Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”

Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”

Źródło: Headway, domena publiczna.

Strategię „dziel i zwyciężaj” wykorzystujemy w sytuacji, gdy chcemy, aby zastosowany algorytm miał możliwie małą złożoność czasową. Metoda ta polega na dzieleniu problemu na mniejsze części, by jak najefektywniej rozwiązać problem główny.

Przykładami algorytmów sortowania opartych na strategii „dziel i zwyciężaj” są np. sortowanie szybkie (*quick sort*) oraz sortowanie przez scalanie (*merge sort*), wyszukiwanie binarne.

Więcej wskazówek dotyczących rozwiązywania problemów informatycznych znajdziesz w e-materiałach:

- Projektowanie algorytmów: metody wstępująca i zstępująca,
- Jak myśleć komputacyjnie?,
- Rozwiąż to inaczej, czyli myślenie komputacyjne i jego metody.

Twoje cele

- Prześledzisz etapy realizacji strategii „dziel i zwyciężaj”.
- Przeanalizujesz przykładowe algorytmy, w których zastosowano strategię „dziel i zwyciężaj”.
- Wyjaśnisz, w jakich okolicznościach warto zastosować metodę „dziel i zwyciężaj”.

- Przeanalizujesz złożoność czasową algorytmu opartego na strategii „dziel i zwyciężaj”.
- Przeanalizujesz algorytm jednoczesnego wyszukiwania minimum i maksimum.

Przeczytaj

Metoda „dziel i zwyciężaj”

Strategię „dziel i zwyciężaj” można podzielić na trzy etapy:

- Podziel problem na podproblemy.
- Znajdź rozwiązanie dla każdego podproblemu.
- Połącz rozwiązania podproblemów w rozwiązanie problemu głównego.

Należy pamiętać, że problem musi zostać podzielony na takie same lub bardzo do siebie podobne podproblemy (przynajmniej dwa). Ponadto zbiory danych, dla których rozwiązywane są podproblemy, powinny być niemal jednakowe i stanowić stałą część całego zbioru danych.

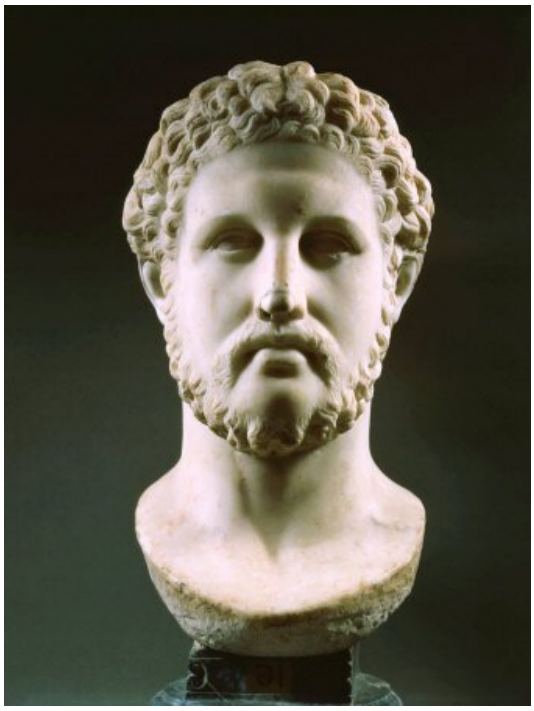
> **Ważne!**
>
> Należy zadbać o poprawne zaprojektowanie trzeciego kroku, polegającego na scalaniu rozwiązań. Jeżeli będzie on nieoptymalny, niekorzystnie wpłynie na końcową złożoność czasową algorytmu.

Wykorzystanie

Omawiana metoda wykorzystywana jest między innymi w algorytmach:

- wyszukiwania binarnego,
- sortowania przez scalanie,
- znajdowania w ciągu największego i najmniejszego elementu jednocześnie,
- sortowania szybkiego.

