Sortowanie pozycyjne liczb

Źródło: Jan Antonin Kolar, domena publiczna.

Współczesne komputery przetwarzają ogromne ilości danych. Użyteczność informacji w znacznym stopniu zależy od tego, w jaki sposób zostaną one zaprezentowane użytkownikom.

W przypadku zbiorów liczb warto zadbać o ich uporządkowanie. Można to zrobić na różne sposoby – m.in. wykorzystując algorytm sortowania pozycyjnego. Jest on, podobnie jak w przypadku sortowania bąbelkowego czy sortowania przez scalanie, algorytmem stabilnym. Oznacza to, że elementy mające równą wartość w zbiorze nieposortowanym po uporządkowaniu będą ułożone w takiej samej kolejności.

W tym e-materiale omówimy ideę, cechy oraz sposób implementacji sortowania pozycyjnego.

Implementację sortowania pozycyjnego liczb w różnych językach programowania przedstawiamy w e-materiałach:

- Sortowanie pozycyjne liczb w języku C++,
- Sortowanie pozycyjne liczb w języku Java,
- Sortowanie pozycyjne liczb w języku Python.

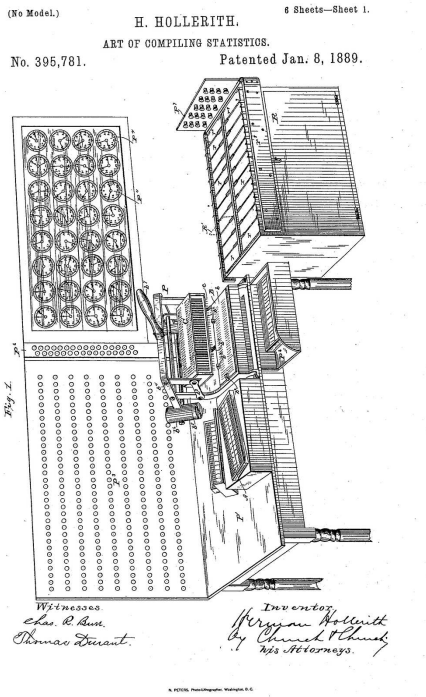
Więcej zadań? Sortowanie pozycyjne liczb – zadania maturalne

Twoje cele

- Przedstawisz koncepcję algorytmu sortowania pozycyjnego.
- Wskażesz różne rodzaje zastosowań dla algorytmu sortowania pozycyjnego.
- Przeanalizujesz implementację algorytmu sortowania pozycyjnego liczb, zapisaną w postaci pseudokodu.
- Zbadasz złożoność czasową algorytmu sortowania pozycyjnego.

Przeczytaj

Początki sortowania pozycyjnego sięgają XIX wieku. W roku 1890 do opracowania spisu ludności Stanów Zjednoczonych po raz pierwszy zastosowano maszynę licząco-analityczną. Jej prototyp został zbudowany w 1887 roku przez amerykańskiego inżyniera i wynalazcę Hermana Holleritha.


(No Model.) 6 Sheets—Sheet 1.

H. HOLLERITH.

ART OF COMPILING STATISTICS.

No. 395,781. Patented Jan. 8, 1889.

Witnesses. Inventor.

Maszyna licząco-analityczna

Źródło: dostępny w internecie: epo.org, tylko do użytku edukacyjnego.

Idea algorytmu sortowania pozycyjnego

W wypadku sortowania pozycyjnego nie porównuje się ze sobą wprost całych liczb zapisanych w tablicy wejściowej. Elementy są porządkowane ze względu na cyfry umieszczone na tych samych pozycjach.

Proces sortowania rozpoczynamy od najmniej znaczących cyfr jedności i kontynuujemy do chwili, w której posortujemy liczby kolejno według wszystkich cyfr, kończąc na tej najbardziej znaczącej (kolejno: jedności, dziesiątki, setki itd.). Do posortowania liczb według cyfr na danej pozycji możemy użyć dowolnego stabilnego algorytmu sortowania.

> **Przykład 1**
>
> Niech $A = \{140, 12, 68, 9, 512, 24, 60\}$ będzie tablicą wejściową. Uporządkujmy jej elementy, wykorzystując algorytm sortowania pozycyjnego.

