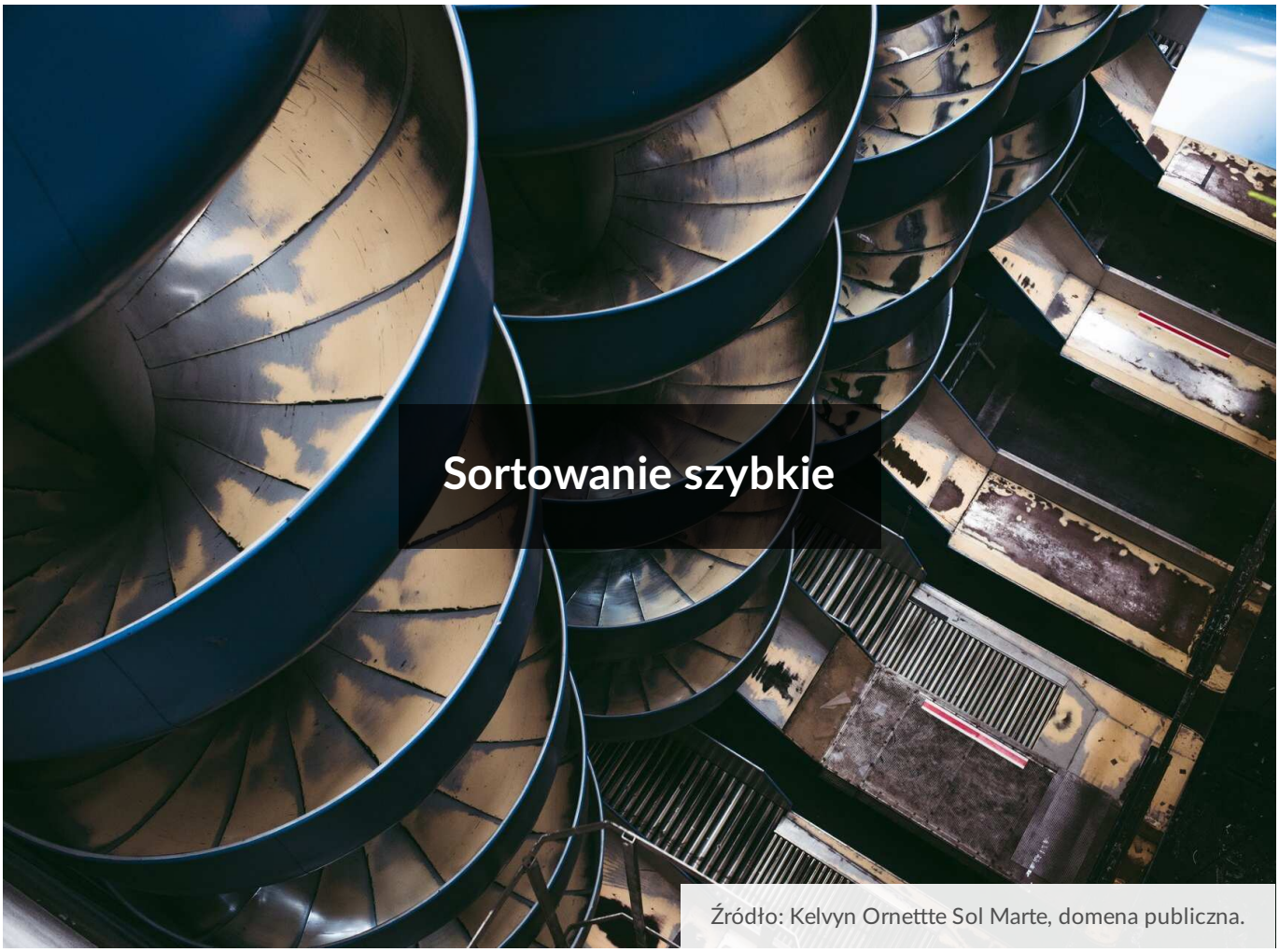
Sortowanie szybkie



Źródło: Kelvyn Ornettte Sol Marte, domena publiczna.

Wśród uporządkowanych przedmiotów łatwiej jest znaleźć to, czego szukasz.
W programach komputerowych porządek jest tak samo ważny, choć tutaj sprowadza się on przede wszystkim do posortowanych liczb. Jak uporządkować liczby? Oczywiście za pomocą algorytmów.

Sortowania używamy np. robiąc zakupy w sklepie internetowym, gdy układamy elementy zgodnie z rosnącą ceną. Chodzi jednak nie tylko o to, żeby dane zostały posortowane – równie ważne jest, by zadanie wykonać w możliwie najkrótszym czasie. Jak to zrobić? Możemy skorzystać z sortowania szybkiego.

Implementację sortowania szybkiego przedstawiamy w e-materiałach:

- Sortowanie szybkie w języku C++,
- Sortowanie szybkie w języku Java,
- Sortowanie szybkie w języku Python.

Więcej zadań? Sortowanie szybkie – zadania maturalne.