> **Ciekawostka**
>
> Zasada „dziel i rządź” (łac. *divide et impera*) mówi o tym, by wzniecać wewnętrzne konflikty wśród wrogów, np. między ludami na podbitych terenach, aby nimi rządzić.
>
> Jej autorstwo przypisuje się Filipowi II Macedońskiemu, ojcu Aleksandra Wielkiego. Stosowali ją również Rzymianie (między innymi Juliusz Cezar).



Filip II Macedoński

Źródło: dostępny w internecie: commons.wikimedia.org [dostęp 21.07.2022], domena publiczna.

Najeźdźcy podpisywali umowy z podbitymi ludami, które odtąd nazywano sprzymierzeńcami. Warunkiem paktów było to, że „sprzymierzeńcy" nie mogli zawierać umów między sobą, przez co prowincje nie mogły się zjednoczyć, aby pokonać Rzymian (czy też innych najeźdźców).

Przykłady zastosowania

Wyszukiwanie binarne

Wyszukiwanie binarne jest metodą szybkiego odnajdowania danych w posortowanym ciągu liczbowym. Jego czasowa złożoność obliczeniowa wynosi $O(\log_2 n)$. Czym jednak różni się ono od przeglądania tablicy metodą liniową element po elemencie i porównywania kolejnych liczb z poszukiwaną wartością?

W przypadku wyszukiwania binarnego wybierany jest element środkowy, względem którego ciąg jest dzielony na dwie części. Sprawdzamy, czy środkowy element jest poszukiwanym elementem. Jeśli nie, sprawdzamy, czy środkowy element jest większy, czy mniejszy od poszukiwanego elementu. Zależnie od tego wybieramy tylko lewy albo prawy podciąg, czyli działamy wyłącznie na jednej z wydzielonych części.

Zaprezentujmy działanie algorytmu na konkretnym przykładzie.

Dany jest następujący posortowany ciąg liczb całkowitych:

```
1 [1,2,5,9,12,15,27,50,94,100]
```

- Lewą najmniejszą liczbą w tym ciągu jest liczba 1 (indeks 0).
- Prawą najmniejszą liczbą w tym ciągu jest liczba 100 (indeks 9).
- Poszukujemy pozycji liczby 27.

Chcemy wyznaczyć indeks elementu środkowego. Dodajemy do siebie indeksy lewej i prawej najmniejszej liczby w tym ciągu – otrzymujemy liczbę 9. Dzielimy ją całkowicie przez 2. Otrzymujemy wynik 4.

Pod indeksem 4 znajduje się liczba 12. Szukana liczba jest od niej większa, zatem wiemy, że liczby o indeksach 4 i mniejszych nie będą już brane pod uwagę.

Zostaje podciąg składający się z liczb większych od 12:

```
[15,27,50,94,100]
```

Lewą najmniejszą liczbą w tym ciągu zostaje liczba o indeksie 5, czyli liczba 15.

Prawą najmniejszą liczbą w tym ciągu zostaje liczba 100 o indeksie 9.

Ponownie dodajemy do siebie wartości indeksów i dzielimy je całkowicie przez 2. Otrzymany wynik to 7.

Liczbą o indeksie równym 7 jest 50.

Poszukiwana liczba 27 jest od niej mniejsza, więc szukany element może się znajdować na miejscach o indeksach od 5 do 7.

Teraz analizowany podciąg wygląda następująco:

```
[15,27,50]
```

Ponawiamy dodawanie do siebie indeksów. Tym razem środkiem zostaje liczba o indeksie 6. Jest to liczba 27, czyli poszukiwana wartość.

Zapiszmy analizowany algorytm za pomocą pseudokodu:

```
funkcja wyszukiwanie_binarne(tablica, szukana, pocz, kon):
    jeżeli pocz > kon:
        zwróć -1

    środek ← (pocz + kon) div 2
    jeżeli tablica[środek] = szukana:
        zwróć środek
    w przeciwnym wypadku jeżeli tablica[środek] < szukana:
        zwróć wyszukiwanie_binarne(tablica, szukana, środek + 1,
    w przeciwnym wypadku:
```

```
11          zwróć wyszukiwanie binarne(tablica, szukana, pocz, środek
```

Przedstawiliśmy rozwiązanie rekurencyjne, ale możliwe jest również zapisanie tego algorytmu w postaci iteracyjnej.

