Sortowanie przez wybieranie w języku Python

Źródło: Brienne Hong, domena publiczna.

Sortowanie przez wybieranie to jeden z najprostszych algorytmów sortowania. W tym e-materiale zajmiemy się jego implementacją w języku Python.

Podstawowe informacje na temat tego algorytmu znajdziesz w e-materiale Sortowanie przez wybieranie.

Implementacje w pozostałych językach programowania zostały omówione w e-materiałach:

- Sortowanie przez wybieranie w języku C++,
- Sortowanie przez wybieranie w języku Java.

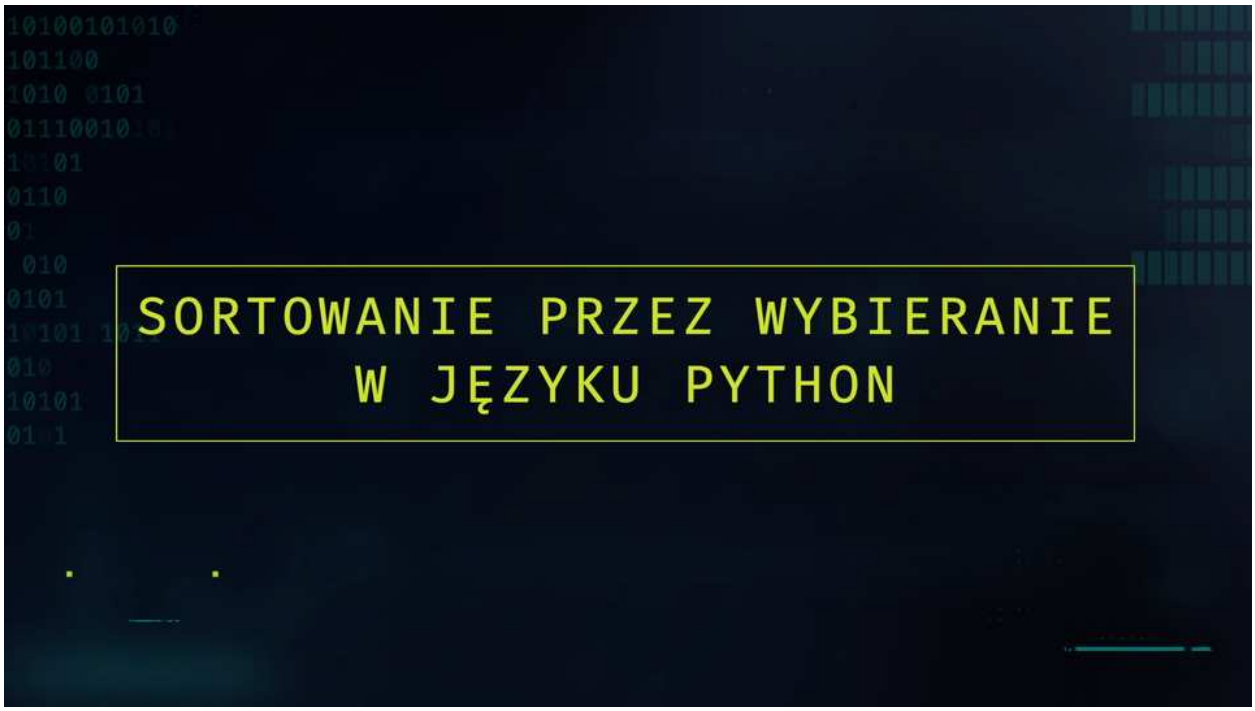
Więcej zadań? Przejdź do: Sortowanie przez wybieranie – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz implementację algorytmu sortowania przez wybieranie w języku Python.
> - Zaimplementujesz algorytm sortowania przez wybieranie w języku Python.
> - Rozwiążesz kilka zadań z wykorzystaniem sortowania przez wybieranie.