> **Twoje cele**
>
> - Prześledzisz pseudokod, który jest zapisem algorytmu sortowania szybkiego.

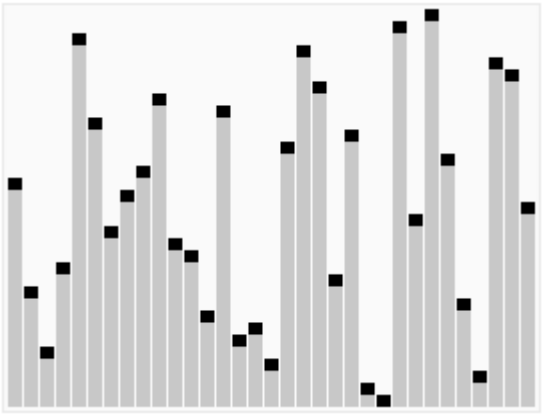
- Przeanalizujesz czas działania algorytmu *quick sort*, w tym dla przypadku optymistycznego i pesymistycznego.
- Przeprowadzisz sortowanie przykładowej tablicy za pomocą algorytmu sortowania szybkiego.

Przeczytaj

Działanie algorytmu *quick sort*

Zasada działania algorytmu *quick sort* jest oparta na metodzie „dziel i zwyciężaj".
W odniesieniu do sortowania szybkiego możemy ją przedstawić w trzech krokach:

1. **Dziel**: tablicę wejściową podziel na dwie podtablice, od indeksu `p` do indeksu `q`, tak aby przed elementem o indeksie `q` znalazły się elementy mniejsze lub równe elementowi osiowemu (ang. *pivot*), natomiast za tym elementem – większe lub równe. W ten sposób otrzymamy jedną tablicę o wartościach nie większych oraz drugą o wartościach nie mniejszych od `pivot`.
2. **Zwyciężaj**: dla powstałych w ten sposób dwóch podtablic wywołujemy rekurencyjnie funkcję sortowania szybkiego, która ponownie dzieli je na dwie podtablice i tak do momentu, aż uzyskamy tablice jednoelementowe.
3. **Połącz**: powstałe podtablice są zazwyczaj łączone w jedną, posortowaną całość. Jednak w przypadku sortowania szybkiego, będziemy wykonywać sortowanie w miejscu. Nie ma więc potrzeby łączenia elementów.



Film dostępny pod adresem /preview/resource/Rbb9mVRZrQpqS

Animacja algorytmu sortowania szybkiego sortującego tablicę losowych wartości.

Animacja przedstawia sortowanie wąskich kolumn od najniższej do najwyższej. Sortowanie przebiega bardzo szybko. Początkowo kolumny podlegające sortowaniu są różnej wielkości, są wymieszane. Następnie za pomocą przemieszczającej się strzałki w postaci łuku kolumny są porównywane i przestawiane od najniższej do najwyższej.

Pseudokod sortowania *quick sort*

Przeanalizujmy pseudokod sortowania *quick sort:*

```
funkcja sortowanieSzybkie(tablica, p, k)
    jeżeli p < k wykonaj:
        q = podziałTablicy(tablica, p, k)
        sortowanieSzybkie(tablica, p, q)
        sortowanieSzybkie(tablica, q + 1, k)
```

- `p` to miejsce, od którego chcemy posortować tablicę; gdy chcemy wykonać sortowanie od początku tablicy, to `p = 1`,
- `k` to miejsce zakończenia sortowania,
- `q` to punkt podziału tablicy.

Rozpoczynamy funkcję `sortowanieSzybkie` dla całej tablicy, którą chcemy posortować. Jeżeli `p` jest mniejsze od `k`, czyli tablica składa się z więcej niż jednego elementu, to wykonujemy następujące czynności. Wywołujemy funkcję `podziałTablicy`, która dzieli tablicę na dwie podtablice, a punkt podziału zapisuje w zmiennej `q`. Teraz rekurencyjnie wywołujemy funkcję `sortowanieSzybkie` dla podtablic. Algorytm kończy działanie, gdy w wyniku podziału pozostaną tablice jednoelementowe.

Pseudokod funkcji `podziałTablicy` wygląda następująco:

```
funkcja podziałTablicy(tablica, p, k)
    pivot = tablica(k)
    i = p - 1
    j = k + 1
    dopóki prawda wykonuj:
        wykonuj i = i + 1
            dopóki i <= k oraz tablica(i) < pivot
        wykonuj j = j - 1
            dopóki j >= p oraz tablica(j) > pivot
        jeżeli i <= j wykonaj:
            zamień tablica(i) z tablica(j)
        w przeciwnym razie wykonaj:
            zwróć j
```