Więcej o wyszukiwaniu binarnym możesz przeczytać w e-materiale Znajdowanie określonego elementu w zbiorze. Znajduje się tam również wersja iteracyjna algorytmu. Zachęcamy do porównania obydwu zapisów.

Złożoność czasowa

Aby zrozumieć złożoność czasową algorytmu wyszukiwania binarnego, warto przeanalizować jego działanie. W każdym kroku algorytmu dzielimy badaną tablicę na pół, odrzucając jedną połowę i kontynuując szukanie w drugiej.

Przykładowo, jeżeli tablica miałaby osiem elementów, to po pierwszym kroku analizowalibyśmy już tylko cztery elementy, po drugim dwa, a po trzecim mielibyśmy już do dyspozycji tylko jeden element.

Dla tablicy 16-elementowej zawężenie wyszukiwania do jednego elementu zajęłoby cztery kroki.

Algorytm wykonuje logarytmiczną liczbę operacji w stosunku do liczby danych wejściowych. Jeżeli tablica ma `n` elementów, potrzebna liczba kroków wynosi `log n`, ponieważ w każdym kroku dzielimy tablicę na połowę.

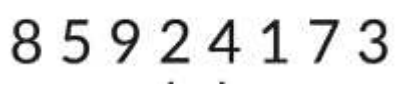
Stąd złożoność czasowa algorytmu wyszukiwania binarnego wynosi:

$$O(\log_2 n)$$

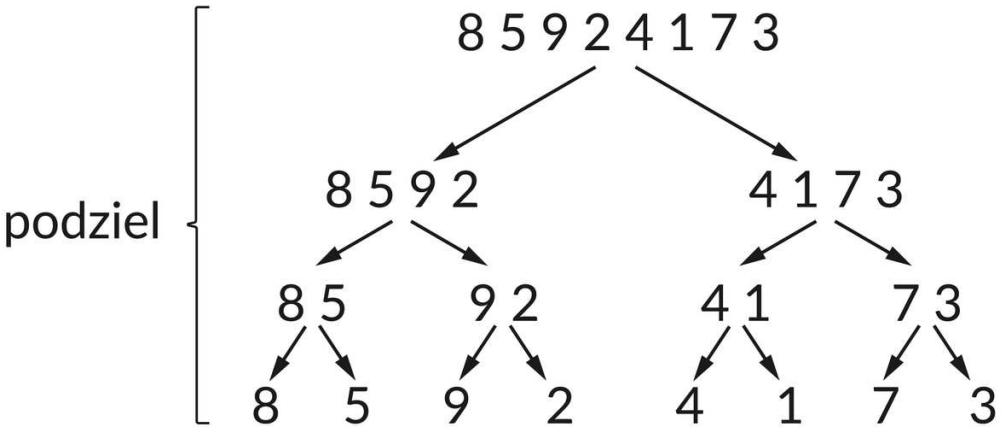
Sortowanie przez scalanie

Przykładem wykorzystania metody „dziel i zwyciężaj” (uwaga: nie myl jej z zasadą „dziel i rządź”!) jest algorytm sortowania przez scalanie. Więcej na jego temat znajdziesz w e-materiale Sortowanie przez scalanie.

Naszym zadaniem jest posortowanie niemalejąco następującego zbioru danych:

