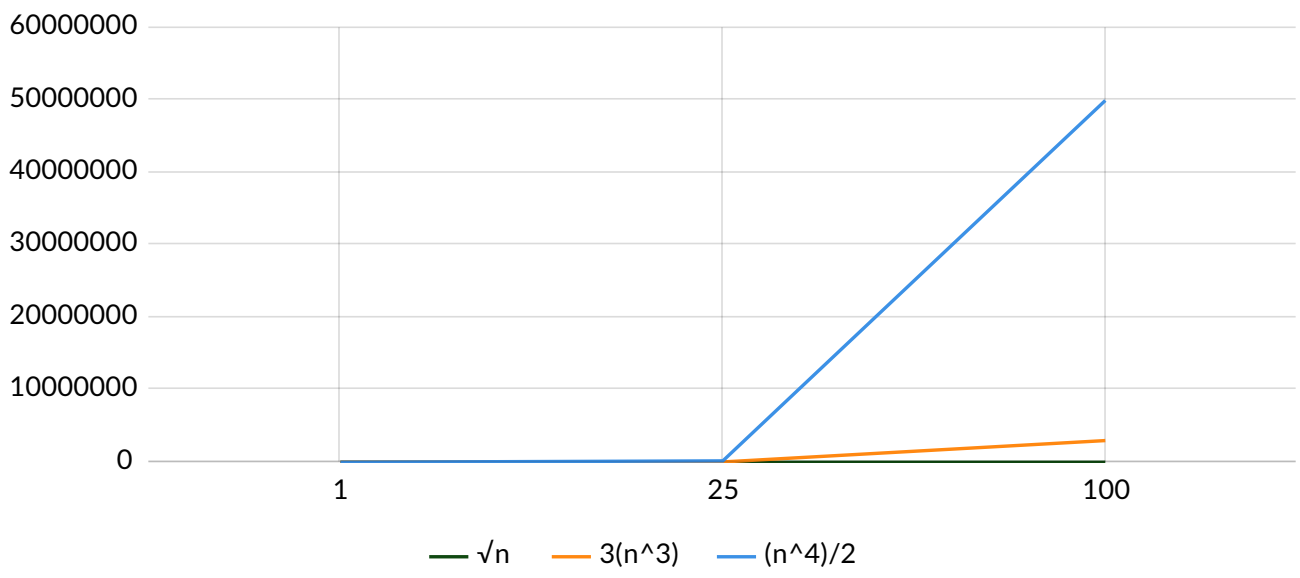
- ( ) tak
- ( ) nie

Ćwiczenie 3

Spośród podanych niżej symboli wybierz ten, który oznacza notację „duże O”.

- ◯ Θ
- ◯ ω
- ◯ o
- ◯ O

Materiał źródłowy do ćwiczeń nr 4–5



Ćwiczenie 4

Dana jest funkcja $f\left(n\right) = \sqrt{n} + \frac{n^4}{2} + 3n^3$. Wypełnij tabelę wartościami, które składniki przyjmują dla danej wartości n. Nie oddzielaj cyfr spacjami i pamiętaj, że części dziesiętne należy oddzielić przecinkiem.

n	$\sqrt{n}$	$\frac{n^4}{2}$	$3n^3$
1			
25			
100			

Ćwiczenie 5

Wskaż, który ze składników funkcji $f\left(n\right) = \sqrt{n} + \frac{n^4}{2} + 3n^3$ ma najmniejszy wpływ na jej tempo wzrostu.

- ○ $\frac{n^4}{2}$
- ○ $3n^3$
- ○ $\sqrt{n}$

Ćwiczenie 6

Połącz w pary funkcje opisujące dokładną liczbę wykonywanych operacji z odpowiadającą im złożonością czasową.

$f(n) = 144$	$O(1)$
$f\left(n\right) = \frac{\log n}{20} + 115$	$O\left(n^2\right)$
$f\left(n\right) = \sqrt{3n} + 4n + 8$	$O(n)$
$f\left(n\right) = \frac{n^2}{4} + 3\sqrt{n} + 10$	$O(n \log n)$
$f(n) = 15 \log n + 5n \log n$	$O(\log n)$

Ćwiczenie 7

Wstaw brakujące wyrażenia tak, aby treść poniższych zdań była prawdziwa.

Złożoność obliczeniowa (czyli złożoność czasowa i pamięciowa) algorytmu w [ ] stopniu decyduje o jego użyteczności.