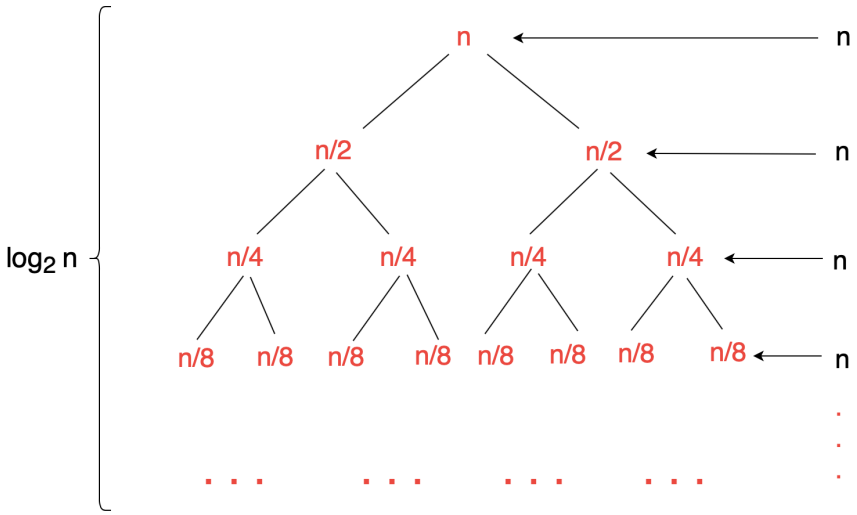
- Za pomocą zmiennej `pivot` zapisujemy liczbę, która odpowiada za to, w której podtablicy umieścimy każdy z elementów. Jeżeli rozpatrywana liczba będzie mniejsza od `pivot`, umieścimy ją w tablicy pierwszej, jeżeli większa, to w drugiej, a jeżeli równa, to może pojawić się zarówno w pierwszej, jak i drugiej. Zmienna `pivot` może być dowolnym elementem tablicy, np. pierwszym, ostatnim lub środkowym. My wybraliśmy ostatni element.

- Następnie w zmiennych `i` oraz `j` zapisujemy miejsca przed pierwszym oraz za ostatnim elementem tablicy, którą chcemy posortować.
- Zwiększamy wówczas indeks `i`, do momentu znalezienia elementu tablicy większego niż `pivot`. Indeks `j` zmniejszamy, aby w końcu znaleźć element mniejszy od `pivot`.
- Po znalezieniu mniejszego elementu zmieniamy te elementów, aby uzyskać w podtablicach odpowiednie wartości.
- Wymienione dwa punkty, opisujące czynności zamiany, wykonujemy do momentu, gdy `i` osiągnie wartość większą niż `j`.
- W tym momencie funkcja zwraca indeks `j`, czyli punkt podziału tablicy.

Czas działania algorytmu

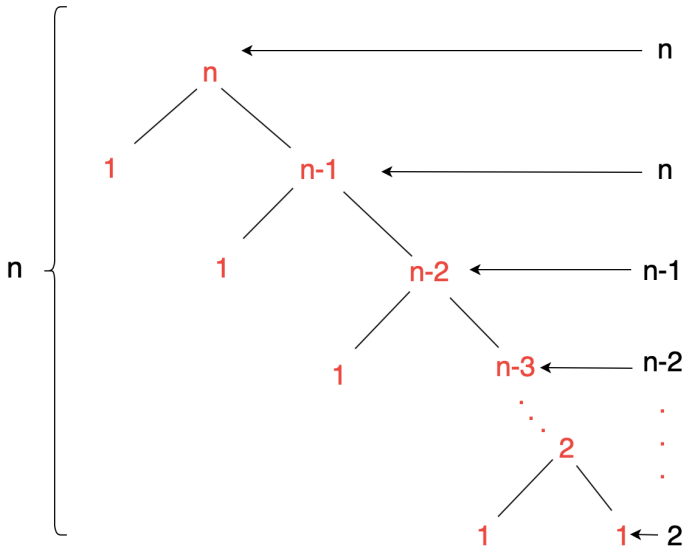
Złożoność obliczeniowa algorytmu sortowania szybkiego jest zależna od rozmiarów części, na jakie jest dzielona tablica. Jeżeli za każdym razem będzie wybierany `pivot` o wartości najmniejszej lub największej w tablicy, podział tablicy będzie nierównomierny. Tym samym poważnie wzrośnie czas działania algorytmu. W takim ujęciu przypadek optymistyczny i pesymistyczny nie są związane ze sposobem uporządkowania elementów w tablicy, ale z wyborem elementu osiowego, czyli ze sposobem dzielenia tablicy.

Rozważmy najpierw **przypadek optymistyczny**, którego złożoność obliczeniowa algorytmu wynosi O(n log n). Jeżeli tablica za każdym razem jest dzielona na dwie równe części, otrzymamy właśnie taki czas działania. Przedstawia to poniższy schemat:



Złożoność O(n $\log_2$ n)

W **przypadku pesymistycznym** tablica jest dzielona na dwie części, z których jedna zawiera jeden element, a druga resztę elementów. Złożoność obliczeniowa wynosi wtedy $O(n^2)$.