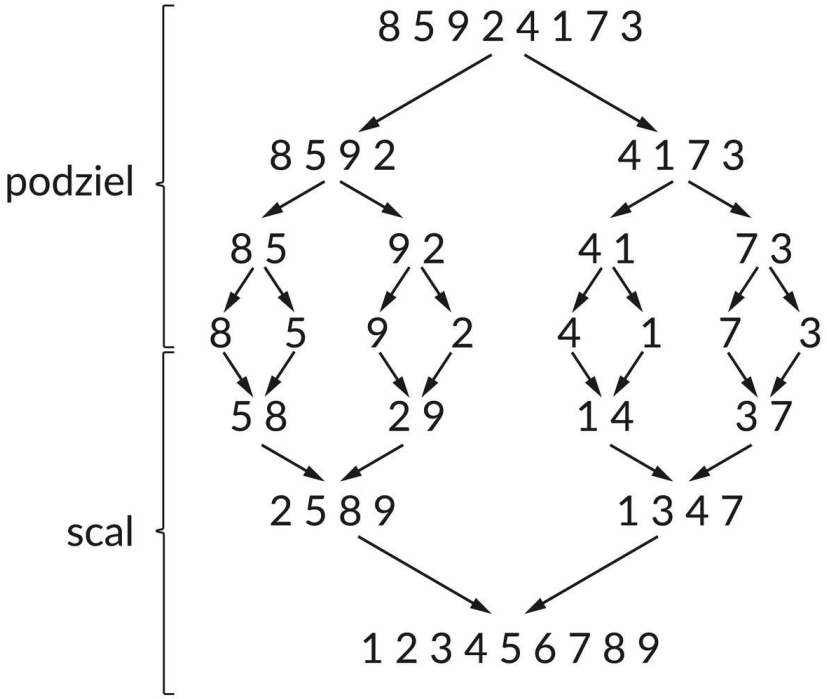


Najpierw należy podzielić tablicę na mniejsze, osobno sortowane tablice. W tym celu tablicę dzielimy na pół – jeżeli liczba elementów jest nieparzysta, pierwsza tablica będzie zawierała o jeden element więcej. Konieczne jest też wyznaczenie elementu środkowego, który stanowi podstawę podziału. Proces powtarzamy tak długo, aż uzyskamy tablice jednoelementowe. Jest to etap dzielenia (**dziel**).



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

Wiemy, że tablica zawierająca jeden element już jest posortowana. Dzięki temu możemy scalić dwie tablice jednoelementowe i uzyskać kolejną posortowaną tablicę dwuelementową. Następnie scalamy dwie posortowane tablice dwuelementowe, aby uzyskać tablicę czteroelementową i tak dalej. Jest to etap scalania (**zwyciężaj**).



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

Szczegółowe omówienie sortowania przez scalanie oraz funkcji `scal` znajduje się w e-materiale Sortowanie przez scalanie.

Złożoność czasowa

Biorąc pod uwagę przedstawiony przykład zastosowania metody „dziel i zwyciężaj", jakim jest sortowanie przez scalanie, przeanalizujemy teraz złożoność czasową algorytmu.

Załóżmy, że tablica, którą mamy posortować, ma rozmiar n. W każdym kolejnym kroku algorytmu (czyli w każdym kolejnym wywołaniu rekurencyjnym funkcji) dzielimy ją na dwie części. W wypadku parzystej liczby elementów są to zawsze dwie równe części.
W przypadku nieparzystej liczby elementów pierwsza część zawiera tylko o jeden element więcej od drugiej. Zatem dla n-elementowej tablicy mamy$\log_2(n)$ poziomów zagnieżdżenia funkcji. Jest to równocześnie złożoność czasowa tej operacji. Jest to etap „dziel".

W każdym kroku szukamy indeksu, który stanowi połowę obecnego łańcucha. Ten krok wykonujemy w czasie $O(1)$, a zatem nie zwiększa on złożoności obliczeniowej.

Pod koniec algorytmu scalamy mniejsze części tablicy w jedną posortowaną. Oryginalna tablica ma rozmiar n, zatem scalenie wszystkich części, które sumarycznie mają rozmiar n, zajmie $O(n)$ czasu. Jest to część „zwyciężaj" omawianej metody.

Algorytm sortowania przez scalanie w każdym przypadku (zarówno pesymistycznym, jak i optymistycznym) ma następującą złożoność czasową:

$$O(n \cdot \log_2 n)$$

Porównanie złożoności czasowych

W przypadku wyszukiwania binarnego jedynym wykonywanym działaniem jest dzielenie w każdym kroku tablicy na pół. Stąd złożoność czasowa tego algorytmu jest logarytmiczna. W przeciwieństwie do sortowania przez scalanie, już po wykonaniu kroku „dziel" z metody „dziel i zwyciężaj" mamy odpowiedź.

