Sortowanie pozycyjne dat

Źródło: Jisun Han, domena publiczna.

W skomputeryzowanym świecie często zdarza się, że potrzebujemy chronologicznie uporządkować różne zbiory danych. Sprawa dodatkowo się komplikuje, gdy w ramach jednej pozycji na liście mamy do czynienia z datą, będącą w zasadzie zbiorem kilku połączonych ze sobą liczb.

Jednym ze sposobów na posortowanie tego typu danych jest wykorzystanie algorytmu sortowania pozycyjnego. W tym e-materiale nauczysz się, jak ułożyć w określony sposób zbiór dat, wykorzystując wspomniany algorytm, oraz jak stworzyć program oparty na jego podstawie.

Implementację sortowania pozycyjnego dat w poszczególnych językach programowania przedstawiamy w e-materiałach:

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

Więcej zadań? Przejdź do: Sortowanie pozycyje dat – zadania maturalne.

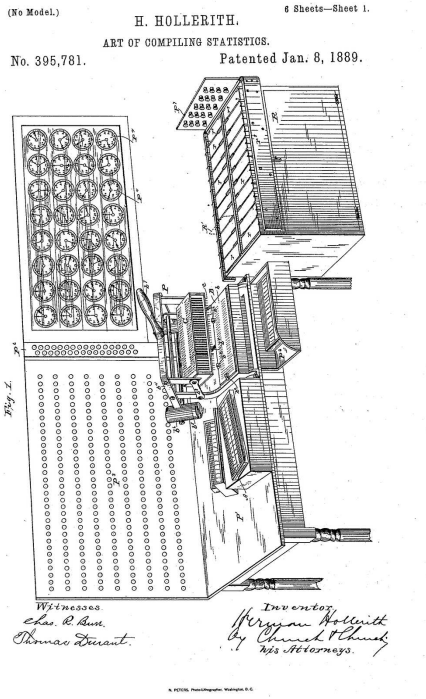
> **Twoje cele**
>
> - Posortujesz daty przy użyciu algorytmu sortowania pozycyjnego.

- Przeanalizujesz zalety i wady sortowania pozycyjnego.
- Prześledzisz algorytm sortowania pozycyjnego dat zapisany za pomocą pseudokodu.

Przeczytaj

Rys historyczny

Początki sortowania pozycyjnego sięgają XIX wieku, kiedy to w roku 1890 Herman Hollerith po raz pierwszy zastosował maszynę licząco-analityczną do opracowania spisu ludności USA. Wynalazek swój opatentował w 1889 r.


(No Model.) 6 Sheets—Sheet 1.
H. HOLLERITH.
ART OF COMPILING STATISTICS.
No. 395,781. Patented Jan. 8, 1889.
Witnesses.
Inventor.
his Attorneys.

Maszyna licząco-analityczna

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

Sortowanie pozycyjne dat – omówienie

Algorytm, którym się zajmujemy, jest bardzo podobny do algorytmu sortowania pozycyjnego słów oraz sortowania pozycyjnego liczb.

Algorytm sortowania pozycyjnego dat polega na sortowaniu dat według cyfr na kolejnych pozycjach, rozpoczynając od najmniej znaczącej (czyli cyfry jedności dnia). Aby było to możliwe, przyjmujemy format zapisu daty, obsługiwany przez algorytm. W tym materiale wybieramy standard **ISO 8601**. Jednym ze sposobów zapisu daty w tym formacie jest postać *RRRR-MM-DD*.

RRRR oznacza czterocyfrowy zapis roku, np. 1998. *MM* oznacza dwucyfrowy zapis miesiąca, np. czerwiec zapiszemy jako 06. *DD* oznacza dwucyfrowy zapis dnia w danym miesiącu, np. 19. Takim sposobem otrzymujemy jednoznaczny zapis daty: 1998-06-19 (19 czerwca 1998 r.).