Złożoność $O(n^2)$

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

Algorytm sortowania szybkiego – złożoność obliczeniowa

Założeniem algorytmu sortowania szybkiego jest – jak sama nazwa wskazuje – możliwie najkrótszy czas sortowania kosztem większego wykorzystania pamięci (wywołania rekurencyjne).

Średnia złożoność obliczeniowa algorytmu szybkiego sortowania wynosi:

$$O\left(n \log n\right)$$

Złożoność obliczeniowa takiego algorytmu w przypadku pesymistycznym to:

$$\frac{n^2}{2}$$

Natomiast jego złożoność obliczeniowa w przypadku optymistycznym to:

$$n + 2\mathrm{T}(\frac{n}{2})$$

Słownik

sortowanie w miejscu

algorytmy sortowania, w których elementy znajdują się cały czas w początkowej tablicy i nie jest potrzebna dodatkowa pamięć

pivot

element osiowy – element tablicy wybrany wedle ustalonego schematu; jest to element dzielący tablicę

Aplet

Polecenie 1

Kluczowym elementem algorytmu sortowania szybkiego jest funkcja podziału (`podzielTablicę`). Jej zadaniem jest rozłożenie wartości tak, aby po lewej stronie znajdowały się wartości mniejsze od zmiennej `pivot`, a po prawej – większe. Sposób wyboru elementu `pivot` może być deterministyczny (weź zawsze i-ty element) lub losowy (weź losowy element) – zaprezentowany przykład jest niezależny od metody wyboru. Możemy nieformalnie opisać tę funkcję w następujący sposób:

```
funkcja podzielTablicę(zbiór "A")
    wybierz wyraz "pivot" ze zbioru "A"

    podziel zbiór "A" na zbiory "A1", "A2", "A3" następująco:
        "A1" zawiera elementy mniejsze od wartości wyrazu "pivo
        "A2" zawiera elementy równe wartości wyrazu "pivot",
        "A3" zawiera elementy większe od wartości wyrazu "pivot

    zwróć zbiory "A1", "A2", "A3"
```

Dla tak zapisanej funkcji podziału algorytm *quick sort* można określić następująco.

```
funkcja sortowanieSzybkie(zbiór "A")
    jeżeli zbiór "A" zawiera więcej niż 1 element
        zbiory "A1", "A2", "A3" = podzielTablicę(A)

        zbiór "B1" = sortowanieSzybkie("A1")
        zbiór "B3" = sortowanieSzybkie("A3")

        połącz uporządkowane zbiory kolejno "B1", "A2", "B3" w
        zwróć uporządkowany zbiór "B"
```

Zapoznaj się z apletem prezentującym szukanie piwotu.

Uruchom aplet prezentujący kroki wykonywane przez algorytm sortowania szybkiego w wyżej przedstawionym wariancie. Sprawdź działanie dla domyślnego przykładu. Wpisz własne wartości tablicy (maks. dł. 15) z przedziału [-99,99] i sprawdź działanie dla różnych zestawów danych. Znajdź pesymistyczny przykład, dla którego algorytm zostanie wykonany w czasie $O(n^2)$.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DNd9FlVmO

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi sortowania szybkiego.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zaznacz poprawną odpowiedź. Wskaż, jaką technikę wykorzystuje algorytm *quick sort.*

- ◯ porównań
- ◯ wyboru wartownika przed indeksem 0
- ◯ dziel i zwyciężaj

Ćwiczenie 2

Wskaż, czym jest sortowanie w miejscu.

- ◯ Jest to algorytm sortowania, którego złożoność czasowa wynosi O(n log n).
- ◯ Jest to algorytm sortowania, do którego może być potrzebna dodatkowa pamięć.
- ◯ Jest to algorytm sortowania, w którym elementy znajdują się cały czas w początkowej tablicy.
- ◯ Jest to algorytm sortowania, w którym elementy w tablicy nie zmieniają swojej pozycji.

Ćwiczenie 3

Wskaż, jaka jest złożoność czasowa przypadku pesymistycznego sortowania szybkiego.

- ☐ text=O(n $\log_2$ n)
- ☐ text=O($\log_2$ n)
- ☐ text=O(n^2)
- ☐ text=O(n)

Ćwiczenie 4

Zaznacz złożoność czasową przypadku optymistycznego sortowania szybkiego.

- ◯ $O(n^2)$
- ◯ $O(n \log_2 n)$
- ◯ $O(n)$
- ◯ $O(\log_2 n)$

Ćwiczenie 5

Zaznacz, czym zajmuje się zmienna `pivot` w algorytmie sortowania szybkiego.

- ◯ Jest to pozycja punktu podziału tablicy.
- ◯ Jest to miejsce zakończenia sortowania.
- ◯ Odpowiada za to, w której części tablicy umieścimy każdy z elementów.
- ◯ Jest to pozycja początku tablicy.