W sortowaniu przez scalanie, po podzieleniu tablic na połowy, należy je jeszcze scalić. Łączna długość tablicy wynosi n, stąd scalenie wszystkich tablic zajmie $O(n)$. Gdy połączymy to z dzieleniem tablicy na połowy (złożoność logarytmiczna), otrzymamy złożoność $O(n \cdot \log_2 n)$.

Algorytm jednoczesnego wyszukiwania minimum i maksimum

Przykładem algorytmu typu „dziel i zwyciężaj" jest algorytm równoczesnego wyszukiwania w zbiorze największej i najmniejszej wartości.

Algorytm jednoczesnego wyszukiwania wartości minimalnej i maksymalnej, który wykorzystuje strategię dziel i zwyciężaj, składa się z następujących kroków:

1. Sprawdź rozmiar tablicy. Jeśli rozmiar wynosi 1, zwróć element jako minimum i maksimum.
2. Podziel tablicę na dwie równe części.
3. Rekurencyjnie wywołaj algorytm dla dwóch połówek tablicy.
4. W wyniku wywołania rekurencyjnego otrzymasz wartości minimalne i maksymalne dla każdej połówki.
5. Porównaj wartości minimalne i maksymalne dla obu połówek i zwróć wyniki.
6. Porównaj wartości minimalne i maksymalne z obu połówek i wybierz mniejszą wartość jako minimum całej tablicy oraz większą wartość jako maksimum całej tablicy.
7. Zwróć minimum i maksimum jako wynik.

Kroki 2-5 są wykonywane rekurencyjnie dla każdej połówki tablicy aż do momentu, gdy rozmiar tablicy wynosi 1. W tym przypadku algorytm zwraca jedyny element tablicy jako zarówno minimum, jak i maksimum.

Algorytm wykorzystuje strategię „dziel i zwyciężaj", dzieląc problem na mniejsze części. Następnie łączy wyniki z mniejszych problemów, aby uzyskać wynik dla całego problemu. W tym przypadku dzieli tablicę na dwie równe części, znajduje wartości minimalne i maksymalne dla każdej połowy, a następnie porównuje je, aby znaleźć wartości minimalne i maksymalne dla całej tablicy.

Pseudokod algorytmu:

```
funkcja ZnajdzMinMax(tablica, pocz, kon)
    jeżeli kon = pocz
        min ← tablica[pocz]
        max ← tablica[pocz]
        zwróć (min, max)
    jeżeli kon = pocz + 1
        jeżeli tablica[pocz] <= tablica[kon]
            min ← tablica[pocz]
            max ← tablica[kon]
        w przeciwnym razie
            min ← tablica[kon]
            max ← tablica[pocz]
        zwróć (min, max)

    polowa ← (pocz + kon) // 2

    (minLewa, maxLewa) ← ZnajdzMinMax(tablica, pocz, polowa)
    (minPrawa, maxPrawa) ← ZnajdzMinMax(tablica, polowa + 1, kon)

```

```
20      min ← min(minLewa, minPrawa)
21      max ← max(maxLewa, maxPrawa)
22
23      zwróć (min, max)
```

Słownik

iteracja

(z łac. *iteratio*) powtarzanie w pętli tych samych instrukcji aż do spełnienia pewnego warunku

metoda „dziel i zwyciężaj”

metoda polegająca na podzieleniu problemu na mniejsze, prostsze do rozwiązania podproblemy; po całkowitym rozbiciu problemu oraz rozwiązaniu podproblemów, łączymy je z powrotem w całość i rozwiązujemy cały problem

rekurencja

technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego

rozwiązanie naiwne

najprostsze, bezpośrednie rozwiązanie danego problemu, pozbawione jakichkolwiek usprawnień i optymalizacji, przez co zazwyczaj jest to rozwiązanie nieoptymalne; przykładami mogą być algorytmy, które sprawdzają wszystkie możliwe rozwiązania (podczas gdy wystarczyłoby sprawdzenie tylko części), przeszukują całe zbiory danych (podczas gdy wystarczyłoby tylko częściowe przeszukiwanie) itp.

