Sortowanie przez scalanie

- Wprowadzenie
- Przeczytaj
- Aplet
- Sprawdź się
- Dla nauczyciela

Źródło: Xavi Cabrera, domena publiczna.

Znamy już różne algorytmy sortowania – mniej lub bardziej skomplikowane, sortujące z różną szybkością. W tym e-materiale omówimy algorytm, który ma mniejszą złożoność czasową niż wszystkie dotąd przez nas omawiane. Jest to rekurencyjny algorytm sortowania przez scalanie (ang. *merge sort*) oparty na zasadzie „dziel i zwyciężaj” (jej nazwa to parafraza łacińskiej maksymy „dziel i rządź”).

Implementacje omawianego algorytmu przedstawiamy w e-materiałach:

- Sortowanie przez scalanie w języku C++,
- Sortowanie przez scalanie w języku Java,
- Sortowanie przez scalanie w języku Python.

Więcej zadań? Sięgnij do: Sortowanie przez scalanie – zadania maturalne.

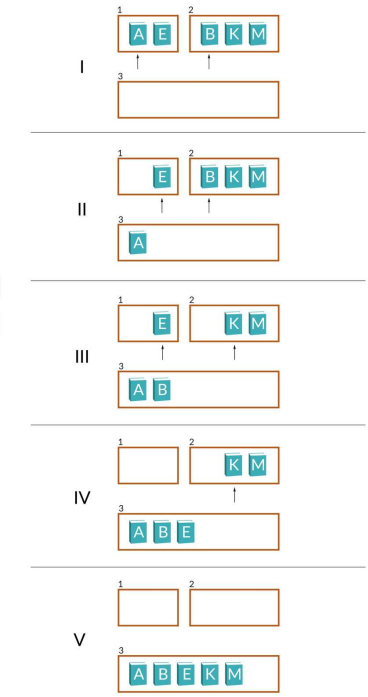
> **Twoje cele**
>
> - Przeanalizujesz algorytm sortowania przez scalanie.
> - Scharakteryzujesz złożoność czasową i pamięciową algorytmu *merge sort.*
> - Posortujesz przykładową tablicę za pomocą algorytmu sortowania przez scalanie.

Przeczytaj

Sortowanie przez scalanie polega na podzieleniu sortowanej struktury danych na dwie części, następnie na rekurencyjnym sortowaniu każdej z tych części i ponownym scaleniu już posortowanych części w jedną całość. Warto wiedzieć, że jako pierwszy tę metodę opisał John von Neumann, znany jako twórca m.in. teorii gier oraz koncepcji automatów komórkowych. Algorytm sortowania przez scalanie oparty jest na metodzie dziel i zwyciężaj.

Scalanie posortowanych tablic

Wyobraźmy sobie, że mamy dwie półki, każda z losową liczbą książek. Książki na półkach ułożone są w kolejności alfabetycznej. Naszym celem jest ułożenie wszystkich książek na trzeciej półce, w kolejności alfabetycznej. By to zrobić, porównujemy ze sobą pierwszą książkę z półki nr 1 z pierwszą książką na półce nr 2. Książka z literą położoną wcześniej w alfabecie, zostaje przeniesiona na trzecią półkę. Czynność powtarzamy, porównując książki, które ułożone są jako pierwsze na półkach. W momencie, gdy skończą nam się książki na jednej z półek, resztę książek z drugiej półki po prostu przenosimy na koniec naszej trzeciej, dodatkowej półki. Aby lepiej zrozumieć algorytm scalania książek na półkach, przeanalizuj rysunek.