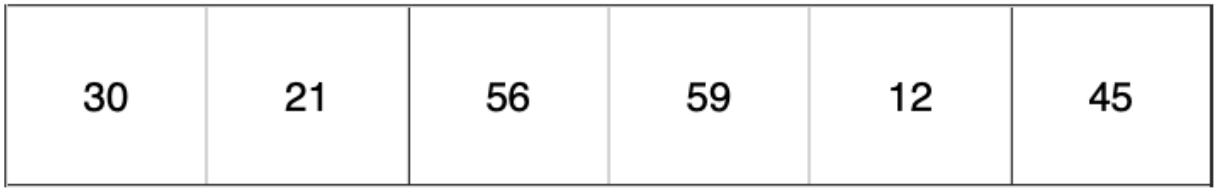
Ćwiczenie 6

Uporządkuj kolejne linijki pseudokodu sortowania quick sort.

- Quicksort(tab, z, a)
- Quicksort(tab, z, t)
- Quicksort(tab, a+1, t)
- a ← Podział(tab, z, t)
- jeżeli z < t wykonuj:

Ćwiczenie 7

Wskaż, które elementy tablicy możemy użyć jako `pivot`. Zaznacz wszystkie poprawne odpowiedzi.



- [ ] 12
- [ ] 56
- [ ] 45
- [ ] 30

Ćwiczenie 8

Wskaż dwa elementy tablicy, które zostaną zamienione jako pierwsze, według algorytmu sortowania szybkiego podanego w poprzedniej sekcji. Wartość zmiennej `pivot` wynosi 45.

30	21	56	59	12	45

- ◯ 30 i 12
- ◯ 56 i 12
- ◯ 30 i 45
- ◯ 56 i 45

Praca domowa

Polecenie 1.1

Przeanalizuj prezentację, z której dowiesz się, w jaki sposób wykonuje się sortowanie szybkie. Następnie spróbuj wykonać takie sortowanie samodzielnie na przykładowej tablicy liczb.

1

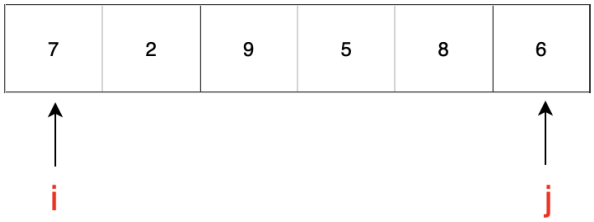
Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Pokażemy teraz krok po kroku, w jaki sposób wykonywany jest algorytm sortowania quick sort. Spróbujmy posortować tablicę `tab = [7, 2, 9, 5, 8, 6]`.

Sprawdźmy na początek, czy dana tablica posiada więcej niż jeden element. Widzimy, że ma sześć elementów. Tym samym możemy przejść do funkcji `podziałTablicy`.

2

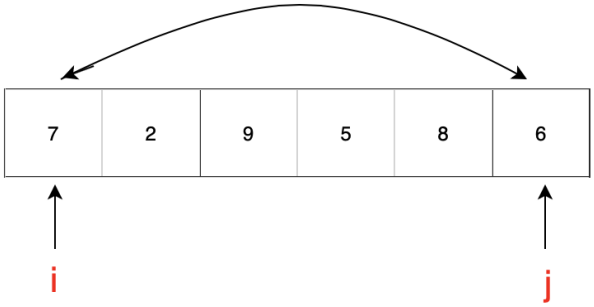


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Teraz wybierzmy `pivot`. Według algorytmu, będzie to nasz ostatni element tablicy, czyli 6. Następnie ustawiamy indeksy `i` oraz `j` na swoich miejscach, to znaczy `i` przed początkiem tablicy, natomiast `j` za końcem. Przesuwamy `i` w prawo, o jedną pozycję, w celu znalezienia elementu większego lub równego `pivot`, a `j` zmniejszamy o jedną, by znaleźć element mniejszy lub równy `pivot`. Na załączonym obrazku widzimy, gdzie znajdują się indeksy po powyższych operacjach.

3

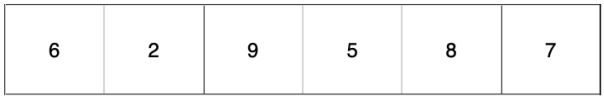


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

W tym momencie możemy dokonać zamiany 7 z 6.

4

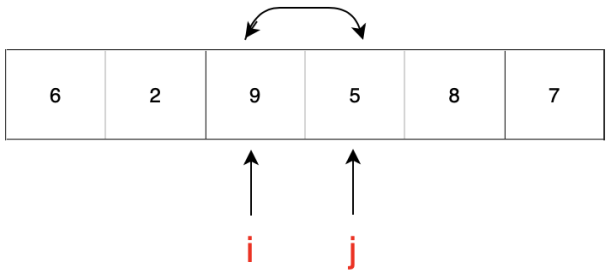


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Po zamianie tablica wygląda jak na powyższym obrazku.

