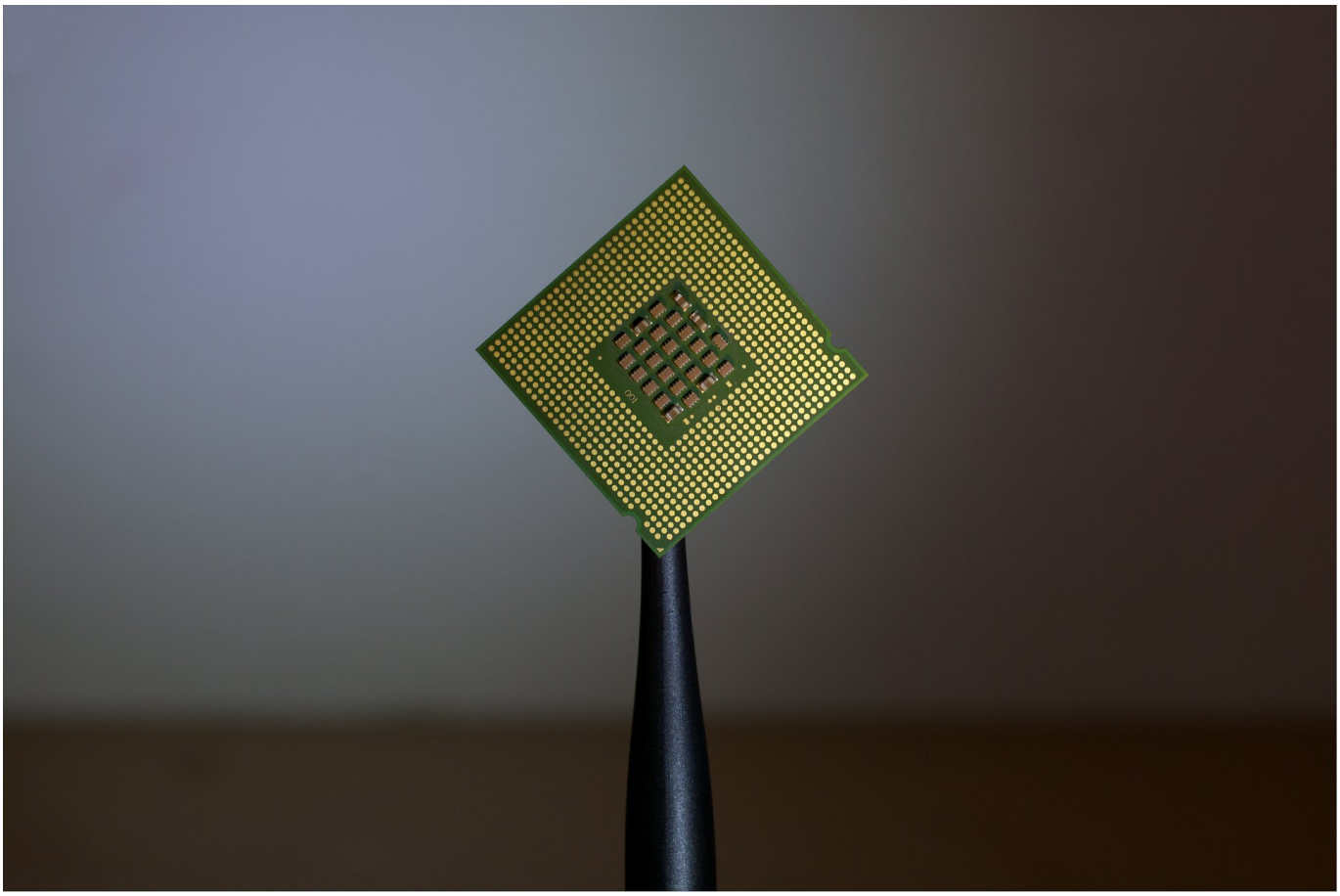


Algorytmy o złożoności logarytmicznej i liniowo-logarytmicznej

Źródło: Brian Kostiuk, domena publiczna.