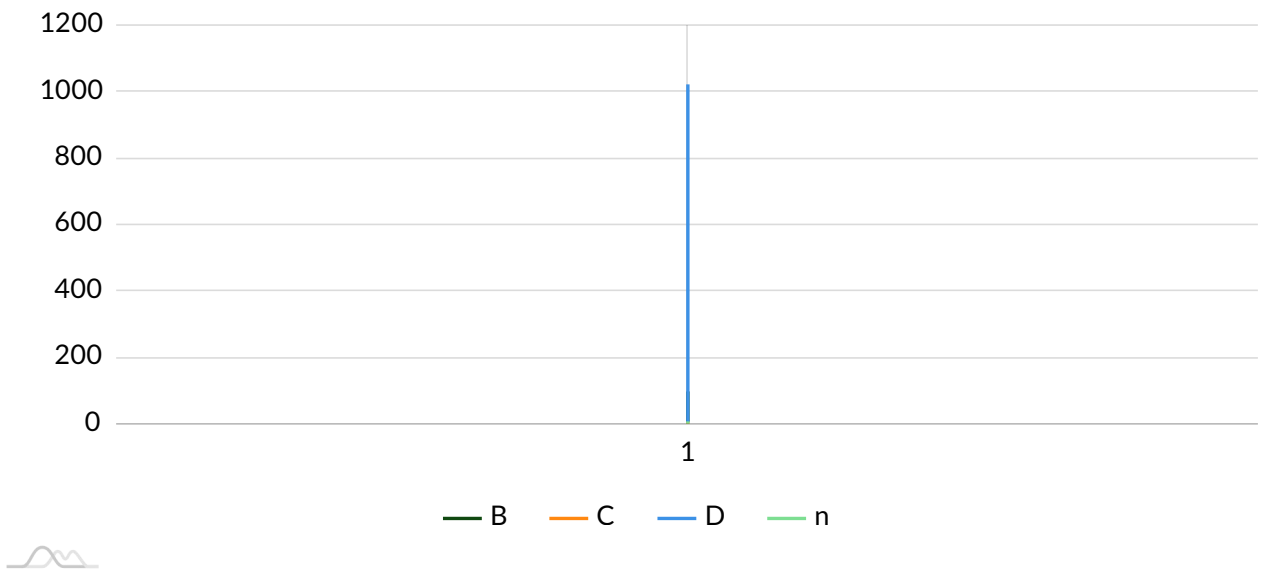
Programy z kwadratową złożonością czasową działają [ ] od tych z liniowo-logarytmiczną.

Jeżeli jesteśmy w stanie zmniejszyć złożoność obliczeniową algorytmu kosztem jego poprawności to [ ].

nie należy tego robić | szybciej | wolniej | dużym | należy to zrobić | małym

Ćwiczenie 8

Zapoznaj się z wykresami przedstawiającymi zależność złożoności obliczeniowej od liczby danych. Legenda do wykresów została sporządzona błędnie. Opisz, w jaki sposób należy ją zmienić, by poprawnie opisywała, do jakiego typu złożoności odnosi się dana wartość na wykresie.



A

B

C

D

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.

Przedmiot: Informatyka

Temat: Złożoność obliczeniowa algorytmów

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wyjaśnisz, czym jest złożoność obliczeniowa algorytmu.
- Scharakteryzujesz, jak określać złożoność obliczeniową algorytmów.
- Przeanalizujesz wybrane rzędy złożoności czasowej.
- Wykonasz ćwiczenia sprawdzające twoją wiedzę z zakresu złożoności obliczeniowej algorytmów.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Złożoność obliczeniowa algorytmów". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć. Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie w parach wykonują polecenie nr 1. Analizują kolejne slajdy. Następnie wykonują polecenie 2 i podają inne, pochodzące z życia codziennego przykłady zdarzeń charakteryzujących się stałą i liniową złożonością czasową.

3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1-6 z sekcji „Sprawdź się”, po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Nauczyciel zadaje pytania podsumowujące, np.
    - czym jest złożoność obliczeniowa?
    - co to jest notacja „duże O”?
    - jak określać złożoność obliczeniową algorytmów?
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 7-8 z sekcji „Sprawdź się”.

Wskazówki metodyczne:

- Nauczyciel może wykorzystać multimedium w sekcji „Przeczytaj” do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Złożoność obliczeniowa algorytmów”.