Animacja

> **Polecenie 1**
>
> Zapoznaj się z animacją. Poszukaj innych filmów, w których algorytmy sortowania zaprezentowano za pomocą tańca. Zastanów się, czy można w podobnym celu wykorzystać jakiś polski taniec.

> **Ważne!**
>
> W animacji dane sortowane są rosnąco.



Film dostępny pod adresem /preview/resource/R16V2RnZXTNY4

Film nawiązujący do treści materiału: Sortowanie przez wybieranie w języku Python.

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi treści animacji.

Na podstawie animacji oceń, czy przedstawiony algorytm nadawałby się do posortowania dużego zbioru danych, które byłyby posortowane w porządku odwrotnym do oczekiwanego. Uzasadnij swoje zdanie.

Przeczytaj

Implementacja algorytmu sortowania przez wybieranie w języku Python

Problem 1

Napisz program sortujący niemalejąco n-elementową tablicę `dane`, zawierającą liczby naturalne. W swojej implementacji wykorzystaj algorytm sortowania przez wybieranie. Przetestuj działanie programu dla następujących danych:

- `n = 7`
- `dane = [5, 7, 86, 32, 64, 54, 2]`

Specyfikacja problemu:

Dane:

- n – liczba naturalna dodatnia; rozmiar tablicy `dane`
- `dane` – n-elementowa tablica liczb naturalnych

Wynik:

Program wypisuje wszystkie elementy tablicy `dane`, posortowane w kolejności niemalejącej.

```
# Tutaj dodaj własny kod. Użyj funkcji
# print()
```

```
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w prezentacji.

1

Zaimplementujemy algorytm w języku Python, wykorzystując sortowanie przez wybieranie. Za jego pomocą posortujemy niemalejąco tablicę liczb naturalnych `dane` dla przykładowych danych podanych w treści zadania.

Zaczynamy od inicjacji zmiennych.

```
1 dane = [5, 7, 86, 32, 64,
  54, 2]
2 n = 7
```

2

Następnie, zgodnie z algorytmem sortowania przez wybieranie, zapisujemy pętlę `for` przechodzącą po wszystkich elementach tablicy. Odpowiada ona wybieraniu kolejnego najmniejszego elementu i wstawianiu go na miejsce w tablicy o danym indeksie.

```
1 dane = [5, 7, 86, 32, 64,
  54, 2]
2 n = 7
3
4 for i in range(n):
```

3

Wewnątrz pętli deklarujemy dodatkową zmienną, która będzie przechowywała indeks najmniejszego elementu w podtablicy, oraz kolejną pętlę `for`, odpowiadającą za sprawdzenie wszystkich elementów podtablicy. Ponieważ przeszukujemy nieposortowany fragment tablicy, a liczba znajdująca się na miejscu o indeksie `i` już została uwzględniona, iterację zaczynamy od indeksu `i + 1`.

```
1 dane = [5, 7, 86, 32, 64,
  54, 2]
2 n = 7
3
4 for i in range(n):
5
6     indeks_najmniejszego =
  i
7     for j in range(i + 1,
  n):
```

4

Sprawdźmy, czy liczba, która znajduje się na miejscu o indeksie `j`, jest mniejsza od liczby

znajdującej się na miejscu o indeksie `indeks_najmniejszego`. Jeżeli tak, zapisujemy indeks `j` do zmiennej `indeks_najmniejszego`.

```
dane = [5, 7, 86, 32, 64, 54, 2]
n = 7

for i in range(n):

    indeks_najmniejszego = i
    for j in range(i + 1, n):
        if dane[j] < dane[indeks_najmniejszego]:

indeks_najmniejszego = j

```

5

Po znalezieniu najmniejszej wartości zamieniamy miejscami elementy oznaczone indeksami `i` oraz `indeks_najmniejszego`.