Algorytmy o złożoności **logarytmicznej** i **liniowo-logarytmicznej** są bardzo często wynikiem optymalizacji czasowej programów o złożoności liniowej bądź kwadratowej. Charakteryzują się one rozbijaniem głównego problemu na mniejsze i tym samym łatwiejsze do rozwiązania **podproblemy**.

Przedstawione podejście pozwala na znaczne zmniejszenie liczby wykonywanych operacji. W rezultacie programy, w których zaimplementowano takie algorytmy, mogą działać na dużych zestawach danych wejściowych.

Więcej informacji o złożoności obliczeniowej znajdziesz w e-materiałach:

- Złożoność obliczeniowa algorytmów,
- Algorytmy o złożoności kwadratowej,
- Algorytmy o złożoności wykładniczej,
- Jak porównywać złożoność obliczeniową?

Twoje cele

- Scharakteryzujesz algorytmy o logarytmicznej i liniowo-logarytmicznej złożoności czasowej.
- Przeanalizujesz przykłady algorytmów o logarytmicznej i liniowo-logarytmicznej złożoności czasowej.

- Porównasz czas działania programów bazujących na algorytmach o złożoności logarytmicznej i liniowo-logarytmicznej oraz programów, w których zaimplementowano algorytmy o złożonościach liniowych i kwadratowych.

Przeczytaj

Zgadywanie wylosowanej liczby

Zacznijmy od pewnej zagadki. Wyobraź sobie, że grasz w grę komputerową i twoim zadaniem jest odgadnięcie, jaką liczbę naturalną z zakresu od 1 do 1000 wylosował komputer. Masz nieograniczoną liczbę prób i za każdym razem otrzymujesz informację zwrotną o tym, czy wybrana przez ciebie liczba jest mniejsza, czy też większa od wylosowanej.

Rozwiązanie naiwne o liniowej złożoności czasowej

Jeżeli mamy nieograniczoną liczbę prób, to pierwszym rozwiązaniem, które przychodzi do głowy, jest podawanie po kolei każdej liczby z zakresu od 1 do 1000. Jednak wymagałoby to od nas dużego zapasu cierpliwości; w najgorszym przypadku musielibyśmy podać kolejno aż tysiąc liczb. Zastanówmy się zatem, jak moglibyśmy ulepszyć nasze rozwiązanie. Może warto skorzystać z otrzymywanej informacji zwrotnej?

Rozwiązanie o logarytmicznej złożoności czasowej

Jeżeli interesuje nas przedział od 1 do 1000, to możemy podać liczbę znajdującą się w jego środku, czyli 500. Załóżmy, że nie trafiliśmy i otrzymaliśmy informację zwrotną, zgodnie z którą poszukiwana liczba jest mniejsza od podanej. Wiemy wtedy, że znajduje się ona gdzieś w przedziale $\langle 1, 499 \rangle$. W tak prosty sposób zmniejszyliśmy przeszukiwany obszar o połowę! Możemy za każdym razem podawać liczbę ze środka interesującego nas przedziału i dzięki informacji zwrotnej zmniejszać go, aż w końcu trafimy na poszukiwaną liczbę.

Polecenie 1

Zapisz w postaci listy kroków opisany algorytm, przy założeniu, że komputer wylosował liczbę 265.

Lista kroków:

Polecenie 2

Porównaj swoje rozwiązanie z prezentacją.

1

Zakres poszukiwań: $\langle 1,\ 1000\rangle$.

Podajemy wartość 500 i otrzymujemy informację, że poszukiwana liczba jest od niej mniejsza.

2

Zakres poszukiwań: $\langle 1,\ 499\rangle$.

Podajemy wartość 250 i otrzymujemy informację, że poszukiwana liczba jest od niej większa.

3

Zakres poszukiwań: $\langle 251,\ 499\rangle$.

Podajemy wartość 375 i otrzymujemy informację, że poszukiwana liczba jest od niej mniejsza.