Na początek rozważamy cyfry jedności. Sortujemy więc liczby za pomocą dowolnego algorytmu stabilnego tak, aby były ustawione niemalejąco według ostatniej cyfry. Wynikiem

pierwszego kroku będzie poniższa tablica.

Indeks:	0	1	2	3	4	5	6
Liczba:	140	60	12	512	24	68	9

W kolejnym kroku ponownie stabilnie sortujemy tablicę, tym razem według cyfr dziesiątek.

Indeks:	0	1	2	3	4	5	6
Liczba:	9	12	512	24	140	60	68

Podczas ostatniej iteracji będziemy rozważać cyfry setek.

Indeks:	0	1	2	3	4	5	6
Liczba:	9	12	24	60	68	140	512

Ostatecznie tablica wynikowa zawierająca liczby posortowane niemalejąco przedstawia się następująco:

A_1**= {9, 12, 24, 60, 68, 140, 512}**

Pseudokod

Poniżej została przedstawiona funkcja w pseudokodzie realizująca sortowanie pozycyjne. Wynikiem jej działania jest uporządkowana niemalejąco tablica wartości. Funkcja przyjmuje dwa argumenty: `tablicaSortowana` - czyli tablica do posortowania oraz `liczbaCyfr` - czyli wartość wskazująca, ile cyfr posiada największa liczba w tablicy.

```
1 funkcja sortowaniePozycyjne(tablicaSortowana, liczbaCyfr)
2     dla i = liczbaCyfr-1, liczbaCyfr-2, ..., 0 wykonuj
3         posortuj stabilnie wartości w tablicy tablicaSortowana we
```

Sortowanie według `i`-tej pozycji oznacza, że dla `i` równego `liczbaCyfr-1` bierzemy cyfrę jedności, dla `i` równego `liczbaCyfr-2` cyfrę dziesiątek - i tak dalej, aż do `i` równego `0`, gdzie bierzemy najbardziej znaczącą cyfrę w liczbie. Dla liczb, które nie posiadają cyfry na danej pozycji, przyjmujemy, że znajduje się tam cyfra 0.

Algorytm sortowania pozycyjnego z wykorzystaniem sortowania kubełkowego

Algorytm sortowania kubełkowego został dokładnie omówiony w e-materiale Sortowanie kubełkowe.

Proces sortowania można wyobrazić sobie jako umieszczanie elementów zbioru w kubełkach, które odpowiadają cyfrom systemu liczbowego. W przypadku systemu dwójkowego do dyspozycji mamy dwa kubełki oznaczone jako 0 i 1; dla systemu dziesiętnego jest ich dziesięć – oznaczają one cyfry od 0 do 9.

Porządkowanie zaczyna się od sprawdzenia cyfr jedności. Liczby zakończone cyfrą 0 trafiają do kubełka oznaczonego symbolem 0, te zakończone cyfrą 1 – do kubełka z numerem 1, a kończące się cyfrą 2 – do kubełka oznaczonego jako 2 itd.

Po ułożeniu wszystkich elementów w pojemnikach wyjmujemy liczby z kubełków. Zaczynamy od kontenera oznaczonego symbolem 0, następnie przechodzimy do kubełka z numerem 1, później 2 - i tak do końca. Liczby opuszczają pojemniki w takiej kolejności, w jakiej zostały do nich włożone.

Wyjmowane kolejno elementy umieszczamy w nowym zbiorze i powtarzamy dla niego opisane czynności. Tym razem sprawdzamy cyfry o wagach większych o 1. Wagą pozycji są kolejne potęgi podstawy systemu liczbowego, w jakim liczba jest zapisana. Elementy znowu trafiają do kubełków, a później jeszcze raz są z nich wyjmowane i tworzą następny zbiór.

Postępujemy tak do momentu, gdy posortujemy liczby według cyfr odpowiadających największym wagom.

> **Przykład 2**
>
> Niech $A = \{833, 997, 268, 69, 100, 584, 321, 333, 512, 660\}$ będzie tablicą wejściową. Uporządkujemy jej elementy, wykorzystując algorytm sortowania pozycyjnego i algorytm sortowania kubełkowego.