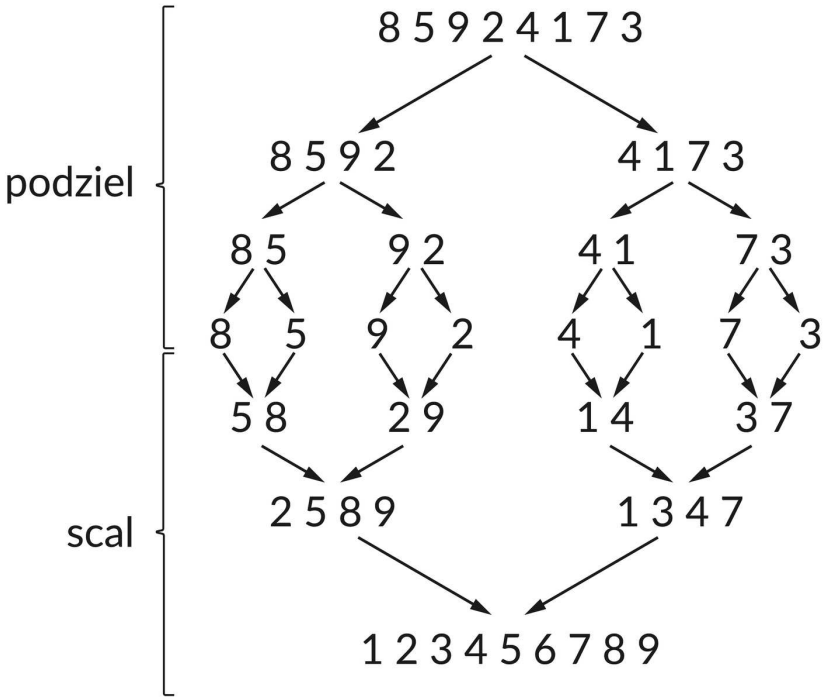
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Etapy sortowania przez scalanie

Chcemy uporządkować niemalejąco `n`-elementową tablicę liczb całkowitych. W jaki sposób to zrobić, korzystając z metody sortowania przez scalanie?

Aby użyć omówionego wcześniej algorytmu scalania, musimy podzielić tablicę na mniejsze, same w sobie posortowane tablice. W tym celu tablicę dzielimy na pół – jeżeli liczba elementów jest nieparzysta, pierwsza tablica będzie miała jeden element więcej.

Proces powtarzamy do momentu, w którym osiągniemy tablice jednoelementowe. Wiemy, że tablica posiadająca jeden element już jest posortowana. Dzięki temu możemy scalić dwie tablice jednoelementowe i uzyskać kolejną, posortowaną dwuelementową tablice. Następnie możemy scalić dwie posortowane tablice dwuelementowe, aby uzyskać czteroelementową itd. Przejdźmy do rysunku przedstawiającego kolejne etapy sortowania przez scalanie na podstawie przykładowej tablicy.



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

Pseudokod sortowania przez scalanie

Specyfikacja:

Dane:

- `n` – liczba naturalna dodatnia, liczba elementów tablicy
- `tablica` – n-elementowa tablica liczb całkowitych

Wynik:

- posortowana niemalejąco tablica całkowitych liczb `tablica`

Algorytm ten można zapisać za pomocą pseudokodu w następujący sposób:

```
funkcja Sortowanie_scalanie(tablica, pocz, kon)
    jeżeli pocz < kon
        połowa ← zaokrąglij w dół do całkowitej((pocz + kon) / 2)
        Sortowanie_scalanie(tablica, pocz, połowa)
        Sortowanie_scalanie(tablica, połowa + 1, kon)
        Scal(tablica, pocz, połowa, kon)

funkcja Scal(tablica, pocz, polowa, kon)
    n1 ← polowa - pocz + 1
    n2 ← kon - polowa

    utwórz tablicę L o rozmiarze n1
    utwórz tablicę R o rozmiarze n2

    dla i od 0 do n1 - 1
        L[i] ← tablica[pocz + i]

    dla j od 0 do n2-1
        R[j] ← tablica[polowa + 1 + j]

    i ← 0
    j ← 0
    k ← pocz

    dopóki i < n1 ORAZ j < n2
        jeżeli L[i] <= R[j]
            tablica[k] ← L[i]
            i ← i + 1
        w przeciwnym razie
            tablica[k] ← R[j]
            j ← j + 1
        k ← k + 1

    dopóki i < n1
        tablica[k] ← L[i]
        i ← i + 1
        k ← k + 1

    dopóki j < n2
        tablica[k] ← R[j]
        j ← j + 1
```

```
42          k ← k + 1
```

W jaki sposób zastosowane jest podejście „dziel i zwyciężaj" w zapisanym pseudokodzie?