4

Zakres poszukiwań: $\langle 251,\ 374\rangle$.

Podajemy wartość 312 i otrzymujemy informację, że poszukiwana liczba jest od niej mniejsza.

5

Zakres poszukiwań: $\langle 51,\ 311\rangle$.

Podajemy wartość 281 i otrzymujemy informację, że poszukiwana liczba jest od niej mniejsza.

6

Zakres poszukiwań: $\langle 251,\ 280 \rangle$.

Podajemy wartość 265 i otrzymujemy informację, że odgadliśmy poszukiwaną liczbę.

Opisany wyżej algorytm to tzw. wyszukiwanie binarne. Dzięki niemu znaleźliśmy wylosowaną liczbę już po 6 próbach. Wykorzystując informację zwrotną przekazywaną przez program, byliśmy w stanie sprowadzić oryginalny problem do mniejszych i łatwiejszych do rozwiązania **podproblemów**. Technika ta nosi nazwę **„dziel i zwyciężaj”**; zajmiemy się nią dokładniej w kolejnych e-materiałach. Teraz udowodnimy, że takie podejście ma logarytmiczną złożoność czasową.

Przejdźmy zatem do nieco bardziej ogólnego przypadku. Przeszukujemy zbiór n liczb naturalnych i po każdej próbie zmniejszamy jego rozmiar o połowę. Zatem w i-tym kroku będziemy mieli do czynienia ze zbiorem o wielkości $\frac{n}{2^i}$. W najgorszym możliwym przypadku powtórzymy tę sama procedurę k razy, aż dojdziemy do zbioru jednoelementowego. Policzmy zatem, ile wynosi k.

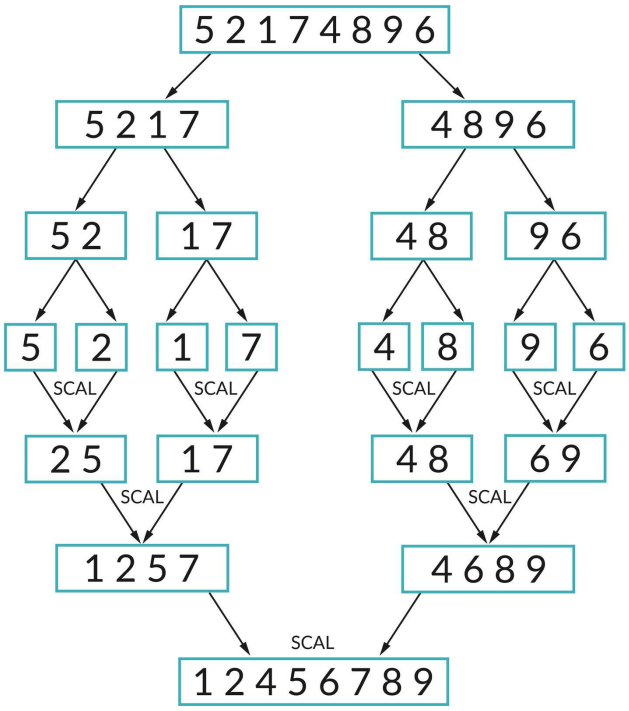
$$\left\lfloor \frac{n}{2^k} \right\rfloor = 1 \Rightarrow 2^k \approx n \Rightarrow k \approx \lfloor \log_2 n \rfloor$$

W najbardziej pesymistycznym przypadku wykonanych zostanie $\log_2 n$ iteracji. Algorytm ma zatem logarytmiczną złożoność czasową.

Sortowanie z wykorzystaniem techniki „dziel i zwyciężaj”

Podczas omawiania złożoności kwadratowej spotkaliśmy się z problemem sortowania n liczb. Okazuje się, że do jego rozwiązania również możemy wykorzystać metodę „dziel i zwyciężaj”. Uzyskamy dzięki temu algorytm rekurencyjny o złożoności czasowej $O(n \log_2 n)$, nazywany sortowaniem przez scalanie (ang. *merge sort*).