Ważne jest, aby nie pomijać wiodących zer. Błędem będzie zapisanie daty 1998-06-19 jako 1998-6-19.

Przydatność tego zapisu w algorytmie wynika z następujących cech:

- cyfry są uszeregowane od najmniej znaczącej do najbardziej znaczącej (od prawej do lewej),
- długość daty jest stała i wynosi 10,
- separatory (znak myślnika) występują na z góry określonych pozycjach: jest to czwarty i siódmy indeks, przy założeniu, że liczymy od zera.

Przy tak reprezentowanej dacie algorytm sortowania pozycyjnego sprowadza się do sortowania dat według kolejnych pozycji od najmniej znaczącej do najbardziej znaczącej cyfry (od prawej do lewej), z pominięciem separatorów na odpowiednich indeksach.

Omówmy przykładowe zadanie, w którym naszym celem jest posortowanie podanego zbioru dat w kolejności niemalejącej przy użyciu algorytmu sortowania pozycyjnego.

Przykład 1

Z = {1960-03-21, 2003-04-09, 1526-01-30, 1680-07-14, 2050-11-02, 1250-12-17, 1337-09-05}

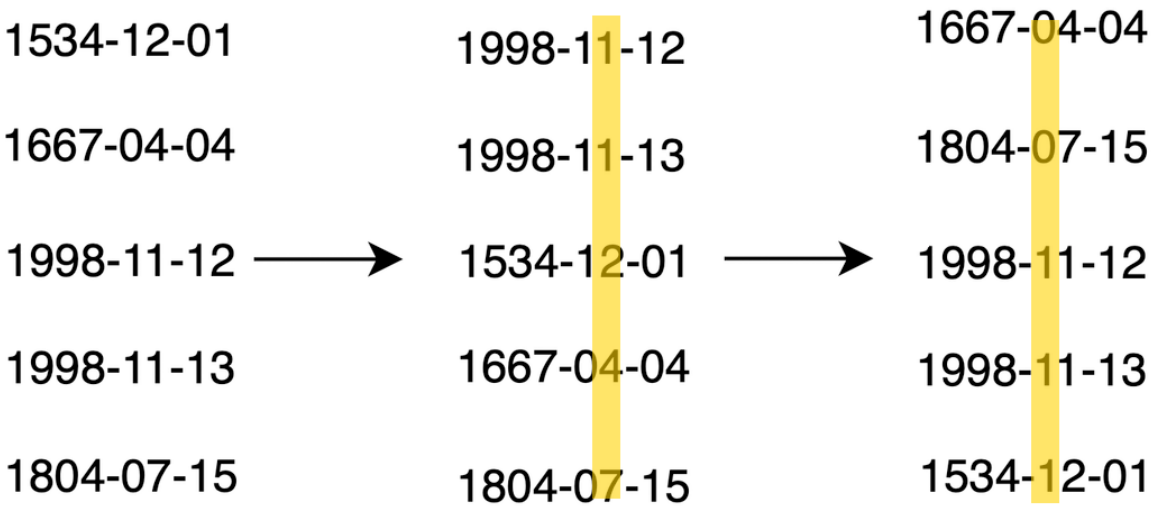
Daty przechowywane są zgodnie z wcześniej przyjętym formatem: *RRRR-MM-DD*.

Sortując daty, rozpatrujemy kolejne cyfry zapisu, rozpoczynając od najmniej znaczącej. Najpierw korzystając z pomocniczego stabilnego algorytmu sortowania, sortujemy daty kolejno według:

- cyfry jedności liczby reprezentującej dni,
- cyfry dziesiątek liczby reprezentującej dni,