```
dane = [5, 7, 86, 32, 64, 54, 2]
n = 7

for i in range(n):

    indeks_najmniejszego = i
    for j in range(i + 1, n):
        if dane[j] < dane[indeks_najmniejszego]:
```

```
9
indeks_najmniejszego = j
10
11      dane[i],
dane[indeks_najmniejszego]
=
dane[indeks_najmniejszego]
, dane[i]
12
```

6

Na koniec wyświetlamy posortowane elementy tablicy.

```
1  dane = [5, 7, 86, 32, 64,
   54, 2]
2  n = 7
3
4  for i in range(n):
5
6      indeks_najmniejszego =
   i
7      for j in range(i + 1,
   n):
8          if dane[j] <
   dane[indeks_najmniejszego]
   :
9
   indeks_najmniejszego = j
10
11     dane[i],
   dane[indeks_najmniejszego]
   =
   dane[indeks_najmniejszego]
   , dane[i]
12
13 for i in range(n):
14     print(dane[i])
15
```

Analiza złożoności czasowej i pamięciowej

Aby wyznaczyć **złożoność czasową** algorytmu sortowania przez wybieranie, przeanalizujmy liczbę operacji podstawowych. Są nimi m.in. porównania i zamiany elementów.

Aby obliczyć liczbę porównań, zastanówmy się, ile razy wykona się wewnętrzna pętla w linii siódmej. W pierwszej iteracji wartość `j` będzie przyjmować wartości: `1`, `2`, ..., `n - 1`. Z każdą kolejną iteracją pętli zewnętrznej pętla wewnętrzna będzie wykonywać o jedną iterację mniej. Możemy zatem policzyć, że pętla ta będzie wykonywać się kolejno `n - 1`, `n - 2`, ..., `1` razy. Sumując wyrażenia za pomocą wzoru na sumę ciągu arytmetycznego, otrzymamy liczbę porównań równą $\frac{n(n-1)}{2}$. Liczba zamian jest równa $n - 1$. Złożoność czasowa algorytmu sortowania przez wybieranie jest zatem kwadratowa. W notacji wielkiego O będzie to złożoność czasowa $O(n^2)$.

> Ćwiczenie 3 w sekcji „Sprawdź się” pozwoli na weryfikację tego wzoru.

Wyznaczenie **złożoności pamięciowej** opiera się na obserwacji, że nie używamy dodatkowej pamięci ponad stałą liczbę zmiennych i tablicę danych wejściowych. Jest ona stała, co w notacji wielkiego O zapisujemy jako $O(1)$. Algorytm sortowania przez wybieranie jest zatem algorytmem sortowania w miejscu.

Słownik

> **operacja podstawowa**
>
> inaczej operacja dominująca; operacja wykonywana wielokrotnie, której nie możemy podzielić na mniejsze elementy; przykładem takich operacji są m.in. podstawowe operacje arytmetyczne (dodawanie, odejmowanie mnożenie, dzielenie), porównania czy inkrementacje
> **porządek sortowania**
>
> określa kryteria sortowania dla określonych danych
> **sortowanie**
>
> porządkowanie elementów zbioru ze względu na wybraną ich cechę (np. wiek, wysokość, kolejność alfabetyczną itp.)
> **tablica**
>
> struktura danych, której zadaniem jest przechowywanie w zorganizowany sposób zbioru danych tego samego typu, dostępnych za pomocą indeksu (klucza)

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który wykorzystując sortowanie przez wybieranie, poda x najmniejszych liczb z n-elementowej tablicy liczb naturalnych `dane`. Swój program przetestuj dla następujących danych:

- `n = 17`
- `dane = [696, 587, 501, 811, 162, 980, 18, 207, 638, 49, 329, 316, 305, 406, 25, 16, 542]`
- `x = 5`

Specyfikacja problemu:

Dane:

- n – liczba elementów tablicy `dane`; liczba naturalna
- `dane` – n-elementowa tablica liczb naturalnych
- x – liczba najmniejszych liczb z tablicy `dane`, które program powinien wypisać; liczba naturalna; $x \leq n$.

Wynik:

Program wypisuje x najmniejszych liczb z n-elementowej tablicy `dane`. Liczby powinny być wypisane w jednej linii i oddzielone znakiem spacji.

Twoje zadania

1. Program wypisuje x najmniejszych liczb z n-elementowej tablicy `dane`.