W przypadku opisywanego algorytmu ciąg wejściowy dzieli się na dwa mniejsze ciągi o podobnej liczbie elementów, sortuje je niezależnie od siebie, a następnie scala się je z powrotem. Rozbija się w ten sposób główny problem na mniejsze podproblemy, które traktuje się tak samo.



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

Sortowaniu przez scalanie poświęcimy osobny e-materiał, więc nie będziemy na razie zagłębiać się w szczegóły tego mechanizmu porządkowania zbiorów. Wspominamy o nim teraz, ponieważ jest on dobrym przykładem algorytmu o liniowo-logarytmicznej złożoności czasowej. Wynika to z faktu, że operacja scalania jest liniowa i będziemy ją wykonywać $\log_2 n$ razy.

Przybliżony czas realizacji algorytmów o złożoności logarytmicznej i liniowo-logarytmicznej

Porównajmy czas wykonywania operacji składających się na algorytmy o logarytmicznej i liniowo-logarytmicznej złożoności czasowej z czasem realizacji algorytmów o złożonościach poznanych poprzednio:

Ilość danych wejściowych	**Czas relizacji algorytmu** $O(\log_2 n)$	**Czas realizacji algorytmu** $O(n)$	**Czas realizacji algorytmu** $O(n \log_2 n)$	**Czas realizacji algorytmu** $O(n^2)$
10^2	10^{-9} sekundy	10^{-7} sekundy	10^{-7} sekundy	10^{-5} sekundy
10^4	10^{-8} sekundy	10^{-5} sekundy	10^{-4} sekundy	$0,1$ sekundy
10^6	10^{-8} sekundy	10^{-3} sekundy	10^{-2} sekundy	1000 sekund ≈ 17 minut
10^8	10^{-8} sekundy	$0,1$ sekundy	2 sekundy	10^7 sekund ≈ 115 dni

Ilość danych wejściowych	**Czas relizacji algorytmu** $O(\log_2 n)$	**Czas realizacji algorytmu** $O(n)$	**Czas realizacji algorytmu** $O(n \log_2 n)$	**Czas realizacji algorytmu** $O(n^2)$
10^{10}	10^{-8} sekundy	10 sekund	330 sekund $\approx$ 5 minut	10^{11} sekund ≈ 3168 lat
10^{12}	10^{-8} sekundy	1000 sekund $\approx$ 17 minut	40000 sekund $\approx$ 11 godzin	10^{15} sekund $\approx 31,6$ mln lat

Z przedstawionego zestawienia wynika, że algorytmy o złożoności logarytmicznej są wykonywane niezwykle szybko nawet dla dużych ilości danych wejściowych. Natomiast algorytmy o złożoności liniowo-logarytmicznej są realizowane o wiele szybciej od algorytmów o złożonościach kwadratowych (ale zarazem wolniej od algorytmów liniowych).

Podsumowując, algorytmy logarytmiczne i liniowo-logarytmiczne charakteryzują się rozbijaniem głównego problemu na mniejsze podproblemy. Jest to dobra metoda optymalizacji czasowej, o której warto pamiętać w trakcie pisania własnych programów.

> **Ciekawostka**
>
> Istnieje złożoność czasowa $O(\log^* n)$, potocznie nazywana „logarytmem z gwiazdką”. Pod tym pojęciem kryje się odwrotna funkcja Ackermanna, która rośnie niewyobrażalnie powoli. Bardzo często traktuje się $\log^* n$ jako stałą o wartości równej 5.