Podczas sortowania posłużymy się systemem dziesiętnym. Potrzebnych jest nam zatem 10 kubełków dla cyfr od 0 do 9. Porządkowanie zaczynamy od cyfr jedności (o wagach 10^0):

Kubełek:	0	1	2	3	4	5	6	7	8	9
Zawartość:	660 100	321	512	333 833	584			997	268	69

Potem zapisujemy liczby znajdujące się w kubełkach ponumerowanych od 0 do 9. Pamiętajmy przy tym, aby elementy opuszczały kubełki w takiej kolejności, w jakiej były w nich umieszczane. W rezultacie otrzymujemy następujący zbiór:

$A_1 = \{100, 660, 321, 512, 833, 333, 584, 997, 268, 69\}$

Następnie powtarzamy opisane czynności – tym razem bierzemy pod uwagę cyfry dziesiątek (o wagach 10^1):

Kubełek:	0	1	2	3	4	5	6	7	8	9
Zawartość:	100	512	321	333 833			69 268 660		584	997

Ponownie zapisujemy liczby z poszczególnych kubełków:

A_2= {100, 512, 321, 833, 333, 660, 268, 69, 584, 997}

Wszystkie liczby w przykładowym zbiorze są najwyżej trzycyfrowe. Pozostał więc jeszcze jeden cykl operacji. Zauważmy, że wśród elementów znajduje się liczba dwucyfrowa – w jej przypadku brakującą pozycję (odpowiadającą wadze 10^2) zastępujemy cyfrą 0.

Kubełek:	0	1	2	3	4	5	6	7	8	9
Zawartość:	069	100	268	333 321		584 512	660		833	997

Po raz ostatni odczytujemy elementy zapisane w kontenerach. Otrzymana tablica wynikowa jest posortowana rosnąco:

A_3= {69, 100, 268, 321, 333, 512, 584, 660, 833, 997}

Zapis w pseudokodzie

Zapoznajmy się teraz z implementacją algorytmu zapisaną w postaci pseudokodu.

Specyfikacja:

Dane:

- `liczbaElementow` – liczba naturalna; długość tablicy z danymi
- `liczbaCyfr` – liczba naturalna; liczba cyfr tworzących największą liczbę z tablicy
- `tablica[0..liczbaElementow-1][0..liczbaCyfr-1]` – dwuwymiarowa tablica liczb naturalnych; dane do posortowania

Wynik:

- `tablica[0..liczbaElementow-1][0..liczbaCyfr-1]` – posortowana niemalejąco tablica liczb naturalnych

> **Ważne!**

> W tym zadaniu występuje nietypowa forma przechowywania liczb. Przykładowo do `tablica[4][3] = {{0,5,0}, {1,0,0}, {1,5,6}, {2,0,0}}` możemy odwołać się na dwa różne sposoby. Możemy wskazać albo jedną **kompletną** tablicę z liczbą, albo pojedynczą cyfrę dowolnej liczby. Dla przykładu `{1,5,6}` oznacza liczbę 156.
>
> **Przykłady:**
> `tablica[0] = {0, 5, 0}`
> `tablica[0][0] = 0`
> `tablica[0][1] = 5`
> `tablica[0][2] = 0`
> `tablica[2] = {1, 5, 6}`
> `tablica[2][0] = 1`
> `tablica[2][1] = 5`
> `tablica[2][2] = 6`

Tablica `kubełki[0..9]` służy do tymczasowego zapamiętania liczb znajdujących się w kubełku odpowiadającym sprawdzanej obecnie cyfrze.

```
dla d = liczbaCyfr-1, liczbaCyfr-2, ..., 0 wykonuj
    dla n = 0, 1, ..., liczbaElementow-1 wykonuj
        cyfra ← tablica[n][d]
        umieść wewnątrz kubełki[cyfra] wartość tablica[n]
    x ← 0
    dla n = 0, 1, ..., 9 wykonuj
        dopóki kubełki[n] zawiera liczby wykonuj
            liczba ← wyjmij liczbę z kubełka kubełki[n]
            tablica[x] ← liczba
            x ← x + 1
```

Ogólny algorytm sortowania pozycyjnego możemy przedstawić w następujący sposób:

```
posortuj_pozycyjnie(tablica_nieuporządkowana)
    dla każdej pozycji elementu tablicy, w kolejności od najmniej
        utwórz kubełki dla każdego znaku (wartości), który może w
        wykonaj sortowanie kubełkowe tablicy według tych znaków (
        dla kolejnych kubełków
            przenieś elementy z kubełka na koniec porządkowanego
    zwróć uporządkowany ciąg
```

Własności sortowania pozycyjnego

Złożoność obliczeniowa zaprezentowanego algorytmu zależy w dużym stopniu od tego, z ilu cyfr składają się liczby zapisane w sortowanej tablicy.