5



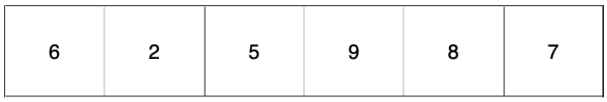
Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Położenie `i` oraz `j` znowu się zmienia. Wartość `i` wzrasta, natomiast `j` zmniejsza się.

Tym samym `i` wskazuje element o wartości 9, a `j` element o wartości 5. Dokonujemy zamiany.

6

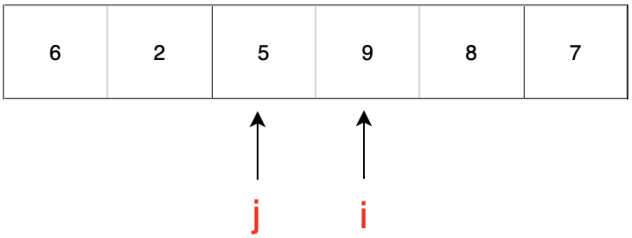


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Tablica po zamianie elementów wskazywanych przez `i` oraz `j`.

7

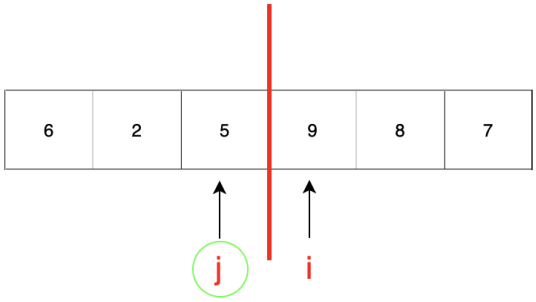


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Ponownie `i` oraz `j` zmienia wartość.

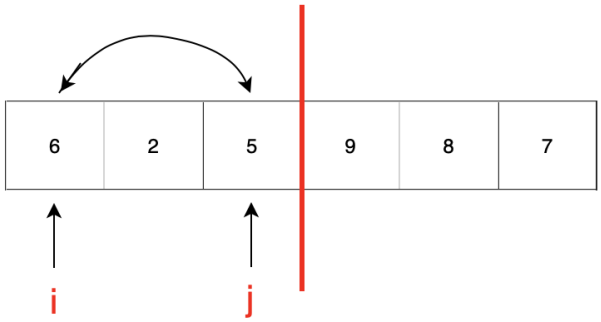
8



Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Tutaj powinna nastąpić zamiana elementów, jednak wartość `i` jest w tym momencie wyższa od wartości `j`. Zwracamy więc `j` jako punkt podziału tablicy. Tablica składa się teraz z dwóch części. Rekurencyjnie wywoływana jest funkcja `sortowanieSzybkie` dla pierwszej części.

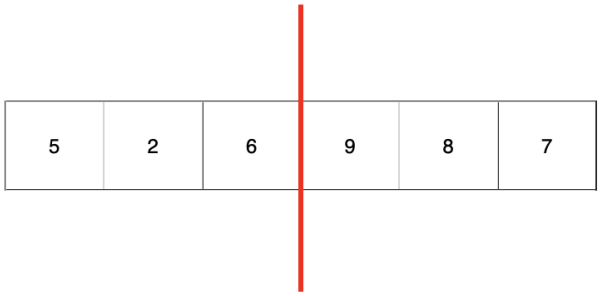


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Wykonujemy ponownie te same kroki co poprzednio, jednak dla mniejszej części tablicy. Nasz `pivot` to teraz element o wartości 5, ponieważ wybieramy ostatnią pozycję. Zmienne `i` oraz `j` ustawiają się przed oraz za naszą podtablicą. Zmieniają wartości, by znaleźć się w odpowiednich pozycjach. Ponieważ 6 jest większe od `pivot`, a 5 jest równe `pivot`, dokonujemy zamiany.

10

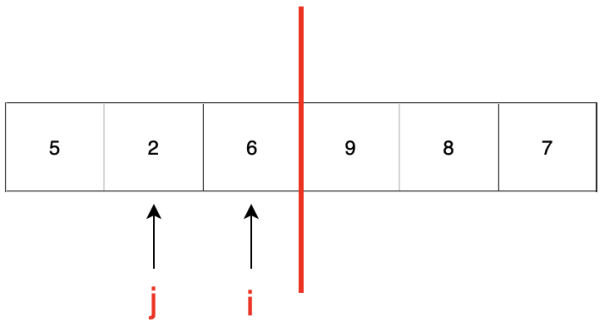


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Tablica po zamianie elementów.

11

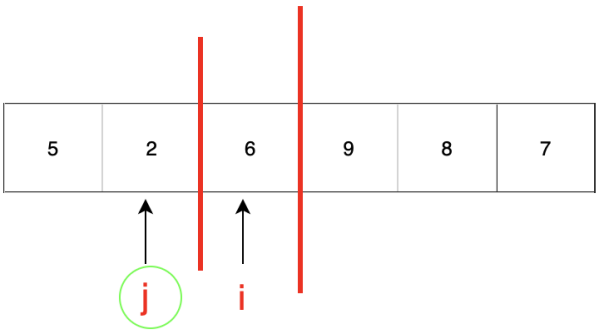


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Zmienne `i` oraz `j` zmieniają swoje pozycje.