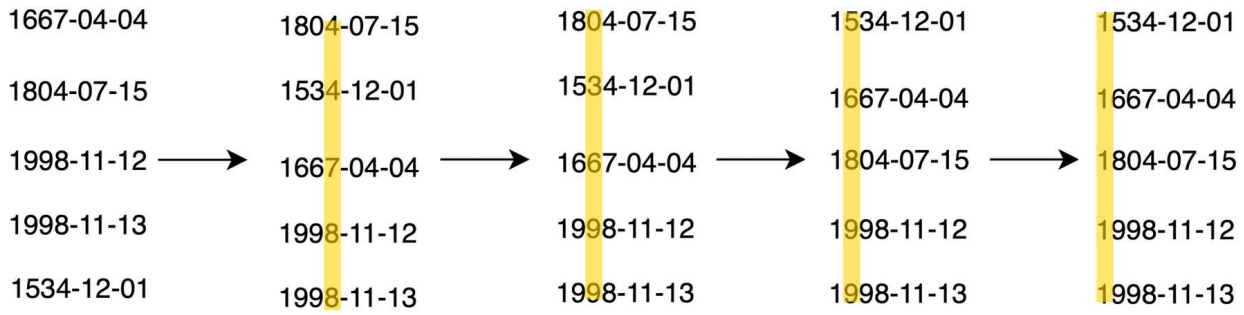
- cyfry jedności liczby reprezentującej miesiąc,

- cyfry dziesiątek liczby reprezentującej miesiąc,



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

- cyfry jedności liczby reprezentującej rok,
- cyfry dziesiątek liczby reprezentującej rok,
- cyfry setek liczby reprezentującej rok,
- cyfry tysięcy liczby reprezentującej rok.



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

Po wykonaniu ostatniego kroku lista dat jest posortowana.

Zasada działania algorytmu sortowania pozycyjnego dat

Do posortowania elementów na konkretnych, odpowiadających sobie pozycjach w datach możemy użyć dowolnego algorytmu sortowania stabilnego.

Algorytm pomocniczy musi być stabilnym algorytmem sortowania – w przeciwnym wypadku sortowanie dat według kolejnych pozycji nie miałoby sensu. Np. przy sortowaniu dat: 1998-11-05 i 1998-11-06 po poprawnym posortowaniu według cyfry jedności dni, algorytm bez gwarancji stabilności mógłby zamienić daty, sortując według pozostałych,

jednakowych cyfr. W konsekwencji: również sam algorytm sortowania pozycyjnego jest stabilny.

W tym e-materiale jako algorytm pomocniczy wybieramy sortowanie kubełkowe, ze względu na znaną z góry liczbę kubełków równą 10 (liczba cyfr w systemie dziesiętnym). Algorytm ten został dokładnie omówiony w e-materiale Sortowanie kubełkowe.

Na początku wydzielamy kubełki, które posłużą za miejsce do przechowania liczb w trakcie wykonywanych operacji. Każdy kubełek oznaczamy jedną cyfrą. Musimy mieć przewidziany oddzielny kubełek dla każdej z cyfr w zależności od zastosowanego systemu liczbowego, w którym zapisana jest data.

Kubełki służą do tego, by w momencie wykonywania operacji przechować daty, które mają tę samą cyfrę na analizowanej pozycji (czyli np. w wyniku analizy cyfr jedności w liczbie dni takie daty jak 1995-12-25 oraz 2001-09-15 znalazłyby się w jednym kubełku). Następnie „wyjmujemy" wszystkie daty z kubełków w kolejności, w jakiej je tam wstawiliśmy. Dzięki temu możemy mówić o posortowaniu liczb stabilnie według odpowiedniej cyfry (w tym przykładzie według cyfry jedności w liczbie dni).

Wielokrotne powtórzenie tego procesu dla kolejnych cyfr (od najmniej do najbardziej znaczącej) pozwoli szybko otrzymać oczekiwane rezultaty. Zasada gospodarowania kubełkami jest prosta: istnieje tyle kubełków, ile jest cyfr w analizowanym systemie liczbowym. Następnie sprawdzamy, jaka cyfra znajduje się na aktualnie rozpatrywanej pozycji, i dopasowujemy ją do kubełka.