Przeanalizujmy wspólnie instrukcję tego algorytmu. Na początek sprawdzamy, czy podana tablica zawiera więcej niż jeden element. Jeżeli odpowiedź brzmi tak, przechodzimy do kolejnych poleceń.

> Linijka `jeżeli pocz < kon` jest odpowiedzialna za sprawdzenie, czy w tablicy znajduje się więcej niż jeden element. Dla przykładowych wartości `pocz = 1, kon = 2` warunek jest spełniony – tablica ma dwa elementy, element 1 oraz element 2. Jeżeli jednak `pocz = 2, kon = 2`, wtedy tablica ma tylko jeden element.

Od tego kroku zostaje wykorzystana strategia „dziel i zwyciężaj". Przyjrzyj się i spróbuj zauważyć podobieństwo opisu tej strategii z zapisanym wcześniej za pomocą pseudokodu algorytmem sortowania.

- Do zmiennej `połowa` zapisywany jest indeks podziału tablicy. Pamiętajmy, żeby wynik zaokrąglić w dół do części całkowitej. Dzięki temu tablica będzie podzielona na dwie części. W przypadku parzystej liczby elementów, będą to dwie równe części, w wypadku nieparzystej – pierwsza część będzie zawierała o jeden element więcej od drugiej.
- Następnie rekurencyjnie wywołujemy funkcję sortowania przez scalanie, osobno dla każdej z dwóch części. Tablica dzielona jest aż do otrzymania tablic jednoelementowych, czyli aż do momentu, w którym algorytm znajdzie przypadek podstawowy.
- Ostatnia instrukcja pseudokodu wywołuje funkcję, która łączy posortowane fragmenty tablic w jedną posortowaną tablicę.

Złożoność czasowa i pamięciowa algorytmu

Tablica dzielona jest zawsze na dwie części (co mogliśmy zaobserwować na rysunku przedstawiającym kolejne etapy sortowania przez scalanie tablicy liczb). 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. Złożoność operacji scalania na każdym poziomie drzewa niezależnie od uporządkowania elementów wynosi $n \cdot c$, gdzie c jest stałą zależną od jakości używanego sprzętu. A więc czas działania algorytmu zarówno dla zbiorów posortowanych, posortowanych odwrotnie jak i nieposortowanych jest proporcjonalny do:

$$n \cdot \log_2 n$$

Zatem złożoność czasowa wynosi:

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

Sortowanie przez scalanie wymaga użycia dodatkowej tablicy, w której będziemy zachowywać wynik scalania, więc dla n-elementowej tablicy złożoność pamięciowa wynosi O(n). W związku z tym, *merge sort* nie jest tzw. sortowaniem w miejscu.

Słownik

dziel i zwyciężaj

metoda, która polega na podzieleniu problemu na mniejsze, podobne problemy, znalezieniu ich rozwiązań, a następnie ponownym połączeniu ich w jedną całość, by otrzymać rozwiązanie całego problemu

John von Neumann

węgierski matematyk, chemik, fizyk i informatyk; jest uznawany za głównego twórcę teorii gier i formalizmu matematycznego mechaniki kwantowej

przypadek podstawowy

przypadek, dla którego funkcja rekurencyjna nie wywołuje samej siebie, a zamiast tego zwraca z góry określoną wartość

rekurencja

sytuacja, w której funkcja wywołuje samą siebie, aż do napotkania przypadku podstawowego

sortowanie w miejscu

tzw. *in situ;* algorytm, który do posortowania zbioru danych potrzebuje stałej ilości pamięci komputera, niezależnej od rozmiaru danych wejściowych; wszelkie potrzebne do otrzymania wyniku obliczenia są wykonywane w pamięci, w której znajdują się dane

złożoność czasowa

cecha algorytmu, która wyraża czas niezbędny do wykonania danego algorytmu dla konkretnych danych, podana jako funkcja rozmiaru tych danych wejściowych; zazwyczaj rozpatruje się liczbę operacji podstawowych (dominujących), w przypadku sortowania najczęściej jest to operacja porównania wartości dwóch elementów, można też przyjąć operację przestawiania elementów

koncepcja automatów komórkowych

polega na zastąpieniu zbioru skomplikowanych równań opisujących zachowanie się wielu układów fizycznych, systemem komórek sąsiadujących ze sobą według pewnego ustalonego wzorca