12

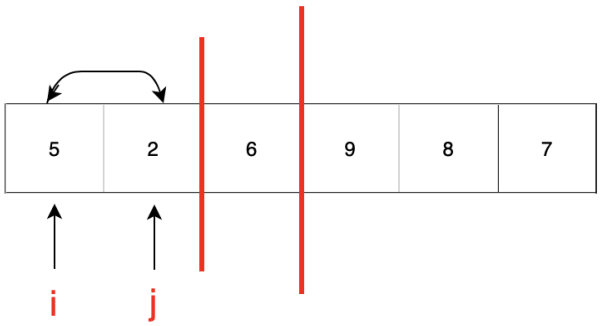


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Teraz powinna nastąpić zamiana, jednak `j` oraz `i` minęły się, więc zwracamy `j` jako punkt podziału tablicy.

13

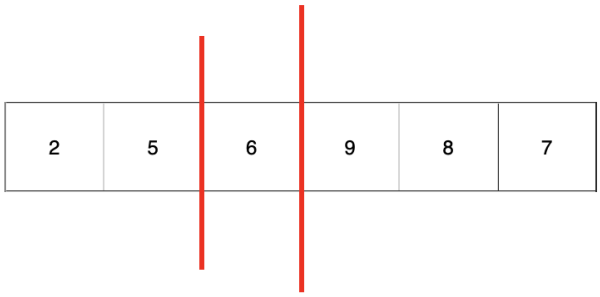


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Dla mniejszej tablicy wykonujemy te same kroki. Ustawiamy wartość `pivot` na 2. Ponieważ 5 jest większe od `pivot`, a 2 jest równe `pivot`, dokonujemy zamiany.

14

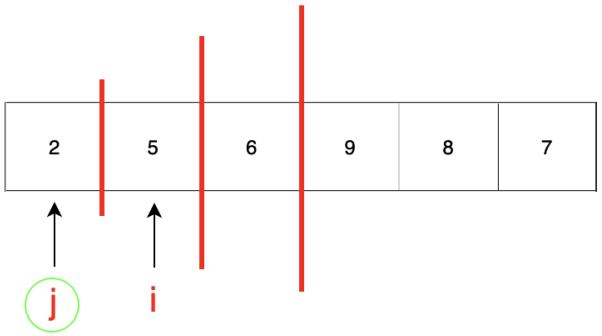


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Tablica po zamianie elementów.

15

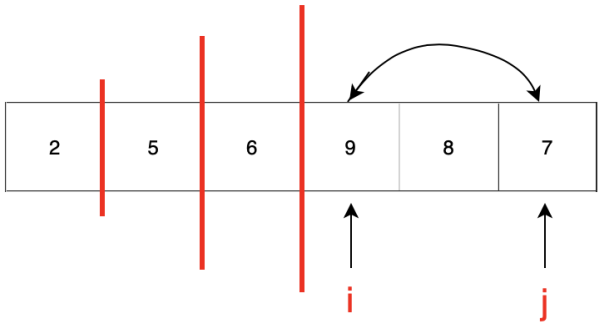


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Zmienne `i` oraz `j` zmieniają swoje pozycje. Powinna nastąpić zamiana, jednak `i` oraz `j` minęły się. Ponownie dzielimy tablicę. Pozostały nam trzy, jednoelementowe podtablice, więc je uznajemy jako posortowane. Przechodzimy do kolejnej części.

16

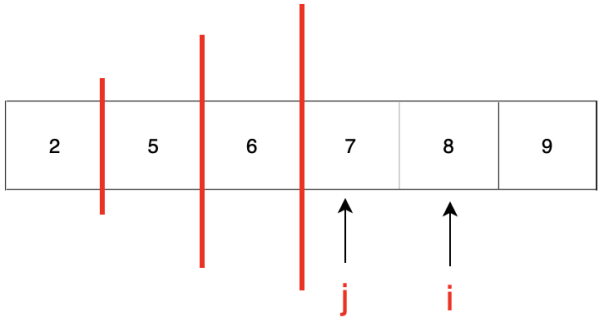


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Teraz `pivot` ustawiamy na wartość 7. Dokonujemy zamiany elementów.

17

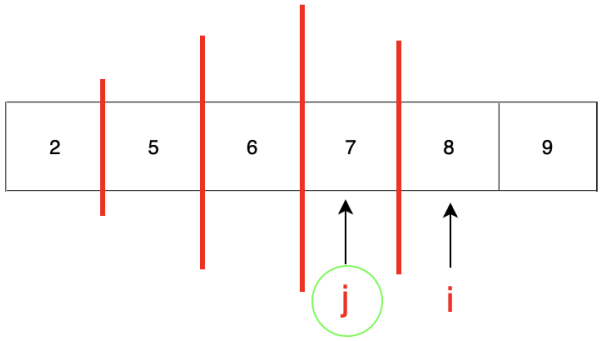


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Zmienne `i` oraz `j` zmieniają swoje pozycje.