Przykład sortowania pozycyjnego dat z użyciem sortowania kubełkowego

Posortujmy niemalejąco następujący zestaw dat:

- 1960-03-21
- 2003-04-09
- 1526-01-30
- 1680-07-14
- 2050-11-02
- 1250-12-17
- 1337-09-05

Algorytm będzie pracował na datach zapisanych w zbiorze dat. Wygląda on następująco:

A_0 = {1960-03-21, 2003-04-09, 1526-01-30, 1680-07-14, 2050-11-02, 1250-12-17, 1337-09-05}

Tak przygotowany zbiór dat możemy zacząć sortować przy użyciu pomocniczego, stabilnego algorytmu sortowania kubełkowego. Umieszczamy daty w kubełkach na

podstawie aktualnie sprawdzanej cyfry. Zaczynamy od ostatniej cyfry daty, czyli mniej znaczącej cyfry opisującej dzień:

Kubełek	0	1	2	3	4	5
Zawartość	1526-01-30	1960-03-21	2050-11-02		1680-07-14	1337-09-05

Następnie wyjmujemy daty z kubełków, zaczynając od kubełka 0:

A_1 = {1526-01-30, 1960-03-21, 2050-11-02, 1680-07-14, 1337-09-05, 1250-12-17, 2003-04-09}

Posortowaliśmy daty według cyfry jedności dnia. Teraz powtarzamy proces dla przedostatniej pozycji, sortując przy tym daty według cyfr dziesiątek w numerze oznaczającym dzień:

Kubełek	0	1	2	3	4	5	6	7
Zawartość	2003-04-09 1337-09-05 2050-11-02	1250-12-17 1680-07-14	1960-03-21	1526-01-30				

Ponownie wyjmujemy daty z każdego kubełka. Niektóre kubełki zawierają więcej niż jedną wartość. W takim przypadku ważne jest, abyśmy wyciąganie zaczynali od daty umieszczonej najwcześniej, czyli zapisanej najniżej. Jest to istotne dla zachowania zasady sortowania stabilnego:

A_2 = {2050-11-02, 1337-09-05, 2003-04-09, 1680-07-14, 1250-12-17, 1960-03-21, 1526-01-30}

Następnie przetwarzamy pozycję, na której w przyjętym zapisie daty znajduje się znak „-”. Pomijamy go i przechodzimy do przetwarzania kolejnej pozycji.

Docieramy do porządkowania według cyfry jedności liczby miesiąca:

Kubełek	0	1	2	3	4	5	6	
Zawartość		1526-01-30 2050-11-02	1250-12-17	1960-03-21	2003-04-09			168

Umieszczamy daty w tablicy zgodnie z zasadami:

A_3 = {2050-11-02, 1526-01-30, 1250-12-17, 1960-03-21, 2003-04-09, 1680-07-14, 1337-09-05}

Ponownie umieszczamy wszystkie daty w kubełkach, tym razem zwracając uwagę na bardziej znaczącą cyfrę miesiąca:

Kubełek	0	1	2	3	4	5	6	7	8	9
Zawartość	1337-09-5 1680-07-14 2003-04-09 1960-03-21 1526-01-30	1250-12-17 2050-11-02								

Zbiór dat ma obecnie postać:

A_4 = {1526-01-30, 1960-03-21, 2003-04-09, 1680-07-14, 1337-09-05, 2050-11-02, 1250-12-17}

Ponownie na kolejnej pozycji znajduje się znak „-”, więc pomijamy go.