Słownik

> **metoda „dziel i zwyciężaj”**
>
> metoda, która pozwala na podział problemu na mniejsze, łatwiejsze do rozwiązania podproblemy; po całkowitym rozbiciu problemu oraz rozwiązaniu podproblemów, łączymy je z powrotem w całość i rozwiązujemy cały dany nam problem
> **rekurencja**
>
> technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego
> **sortowanie przez scalanie**
>
> jeden z algorytmów sortowania bazujący na metodzie „dziel i zwyciężaj”. Algorytm możemy podzielić na dwa główne etapy:
>
> - podzielenie nieposortowanej listy na n list podrzędnych, z których każda zawiera jeden element;
> - łączenie kolejnych podlist, aby utworzyć nowe posortowane podlisty, aż pozostanie tylko jedna lista

wyszukiwanie binarne

metoda odnajdowania elementów w uporządkowanych zbiorach; polega ona na wielokrotnym dzieleniu przeszukiwanego zbioru na dwie równe części i porównywaniu wartości szukanej z elementem znajdującym się w środku dzielonego obszaru

złożoność czasowa

czas niezbędny do rozwiązania określonego problemu, podawany zazwyczaj w liczbach operacji dominujących (zależy ona od ilości danych wejściowych)

Aplet

Polecenie 1

Wyszukiwanie binarne to algorytm służący do znajdowania pozycji elementu w **posortowanej tablicy**. Zamiast klasycznego sprawdzania kolejnych elementów tablicy, stosowana jest metoda „dziel i zwyciężaj”. Procedurę można opisać jako listę następujących czynności:

1. Sprawdź element znajdujący się w środku zbioru.
2. Jeśli sprawdzany element jest równy elementowi wyszukiwanemu, zwróć jego pozycję.
3. Jeśli sprawdzany element jest większy od szukanego, przeprowadź wyszukiwanie binarne w pierwszej połowie tablicy.
4. Jeśli sprawdzany element jest mniejszy od szukanego, przeprowadź wyszukiwanie binarne w drugiej połowie tablicy.
5. Wykonuj przedstawione czynności do momentu, w którym znajdziesz szukany element lub zabraknie nowych elementów do sprawdzenia.

Średnia złożoność czasowa wyszukiwania binarnego to $O(\log n)$, ponieważ $\log n$ to maksymalna liczba podziałów tablicy o n elementach.

Uruchom aplet przedstawiający procedurę wyszukiwania binarnego. Wpisz wartość, która ma zostać wyszukana i naciśnij przycisk `Wyszukaj`. Sprawdź, jak program wykorzystujący algorytm wyszukiwania binarnego zachowa się dla różnych danych wejściowych.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DwY8eJarE

Polecenie 2

Uruchom aplet przedstawiający sortowanie przez kopcowanie (ang. *heapsort*). Jest to algorytm sortowania, którego złożoność czasowa w średnim i pesymistycznym przypadku wynosi $O(n \log n)$, gdzie n to liczba danych do posortowania. Algorytm dzieli dane wejściowe na posortowaną i nieposortowaną część. Iteracyjne pobiera i usuwa z nieposortowanej części największy element, a następnie wstawia go do posortowanej części.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DwY8eJarE

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w apletach.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż, jaką złożoność obliczeniową ma algorytm wyszukiwania binarnego.

- ◯ Złożoność liniowo-logarytmiczną.
- ◯ Złożoność kwadratową.
- ◯ Złożoność logarytmiczną.
- ◯ Złożoność liniową.

Ćwiczenie 2

Wskaż poprawną definicję algorytmu rekurencyjnego.

- ◯ Algorytm, który rozwiązując dany problem, wywołuje inne algorytmy (jeden lub więcej razy).
- ◯ Algorytm, który rozwiązując dany problem, korzysta z podstawowych struktur danych.
- ◯ Algorytm, który rozwiązując dany problem, wywołuje sam siebie (jeden lub więcej razy).
- ◯ Algorytm, który rozwiązując dany problem, generuje wszystkie możliwe odpowiedzi, a następnie sprawdza, które z nich są poprawne.

Ćwiczenie 3

Wskaż, jaką złożoność obliczeniową ma algorytm sortowania przez scalanie.

- ◯ Złożoność logarytmiczną.
- ◯ Złożoność liniową.
- ◯ Złożoność liniowo-logarytmiczną.
- ◯ Złożoność kwadratową.