> **Ciekawostka**
>
> Warto pamiętać, że używając algorytmu sortowania pozycyjnego, nie musimy zapisywać liczb w systemie dziesiętnym. Zwiększenie podstawy systemu liczbowego przekłada się na przyspieszenie działania programu. Zwiększa się też rozmiar wykorzystywanej pamięci.

Algorytm sortowania pozycyjnego jest algorytmem stabilnym, co oznacza, że elementy o równej wartości po posortowaniu będą pojawiać się w takiej samej kolejności jak w nieposortowanym zbiorze.

W przypadku sortowania pozycyjnego wykonuje się tyle cykli operacji, ile cyfr użyto do zapisania największej liczby w sortowanym zbiorze (przyjmijmy, że liczbę cyfr oznaczymy jako d). Całkowita złożoność czasowa będzie zależeć od użytego dodatkowego algorytmu sortowania (oznaczmy złożoność dodatkowego algorytmu jako m). Złożoność czasowa wynosi więc $O(dm)$.

W przypadku użycia sortowania kubełkowego, w każdym przebiegu sprawdzane są wszystkie elementy tablicy (o rozmiarze n), a następnie pobierana jest zawartość każdego z k kubełków. Złożoność obliczeniowa algorytmu wynosi zatem $O(d(n + k))$.

Zwiększenie liczby kubełków skutkuje zmniejszeniem liczby potrzebnych cyfr (im większa jest wartość k, tym mniejsze jest d). W rezultacie skraca się czas potrzebny do zrealizowania algorytmu.

Zastosowanie sortowania pozycyjnego

Sortownie pozycyjne rzadko wykorzystywane jest w praktyce. Dobrze się sprawdza, gdy należy uporządkować zbiór liczb całkowitych podobnych do siebie (mających np. taką samą liczbę cyfr bądź rozpoczynających się od identycznych ciągów cyfr) albo należących do niewielkiego przedziału. Metoda ta jest z kolei trudna do zastosowania w przypadku porządkowania liczb zmiennoprzecinkowych.

Po zmodyfikowaniu algorytm sortowania pozycyjnego może być z powodzeniem stosowany do znajdowania duplikatów w tablicy.

Słownik

> **liczba zmiennoprzecinkowa**
>
> reprezentacja liczby rzeczywistej zapisanej w postaci wykładniczej
> **sortowanie stabilne**

mechanizm sortowania, w przypadku którego elementy o identycznych wartościach są umieszczane w uporządkowanym zbiorze w takiej samej kolejności, w jakiej znajdowały się przed sortowaniem

system liczbowy

zbiór reguł dotyczących zapisu liczb z wykorzystaniem kombinacji rozmaitych symboli; wyróżniamy systemy liczbowe addytywne (np. rzymski) i pozycyjne (np. dziesiętny)

Schemat interaktywny

Polecenie 1

Gdy używamy algorytmu sortowania pozycyjnego, niezbędna jest znajomość liczby cyfr, z których składa się każdy element porządkowanego zbioru. Bez tej informacji nie wiemy, ile powtórzeń cyklu sortowania należy wykonać, by posortować liczby według cyfr na kolejnych pozycjach danej liczby począwszy od cyfry najmniej znaczącej.

W przedstawionej aplikacji wykorzystywana jest funkcja `ObliczLiczbęCyfrWLiczbie()`. Powinna ona zwracać zmienną `LiczbaCyfr` – liczbę naturalną informującą o tym, z ilu cyfr składa się liczba podana jako argument. Niestety, kod funkcji jest niekompletny. Twoim zadaniem jest wpisanie brakujących instrukcji. Główna funkcja programu została już zdefiniowana.