18

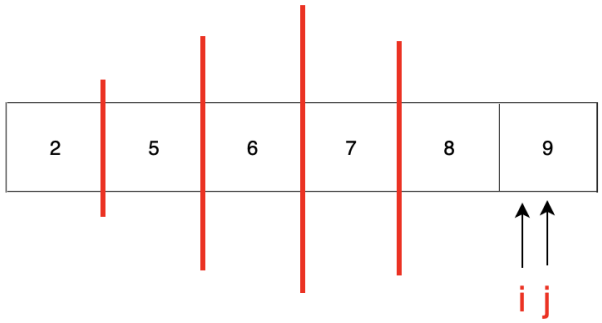


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Nie wykonujemy zamiany elementów. Zmienne minęły się, więc `j` zwracamy jako punkt podziału tablicy.

19

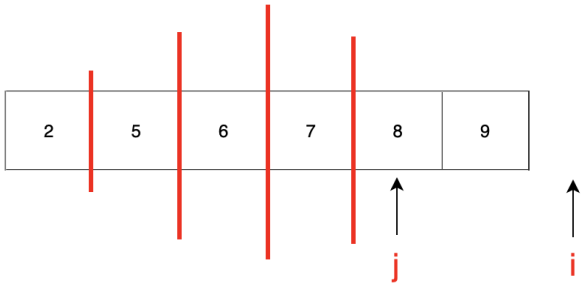


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Pozostała tablica jednoelementowa, którą już się nie zajmujemy. Wystarczy posortować jeszcze tablicę dwuelementową. `Pivot` to ostatni element, czyli liczba 9. Zmienne `i` oraz `j` znajdują się w tej samej pozycji. Po operacji zamiany nie zachodzi więc zmiana kolejności elementów w tablicy.

20

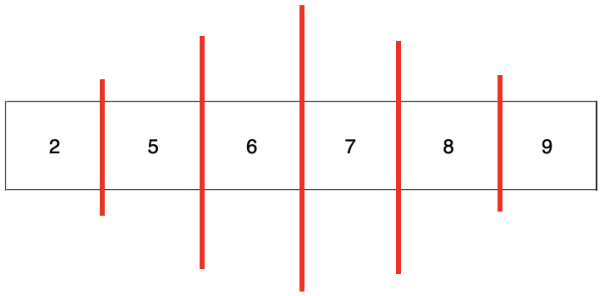


Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

Zmienne `i` oraz `j` minęły się, więc `j` jest naszym punktem podziału tablicy.

21



Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ppzx1Yyem

W ten sposób otrzymaliśmy tablice jednoelementowe. A zatem kończymy algorytm. Uzyskaliśmy posortowaną tablicę.

Specyfikacja problemu:

Dane:

- `n` – liczba naturalna; rozmiar tablicy
- `tab` – `n`-elementowa tablica liczb całkowitych

Wynik:

Program wyświetla posortowaną za pomocą algorytmu sortowania szybkiego niemalejąco tablicę `tab`.

```
1
```

```
1
```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie szybkie

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres 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 + 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 pseudokod, który jest zapisem algorytmu sortowania szybkiego.

- Przeanalizujesz czas działania algorytmu *quick sort*, w tym dla przypadku optymistycznego i pesymistycznego.
- Przeprowadzisz sortowanie przykładowej tablicy za pomocą algorytmu sortowania szybkiego.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

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: „Sortowanie szybkie". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Prowadzący wyświetla na tablicy interaktywnej zawartość sekcji „Wprowadzenie" i omawia cele do osiągnięcia w trakcie lekcji.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu zawartego w sekcji „Przeczytaj”, jeśli nauczyciel - na podstawie raportu na platformie - uważa, że przygotowanie uczniów jest wystarczające, może pominąć tę czynność.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Aplet”. Uczniowie uruchomiają aplet prezentujący kroki wykonywane przez algorytm Quicksort w przedstawionym wariancie. Sprawdzają działanie dla domyślnego przykładu. Wpisują własne wartości tablicy (maks. dł. 15) z przedziału [-99,99] i sprawdzają działanie dla różnych zestawów danych. Znajdują pesymistyczny przykład, dla którego algorytm zostanie wykonany w czasie $O(n^2)$
3. **Ćwiczenie umiejętności.** Liga zadaniowa - uczniowie wykonują indywidualnie na czas ćwiczenia nr 1-8 z sekcji „Sprawdź się”, a następnie omawiają zadania na forum.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń.

Praca domowa:

1. Uczniowie wykonują pracę domową z sekcji „Sprawdź się”. Analizują przedstawioną prezentację, która utrwala wiedzę dotyczącą tego, w jaki sposób wykonuje się sortowanie szybkie. Następnie próbują wykonać takie sortowanie samodzielnie na przykładowej tablicy liczb, wykorzystując wybrany język programowania.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.