Ćwiczenie 4

Wskaż, ile czasu będzie wykonywany ten algorytm w sytuacji, gdy $n = 10^9$? Niech funkcja f, opisująca liczbę operacji składających się na algorytm w zależności od ilości danych wejściowych, ma postać: $f(n) = n \log_{10} n$. Przyjmij, że komputer wykonuje 10^9 operacji w ciągu 1 sekundy.

- ◯ Około 9 minut.
- ◯ Około 9 dni.
- ◯ Około 9 sekund.
- ◯ Około 9 godzin.

Ćwiczenie 5

Wskaż, czy algorytm ten ma złożoność logarytmiczną, wiedząc, że funkcja f, opisująca liczbę operacji składających się na algorytm w zależności od ilości danych wejściowych, ma następującą postać: $f(n) = \log_2(n) + n \log_2(n)$.

- ◯ Tak, ma on złożoność logarytmiczną.
- ◯ Nie, nie ma on złożoności logarytmicznej.

Ćwiczenie 6

Wstaw brakujące wyrażenia tak, aby treść poniższego tekstu była prawdziwa.

Algorytmy o złożoności logarytmicznej i liniowo-logarytmicznej bardzo często korzystają z metody [ ]. Polega ona na podziale głównego problemu na mniejsze [ ]. Algorytmy o złożoności liniowo-logarytmicznej są wykonywane szybciej od [ ], ale za to wolniej od [ ].

„łącz i zwyciężaj" | kwadratowych | „dziel i zwyciężaj" | podproblemy | nadproblemy | liniowych

Ćwiczenie 7

Uporządkuj poniższe złożoności obliczeniowe od najszybszej do najwolniejszej dla dużej ilości danych wejściowych.

złożoność liniowo-logarytmiczna

złożoność stała

złożoność liniowa

złożoność kwadratowa

złożoność logarytmiczna

Ćwiczenie 8

Połącz w pary algorytmy z odpowiadającymi im złożonościami obliczeniowymi.

dodawanie dwóch liczb całkowitych	złożoność logarytmiczna
sortowanie przez scalanie	złożoność stała
wyszukiwanie najmniejszej liczby w zbiorze	złożoność liniowa
wyszukiwanie binarne	złożoność liniowo-logarytmiczna

Dla nauczyciela

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

Temat: Algorytmy o złożoności logarytmicznej i liniowo-logarytmicznej

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 podstawowy. Uczeń:

3) wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

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

4) ilustruje i wyjaśnia rolę pojęć, obiektów i operacji matematycznych w projektowaniu rozwiązań problemów informatycznych i z innych dziedzin, posługuje się pojęciem logarytmu;

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

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

a) wyszukiwanie elementów liniowe i przez połowienie (do znajdowania elementów w zbiorze, sortowania przez wstawianie, przybliżonego rozwiązywania równań, sprawdzania przynależności punktu do wielokąta wypukłego),

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

c) metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),

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):

- Scharakteryzujesz algorytmy o logarytmicznej i liniowo-logarytmicznej złożoności czasowej.
- Przeanalizujesz przykłady algorytmów o logarytmicznej i liniowo-logarytmicznej złożoności czasowej.
- Porównasz czas działania programów bazujących na algorytmach o złożoności logarytmicznej i liniowo-logarytmicznej oraz programów, w których zaimplementowano algorytmy o złożonościach liniowych i kwadratowych.

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 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: „Algorytmy o złożoności logarytmicznej i liniowo-logarytmicznej". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat oraz cele zajęć, omawiając lub ustalając razem z uczniami 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.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające prosi o ciche zapoznanie się z treścią w sekcji „Przeczytaj".
2. **Praca z multimedium.** Uczniowie w zespołach dwuosobowych zapoznają się z treścią polecenia 1 z sekcji „Aplet" i wspólnie analizują kolejne kroki rozwiązania postawionego problemu.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia 1-8 z sekcji „Sprawdź się". Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują polecenie 2 z sekcji „Aplet".

Wskazówki metodyczne:

- Treści w sekcji „Aplet” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.