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi sortowania pozycyjnego.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zdecyduj, czy podane zdanie jest prawdziwe, czy fałszywe.
Proces sortowania pozycyjnego rozpoczynamy od najbardziej znaczących cyfr porównywanych liczb.

- ◯ fałsz
- ◯ prawda

Ćwiczenie 2

Wskaż, co przechowują kubełki w trakcie wykonywania sortowania pozycyjnego z wykorzystaniem sortowania kubełkowego.

- ◯ aktualnie sprawdzane cyfry z odpowiednich pozycji liczby
- ◯ indeksy liczb w tablicy, które są już posortowane
- ◯ liczby o takich samych cyfrach na wybranej pozycji
- ◯ liczbę cyfr każdej z sortowanych liczb

Ćwiczenie 3

Uzupełnij tekst właściwym słowem.

Jeżeli będziemy sortować liczby zapisane w systemie binarnym, wówczas algorytm sortowania pozycyjnego z wykorzystaniem sortowania kubełkowego będzie [ ] efektywny niż w przypadku zastosowania go dla innych systemów liczbowych.

bardziej | mniej

Ćwiczenie 4

Uzupełnij tekst odpowiednią cyfrą.

Podczas sortowania pozycyjnego następującej tablicy: [245, 50593, 4123, 5012, 921], sortowanie dodatkowym algorytmem stabilnym wykonamy _____ razy.

Ćwiczenie 5

Zdecyduj, czy podane zdanie jest prawdziwe, czy fałszywe.

Jeżeli będziemy sortować liczby zapisane w systemie szesnastkowym, wówczas algorytm sortowania pozycyjnego z wykorzystaniem sortowania kubełkowego będzie mniej optymalny niż przy operowaniu na liczbach zapisanych w systemie dziesiętnym.

- ◯ fałsz
- ◯ prawda

Ćwiczenie 6

Wskaż liczby, które zmienią pozycję po pierwszej iteracji sortowania pozycyjnego.

312	943	2916	55	47	111119

- ◯ 55 oraz 47
- ◯ 312 oraz 111119
- ◯ 2916 oraz 55
- ◯ 312 oraz 943

Ćwiczenie 7

Zdecyduj, czy podane zdanie jest prawdziwe, czy fałszywe.

Algorytm sortowania pozycyjnego najlepiej sprawdzi się przy sortowaniu liczb o takiej samej długości.

- ◯ fałsz
- ◯ prawda

Ćwiczenie 8

Określ, jak będzie wyglądał zbiór wynikowy po pierwszej iteracji sortowania pozycyjnego w kolejności rosnącej. Zbiór wejściowy to: [893, 7, 211, 29].

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie pozycyjne liczb

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. 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;

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),

i) struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),

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):

- Przedstawisz koncepcję algorytmu sortowania pozycyjnego.
- Wskażesz różne rodzaje zastosowań dla algorytmu sortowania pozycyjnego.
- Przeanalizujesz implementację algorytmu sortowania pozycyjnego liczb, zapisaną w postaci pseudokodu.
- Zbadasz złożoność czasową algorytmu sortowania pozycyjnego.

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 pozycyjne liczb". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel prosi wybraną osobę o odczytanie tematu lekcji, a następnie określa cele i 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.** 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 „Schemat interaktywny". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium. W przedstawionej aplikacji wykorzystywana jest funkcja ObliczLiczbęCyfrWLiczbie(). Powinna ona zwracać zmienną LiczbaCyfr – informację o tym, z ilu cyfr składa się liczba podana jako argument. Niestety, kod funkcji jest niekompletny. Zadaniem uczniów jest wpisanie brakujących instrukcji. Główna funkcja programu została już zdefiniowana.
3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1-5 z sekcji „Sprawdź się", po ich wykonaniu porównują otrzymane wyniki z inną osobą.

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. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 6-8 z sekcji „Sprawdź się".
2. Przeanalizuj implementację algorytmu zaproponowaną w sekcji „Schemat interaktywny". Rozbij ją na etapy i spróbuj dodać do niej komentarze wyjaśniające kolejne etapy działania.

Wskazówki metodyczne:

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