Następnie przechodzimy do sortowania według cyfry jedności roku:

Kubełek	0	1	2	3	4	5	6	7
Zawartość	1250-12-17 2050-11-02 1680-07-14 1960-03-21			2003-04-09			1526-01-30	1337-09-05

Ponownie wyjmujemy wszystkie daty. Teraz zbiór dat prezentuje się następująco:

A_5 = {1960-03-21, 1680-07-14, 2050-11-02, 1250-12-17, 2003-04-09, 1526-01-30, 1337-09-05}

Wykonanie algorytmu dla cyfry dziesiątek roku poskutkuje powstaniem kubełków:

Kubełek	0	1	2	3	4	5	6
Zawartość	2003-04-09		1526-01-30	1337-09-05		1250-12-17 2050-11-02	1960-0

Otrzymujemy następującą listę:

A_6 = {2003-04-09, 1526-01-30, 1337-09-05, 2050-11-02, 1250-12-17, 1960-03-21, 1680-07-14}

Przedostatni przebieg algorytmu (dla cyfry setek roku) utworzy następującą strukturę:

Kubełek	0	1	2	3	4	5	6
Zawartość	2050-11-02 2003-04-09		1250-12-17	1337-09-05		1526-01-30	1680-07

Po wypisaniu dat w odpowiedniej kolejności otrzymujemy:

A_7 = {2003-04-09, 2050-11-02, 1250-12-17, 1337-09-05, 1526-01-30, 1680-07-14, 1960-03-21}

Pozostaje ostatnie powtórzenie, w którym uwzględniamy cyfrę tysięcy roku:

Kubełek	0	1	2	3	4	5	6	7	8	9
Zawartość		1960-03-21 1680-07-14 1526-01-30 1337-09-05 1250-12-17	2050-11-02 2003-04-09							

Wynikowy zbiór ma postać:

$A_{wyjściowa}$ = {1250-12-17, 1337-09-05, 1526-01-30, 1680-07-14, 1960-03-21, 2003-04-09, 2050-11-02}

Jak widać, algorytm spełnił swoje zadanie, wykonując tylko osiem powtórzeń algorytmu sortowania kubełkowego (tyle, ile jest cyfr w dacie).

Pseudokod

Zgodnie z założeniami pseudokod będzie działał na datach zapisanych w formacie *RRRR-MM-DD*. Dla innego formatu należałoby go odpowiednio dostosować.

Specyfikacja problemu:

Dane:

- `rozmiar` – liczba całkowita przechowująca informację dotyczącą liczby sortowanych dat
- `tablica[0..rozmiar - 1]` – tablica jednowymiarowa przechowująca sortowane daty w formie napisu; daty w tablicy przechowywane są jako napisy w formacie *RRRR-MM-DD*

Wynik:

- `tablica[0..rozmiar - 1]` – zawiera daty posortowane chronologicznie (od najwcześniejszej do najpóźniejszej)

Aby posortować daty pozycyjnie, utworzymy pętlę `dla` uwzględniającą wszystkie pozycje, według których powinniśmy posortować daty w tablicy `tablica[]`. Jest ich 10. Pominiemy również separatory znajdujące się w każdej dacie na indeksach 4 i 7 (licząc od 0).

```
dla pozycja = 9, 8, ..., 0 wykonuj:
  jeżeli pozycja ≠ 4 i pozycja ≠ 7:
    sortowanie_kubełkowe(pozycja)
```

Jak widać, skorzystaliśmy z pomocniczego sortowania kubełkowego. Przedstawmy jego zapis w postaci pseudokodu.

Niech tablica `kubełki[0..9]` będzie tablicą list tymczasowo przechowujących rozpatrywane aktualnie daty. Lista jest obiektem, który działa na podobnej zasadzie co tablica, lecz nie musi mieć stałego rozmiaru – można zarówno dopisywać, jak i wyciągać z niej kolejne liczby.