sortowanie przez scalanie

jeden z algorytmów sortowania bazujący na metodzie „dziel i zwyciężaj”; możemy w nim wyróżnić 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, polegająca 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; złożoność czasowa zależy od ilości danych wejściowych

Symulacja interaktywna

Symulacja 1

Przeanalizuj symulację interaktywną, która wizualizuje metodę „dziel i zwyciężaj” na przykładzie wyszukiwania binarnego.

W symulacji możesz wprowadzić własną listę (pamiętaj, by była posortowana) oraz szukany element. Symulacja ilustruje, w jaki sposób z listy liczb wyodrębniane są coraz mniejsze listy, zgodnie z metodą „dziel i zwyciężaj”.

Symulacja obrazująca wyszukiwanie binarne

Polecenie 1

Zaproponuj własne dane do wprowadzenia do symulacji. Podaj liczbę kroków dla tych danych.

Polecenie 2

Zapisz algorytm wyszukiwania binarnego. Wykorzystaj listę kroków.

Polecenie 3

Zastanów się, do rozwiązania jakich problemów z życia codziennego możesz wykorzystać sortowanie przez scalanie.

Polecenie 4

Zapisz, jakie problemy z życia codziennego można rozwiązać za pomocą wyszukiwania binarnego.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zaznacz wszystkie poprawne zakończenia zdania.
Dzięki rozbijaniu kodu programu na funkcje staje się on...

- [ ] gorszy.
- [ ] poprawny w działaniu.
- [ ] bardziej uniwersalny.
- [ ] łatwiejszy do zrozumienia.

Ćwiczenie 2

Zdecyduj, czy zdanie jest prawdziwe.
Problem dzielimy zawsze na dwa podproblemy.

- ( ) fałsz
- ( ) prawda

Ćwiczenie 3

Uporządkuj w odpowiedniej kolejności trzy główne etapy metody „dziel i zwyciężaj”.

1. Znajdź rozwiązanie dla każdego podproblemu.
2. Połącz rozwiązania podproblemów w rozwiązanie problemu głównego.
3. Podziel problem na podproblemy.

Ćwiczenie 4

Wskaż, jaką złożoność czasową ma algorytm wyszukiwania binarnego.

- $O(\log_3 n)$
- $O(n \log_2 n)$
- $O(n)$
- $O(\log_2 n)$

Ćwiczenie 5

Dokończ zdanie.

Jeżeli w trakcie wyszukiwania binarnego wybierzemy liczbę ze środka ciągu posortowanego nierosnąco i okaże się, że jest ona mniejsza od liczby szukanej, to...

- odrzucamy wszystkie liczby znajdujące się na lewo od elementu środkowego oraz ten element.
- kończymy algorytm.
- znaleźliśmy szukaną liczbę.
- odrzucamy wszystkie liczby znajdujące się na prawo od elementu środkowego oraz ten element.

Ćwiczenie 6

Uzupełnij pseudokod rekurencyjnego algorytmu wyszukiwania binarnego, wstawiając w puste miejsca właściwe fragmenty. Parametr `lewy` zawiera indeks lewego krańca przeszukiwanego ciągu, parametr `prawy` – indeks prawego krańca ciągu, a parametr `k` – szukany element.

```
funkcja WyszukiwanieBinarne(lewy, prawy, k):
    Jeżeli lewy > prawy:
        [          ]
        zakończ

    srodek ← (lewy + prawy) / 2

    Jeżeli ciag[srodek] == k:
        [          ]
        zakończ

    Jeżeli ciag[srodek] < k:
        [          ]
    W przeciwnym razie:
        [          ]
```