Ćwiczenie 2

Napisz program, który sortuje niemalejąco n-elementową tablicę `dane` z użyciem algorytmu sortowania przez wybieranie i wypisuje dwie liczby naturalne będące liczbą porównań i liczbą przestawień wykonaną przez algorytm. Swój program przetestuj dla następujących zestawów danych:

- `n1 = 17`,
  `dane1 = [8, 19, 35, 67, 76, 101, 112, 124, 147, 153, 178, 199, 216, 235, 245, 261, 281]`
- `n2 = 17`,
  `dane2 = [274, 258, 247, 227, 210, 190, 174, 166, 152, 123, 108, 98, 68, 57, 47, 22, 9]`
- `n3 = 17`,
  `dane3 = [22, 271, 115, 199, 95, 260, 265, 161, 186, 151, 256, 255, 224, 36, 138, 142, 236]`
- `n4 = 7`, `dane4 = [117, 151, 191, 190, 239, 128, 139]`

Specyfikacja problemu:

Dane:

- `n` – liczba elementów w tablicy `dane`
- `dane` – n-elementowa tablica liczb naturalnych do posortowania

Wynik:

Program wypisuje dwie liczby: liczbę porównań oraz liczbę przestawień wykonanych w trakcie działania algorytmu sortowania. Liczby powinny być wypisane w jednej linii i oddzielone znakiem spacji.

> **Uwaga!**
>
> Program powinien wypisać po dwie liczby dla każdego zestawu danych, co daje w sumie osiem liczb wypisanych w czterech liniach.

> **Przykład 1**
>
> Dla zestawu danych:
>
> - `n = 4`
> - `dane = [15, 17, 5, 4]`
>
> program powininen wypisać: 6 3.
>
> Algorytm kolejno wykonuje następujące operacje:
>
> - porównanie 17 z 15
> - porównanie 5 z 15

- porównanie 4 z 5
- zamiana 15 z 4
- porównanie 5 z 17
- porównanie 15 z 5
- zamiana 17 z 5
- porównanie 15 z 17
- zamiana 17 z 15

Twoje zadania

1. Program wypisuje dwie liczby dla każdego zestawu danych: liczbę porównań i liczbę przestawień wykonanych w trakcie działania algorytmu sortowania.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Sortowanie przez wybieranie w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy

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:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

c) porządkowania ciągu liczb: przez wstawianie i metodą bąbelkową,

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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 implementację algorytmu sortowania przez wybieranie w języku Python.
- Zaimplementujesz algorytm sortowania przez wybieranie w języku Python.
- Rozwiążesz kilka zadań z wykorzystaniem sortowania przez wybieranie.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Sortowanie przez wybieranie w języku Python". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Animacja".

Faza wstępna:

1. Wyświetlenie przez nauczyciela tematu i celów lekcji. Określenie wiążących dla uczniów kryteriów 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 multimedium.** Nauczyciel wyświetla zawartość sekcji „Animacja". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium. Zapisują ewentualne problemy i pytania. Po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
2. **Praca z tekstem.** Nauczyciel, odwołując się do pracy uczniów przed lekcją, prosi wybrane osoby o zaprezentowanie rozwiązań problemu 1 z sekcji „Przeczytaj". Nauczyciel komentuje zaprezentowane programy, odpowiada na ewentualne pytania uczniów.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1, a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Python.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- W ramach projektu można uczniom zaproponować odtworzenie zaprezentowanego w animacji tańca lub przygotowanie własnego układu, który prezentowałby algorytm sortowania przez wybieranie.