Funkcja `kod(znak)` zamienia znak `znak` na jego kod ASCII.

```
funkcja sortowanie_kubełkowe(pozycja):
  dla n = 0, 1, ..., rozmiar - 1 wykonuj:
    // konwertuj znak tablica[n][pozycja] na liczbę z zakresu <0,
    cyfra ← kod(tablica[n][pozycja]) - kod("0")
    umieść wewnątrz kubełki[cyfra] wartość tablica[n]

  x ← 0
  dla cyfra = 0, 1, ..., 9 wykonuj
    dopóki kubełki[cyfra] zawiera liczby wykonuj
      tablica[x] ← pobierz najwcześniej dodaną wartość z kubełki[
      x ← x + 1
```

Cechy sortowania pozycyjnego dat

Sortowanie pozycyjne jest wydajną metodą sortowania liczb naturalnych o podobnej liczbie cyfr. Dlatego też świetnie nadaje się do sortowania dat, które w większości przypadków będą liczbami dokładnie ośmiocyfrowymi (z pominięciem ewentualnych separatorów takich jak myślniki czy kropki).

Sortowanie pozycyjne jest stabilne. A zatem jeśli w zbiorze wejściowym wystąpią powtarzające się daty, ich kolejność względem siebie się nie zmieni.

Oszacujmy złożoność czasową tego algorytmu sortowania. Wprowadźmy następujące oznaczenia:

- N – liczba dat do posortowania,
- d – liczba cyfr w danym formacie daty bez separatorów.

Rozpoczynamy od analizy liczby operacji dodawania i usuwania dat z kubełków w funkcji `sortowanie_kubełkowe()`. Instrukcje dodawania elementów do kubełków wykonają się N razy. Instrukcje usuwania elementów z kubełka umieszczone w pętli `dopóki` wykonają się również N razy (tyle, ile liczb znalazło się sumarycznie w kubełkach). Zatem złożoność obliczeniowa tej funkcji to $2 \cdot N$.

Główna pętla programu wykona się $d + 2$ razy. Pomijając separatory, będzie to d.

W niepominiętych iteracjach (ich liczba wynosi d) wywołana zostanie funkcja `sortowanie_kubełkowe()`, której złożoność to $2 \cdot N$. Zatem złożoność całego programu to $O(d \cdot N)$.

Jeżeli przyjmiemy d jako stałą równą 8, to sortowanie pozycyjne dat ma złożoność liniową.

Słownik

lista

struktura danych służąca do reprezentacji zbiorów dynamicznych, w wykorzystaniu praktycznym różni się od tablicy tym, że jej elementy można na bieżąco modyfikować

stabilny algorytm sortowania

algorytm sortowania gwarantujący zachowanie kolejności elementów o tej samej wartości w tablicy wynikowej, identycznej jak w tablicy wejściowej

złożoność czasowa

cecha algorytmu określająca tendencję, z jaką rośnie czas potrzebny na wykonanie algorytmu wraz ze wzrostem rozmiaru danych

Schemat interaktywny

Polecenie 1

Czasem sortowane daty nie posiadają zer poprzedzających liczbę dni czy miesięcy – może to powodować sytuację, w której daty będą miały różną długość. Twoim zadaniem jest dodanie zer do liczby dni lub miesięcy, jeśli zajdzie taka potrzeba, a także zamiana znaku „.” na znak „-”. Data „5.11.2020” powinna zamienić się w datę „05-11-2020”.

Uzupełnij pętlę warunkową. Powinna ona zapisać do zmiennej `wynik` prawidłową postać modyfikowanej daty.

Specyfikacja problemu:

Dane:

- `data` – łańcuch znaków; data zapisana w formacie `dzień.miesiąc.rok`; nie zawiera zer nieznaczących