Aplet

Polecenie 1

Uruchom aplet przedstawiający kolejne kroki algorytmu *merge sort* dla przykładowej tablicy. Przetestuj algorytm sortowania dla n-elementowej tablicy. Tablicę możesz wypełnić liczbami z przedziału [-99, 99], wartość n nie może być większa od 15.

***Specyfikacja**:*

Dane:

- n – liczba naturalna dodatnia; liczba elementów w tablicy `tablica`
- `tablica` – n-elementowa tablica liczb całkowitych

Wynik:

- posortowana niemalejąco tablica liczb całkowitych `tablica`

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/DpUTI4esO

Polecenie 2

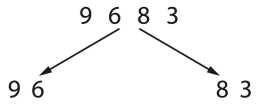
Przeanalizuj prezentację, w której zapoznasz się ze sposobem wykonywania sortowania przez scalanie na podstawie algorytmu, który został już przedstawiony.

Następnie spróbuj posortować większą tablicę (na przykład 8-elementową) samodzielnie.

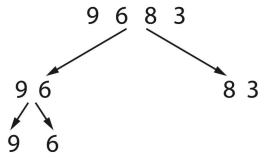
1

Przedstawimy teraz krok po kroku, w jaki sposób posortować tablicę za pomocą algorytmu sortowania przez scalanie. Naszą przykładową tablicą będzie: `tablica = [9, 6, 8, 3]`.

2



Sprawdzamy, czy tablica posiada więcej niż 1 element. Ma 4 elementy, więc możemy przejść do podziału tablicy na dwie równe części.



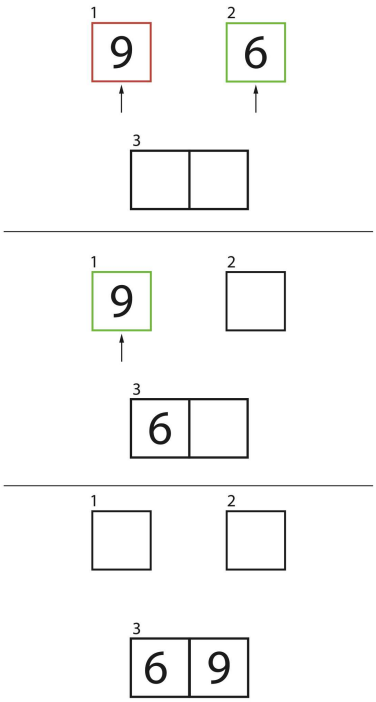
3

Wywołujemy rekurencyjnie funkcję sortowania dla pierwszej podtablicy. Tutaj ponownie następuje podział na dwie jednakowe części.

4

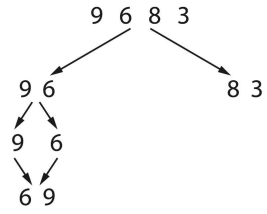
Otrzymaliśmy tablice jednoelementowe, więc nie wykonujemy dalszego podziału. Realizujemy ich scalanie według schematu. Porównujemy

dwa pierwsze elementy tablic. Ten o mniejszej wartości zapisywany jest w dodatkowej, trzeciej tablicy. Na kolejnym slajdzie pokazano proces scalania dwóch jednoelementowych tablic.

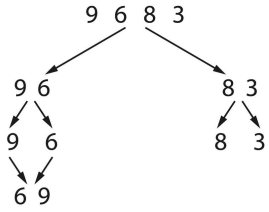


5

6



Zapisujemy scaloną tablicę. Teraz rekurencyjnie wywołujemy funkcję sortowania dla drugiej części początkowej tablicy.