`wypisz(srodek)` `WyszukiwanieBinarne(lewy, srodek - 1, k)`

`wypisz("Tego elementu nie ma w ciągu")`

`WyszukiwanieBinarne(srodek + 1, prawy, k)`

Ćwiczenie 7

Dany jest następujący algorytm:

```
funkcja funkcja_1(tablica, pocz, kon)
    jeżeli pocz < kon
        indeks_pivot ← funkcja_2(tablica, pocz, kon)
        funkcja_1(tablica, pocz, indeks_pivot - 1)
        funkcja_1(tablica, indeks_pivot + 1, kon)

funkcja funkcja_2(tablica, pocz, kon)
    pivot ← tablica[kon]
    indeks_pivot ← pocz

    dla i od pocz do kon - 1
        jeżeli tablica[i] <= pivot
            zamień tablica[i] z tablica[indeks_pivot]
            indeks_pivot ← indeks_pivot + 1

    zamień tablica[indeks_pivot] z tablica[kon]

    zwróć indeks_pivot
```

Algorytm wykorzystuje metodę „dziel i zwyciężaj”. Wskaż, która funkcja odpowiedzialna jest za część „dziel”, a która za część „zwyciężaj”.

Część „dziel”: [ ]

Część „zwyciężaj”: [ ]

funkcja_1 | funkcja_2

Ćwiczenie 8

Zapisz za pomocą pseudokodu algorytm jednoczesnego wyszukiwania minimum i maksimum zbioru liczb, wykorzystując strategię „dziel i zwyciężaj”.

Specyfikacja problemu:

Dane:

- `tablica` – tablica liczb całkowitych

Wynik:

- `max` – największa liczba w tablicy
- `min` – najmniejsza liczba w tablicy

```
1
```

Dla nauczyciela

Autor: Bartosz Zadrożny
Przedmiot: Informatyka

Temat: Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”

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.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

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;

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:

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):

- Prześledzisz etapy realizacji strategii „dziel i zwyciężaj".
- Przeanalizujesz przykładowe algorytmy, w których zastosowano strategię „dziel i zwyciężaj".
- Wyjaśnisz, w jakich okolicznościach warto zastosować metodę „dziel i zwyciężaj".
- Przeanalizujesz złożoność czasową algorytmu opartego na strategii „dziel i zwyciężaj".
- Przeanalizujesz algorytm jednoczesnego wyszukiwania minimum i maksimum.

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: „Rozwiązywanie problemów informatycznych – strategia dziel i zwyciężaj". Uczniowie

mają zapoznać się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel, sprawdzając przygotowanie uczniów do lekcji, prosi wybrane osoby o wyjaśnienie, z jakich etapów składa się strategia „dziel i zwyciężaj” oraz w jakich algorytmach jest ona używana. Następnie wspólnie z uczniami analizuje przedstawione w sekcji „Przeczytaj” zastosowania tej metody w wyszukiwaniu binarnym oraz w algorytmie sortowania przez scalanie. W kolejnym kroku uczniowie pracując w parach, implementują omówione algorytmy w wybranym przez siebie języku programowania. Nauczyciel sprawdza poprawność rozwiązań.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Symulacja interaktywna”, czyta treść polecenia nr 1 „Przeanalizuj symulację interaktywną, która wizualizuje metodę „dziel i zwyciężaj” na przykładzie wyszukiwania binarnego” i omawia kolejne kroki rozwiązania. Uczniowie uruchamiają symulację i wykonują polecenie 2: „Wymyśl własne dane, które wprowadzisz do symulacji. Zastanów się, ile razy nastąpi podział tablicy dla twojego przykładu”.
3. **Ćwiczenie umiejętności.** Uczniowie indywidualnie wykonują ćwiczenia 1-4 z sekcji „Sprawdź się”. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie”. W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…”.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 5-8 z sekcji „Sprawdź się”.
2. Uczniowie wykonują polecenie 3 oraz 4 z sekcji „Symulacja interaktywna”.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Symulacja interaktywna”, „Sprawdź się” jako materiał do lekcji powtórkowej.