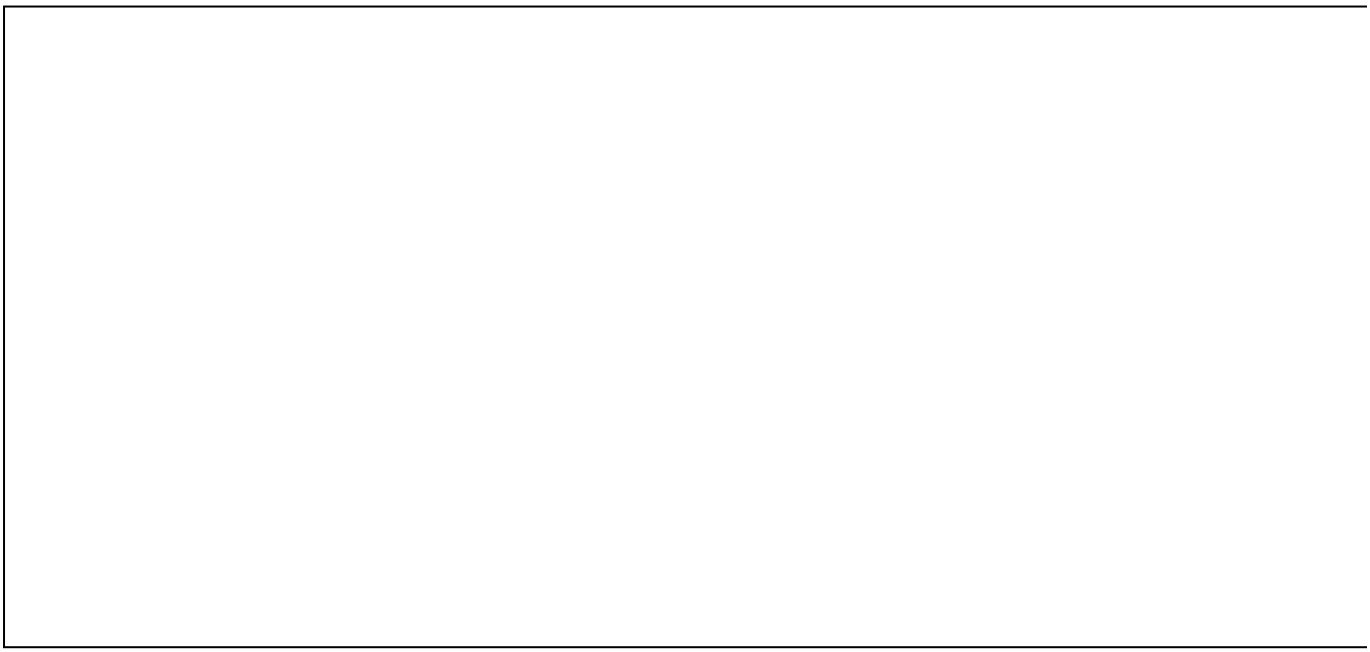
Wynik:

- `wynik` – łańcuch znaków; data zapisana w formacie `dzień-miesiąc-rok`

Polecenie 2

Zapisz program z polecenia 1 w wybranym języku programowania: C++, Java, Python.

```
1
```



Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zdecyduj, czy zdania są prawdziwe czy fałszywe.

Stwierdzenie	Prawda	Fałsz
Dla sortowania pozycyjnego nie istnieją przypadki optymistyczne oraz pesymistyczne, które zachodziłyby wraz z innym ułożeniem liczb w tablicy.	◯	◯
Sortowanie pozycyjne dat nie jest algorytmem stabilnym, ponieważ nie stosuje przypisania stałego indeksu do każdej z sortowanych liczb.	◯	◯

Ćwiczenie 2

Wskaż, jaka będzie kolejność sortowanych dat dla siódmej iteracji algorytmu. Zbiór danych do posortowania to: `["2010-05-15", "1945-12-12", "1968-04-19", "1459-08-24", "1852-01-01"]`.

◯ `["1852-01-01", "1945-12-12", "1459-08-24", "2010-05-15", 1968-04-19"]`

◯ `["1852-01-01", "1945-12-12", 1968-04-19", "2010-05-15", "1459-08-24"]`

◯ `["2010-05-15", "1852-01-01", "1945-12-12", 1968-04-19", "1459-08-24"]`

◯ `["1852-01-01", "1945-12-12", "2010-05-15", 1968-04-19", "1459-08-24"]`

Ćwiczenie 3

Wskaż, które daty zostaną wyciągnięte z kubełka w pierwszej kolejności, jeśli daty sortowane są pozycyjnie przy użyciu pomocniczego algorytmu sortowania kubełkowego.