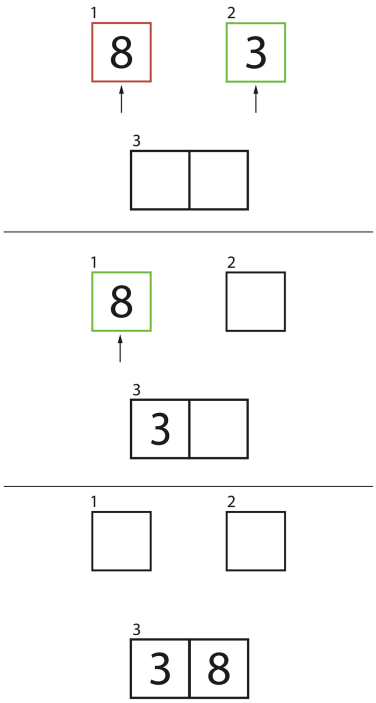
7

Podtablica posiada więcej niż 1 element, więc dzielimy ją na dwie jednakowe części.

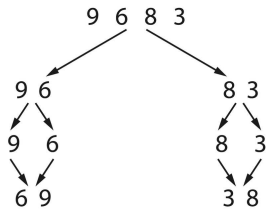
8

Uzyskaliśmy dwie tablice jednoelementowe. Przechodzimy do ich scalania, które przedstawione jest na rysunku w kolejnym slajdzie.

9



10

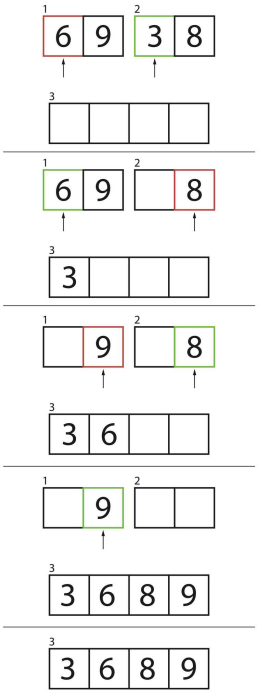


Zapisujemy otrzymaną tablicę.

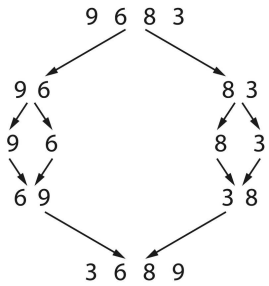
11

Teraz możemy przystąpić do scalenia dwuelementowych, posortowanych podtablic. W tym celu porównujemy ich pierwsze elementy. Ten o wartości mniejszej zapisywany jest w trzeciej, dodatkowej tablicy, a w następnym porównaniu sprawdzany jest element kolejny. Operacje wykonujemy do momentu, gdy podtablice staną się puste.

12



13



Wynik naszego scalania zapisujemy na końcu drzewa. W ten sposób otrzymaliśmy posortowaną tablicę za pomocą algorytmu sortowania przez scalanie.

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

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Ćwiczenie 2

Ćwiczenie 3

Ćwiczenie 4

Ćwiczenie 5

Ćwiczenie 6

Ćwiczenie 7

Ćwiczenie 8

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie przez scalanie

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

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

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

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

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

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

e) sortowania ciągu liczb przez scalanie,

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):

- Przeanalizujesz algorytm sortowania przez scalanie.
- Scharakteryzujesz złożoność czasową i pamięciową algorytmu *merge sort.*
- Posortujesz przykładową tablicę za pomocą algorytmu sortowania przez scalanie.

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: „Sortowanie przez scalanie”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie”.

2. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. **Praca z multimedium.** Uczniowie pracują w parach. Analizują treść polecenia nr 1 „Uruchom aplet przedstawiający kolejne kroki algorytmu merge sort dla przykładowej tablicy. Sprawdź działanie dla domyślnej tablicy. Przetestuj algorytm sortowania dla n-elementowej tablicy. Tablicę możesz wypełnić liczbami z przedziału [-99, 99], wartość n nie może być większa od 15" z sekcji „Aplet". Wybrana grupa omawia rozwiązanie na forum klasy.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1-8 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

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

Wskazówki metodyczne:

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