- ◯ umieszczone jako ostatnie
- ◯ umieszczone jako pierwsze

Ćwiczenie 4

Uzupełnij tekst właściwymi fragmentami.

Algorytm sortowania pozycyjnego dat określany jest jako [ ], ponieważ pierwotna kolejność liczb (na podstawie indeksu) o tej samej wartości [ ] zachowana.

jest | nie jest | stabilny | niestabilny

Ćwiczenie 5

Daty reprezentowane są jako ciągi znaków.

Zdecyduj, czy podane stwierdzenie jest prawdziwe czy fałszywe.
Algorytm nigdy nie zadziała, jeżeli w tablicy umieścimy daty zawierające rok o różnej liczbie cyfr.

- ◯ fałsz
- ◯ prawda

Ćwiczenie 6

Wskaż, do czego służą kubełki w trakcie sortowania pozycyjnego, w którym sortowanie kubełkowe jest algorytmem pomocniczym. Zaznacz wszystkie poprawne odpowiedzi.

- [ ] pełnienie funkcji zbioru zmiennych pomocniczych
- [ ] uporządkowanie liczb według aktualnie sprawdzanej pozycji
- [ ] rozbicie daty na trzy elementy: rok, miesiąc oraz dzień
- [ ] przechowanie indeksów zamienianych ze sobą dat

Ćwiczenie 7

W tym zadaniu format daty to *RRRRMMDD*. Jeżeli rok jest krótszy niż czterocyfrowy, uzupełniamy go zerami wiodącymi.

Dokończ zdanie.
Po wykonaniu pierwszej iteracji sortowania pozycyjnego dokonamy przestawienia następujących wartości...

20771211	14100714	9661033	32491025

- ( ) 0966-10-33 oraz 3249-10-25.
- ( ) 1410-07-14 oraz 0966-10-33.
- ( ) 2077-12-11 oraz 0966-10-33.

Ćwiczenie 8

Uzupełnij zdanie, wpisując w puste miejsce odpowiednią wartość.

Program ma za zadanie posortować istotne daty z okresu renesansu. Format, w jakim dane zostały wprowadzone, to *RRRR-MM-DD*. Załóżmy, że główna pętla ma za zadanie uwzględnić wszystkie znaki w zadanym formacie daty (również myślniki). W takim razie minimalna liczba iteracji tej pętli wynosi _____.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie pozycyjne dat

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;

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:

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),

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

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):

- Posortujesz daty przy użyciu algorytmu sortowania pozycyjnego.
- Przeanalizujesz zalety i wady sortowania pozycyjnego.
- Prześledzisz algorytm sortowania pozycyjnego dat zapisany za pomocą pseudokodu.

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 dat". Uczniowie zapoznają 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.** 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". W razie potrzeby nauczyciel wyjaśnia niezrozumiałe kwestie.
2. **Praca z multimedium.** Nauczyciel czyta polecenie nr 1 z sekcji „Schemat interaktywny". Prosi uczniów, aby wykonali je w parach. Następnie wybrana osoba prezentuje propozycję odpowiedzi, a pozostali uczniowie ustosunkowują się do niej. Nauczyciel w razie potrzeby uzupełnia ją, udziela też uczniom informacji zwrotnej.
3. **Ćwiczenie umiejętności.** Liga zadaniowa - uczniowie wykonują indywidualnie na czas ćwiczenia nr 1-5 z sekcji „Sprawdź się", a następnie omawiają zadania na forum.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 6-8 z sekcji „Sprawdź się".
2. Uczniowie zapisują program wykonany w sekcji „Schemat interaktywny" w wybranym przez siebie języku programowania.

Wskazówki metodyczne:

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