

3.7 Wybieranie, sortowanie

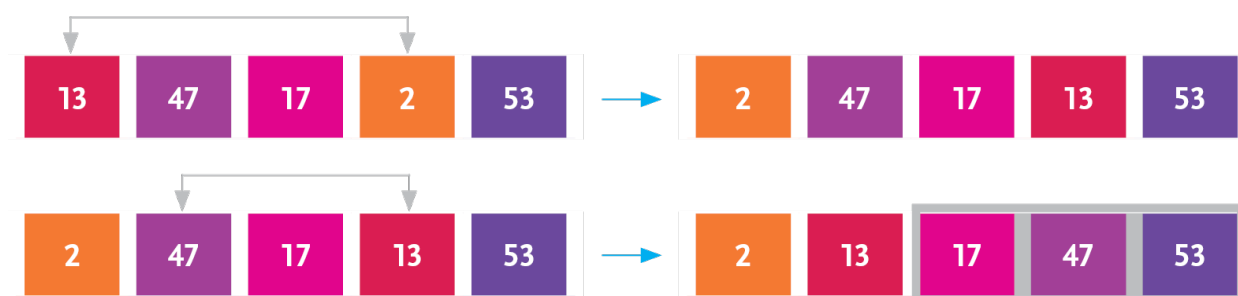
DOWIESZ SIĘ

- na czym polega sortowanie przez wybieranie,
- jak rozbić algorytm sortowania na procedury,
- jak zrealizować algorytm porządkowania przez wybieranie w Scratchu.

Sortowanie przez zliczanie to jeden z algorytmów sortowania, działający na niewielkich liczbach całkowitych. Teraz pora na bardziej uniwersalny sposób porządkowania, który może być stosowany dla dowolnych liczb – sortowanie przez wybieranie.

ZASADY SORTOWANIA PRZEZ WYBIERANIE

Aby uporządkować dany ciąg liczb rosnąco za pomocą metody sortowania przez wybieranie, należy znaleźć najmniejszą liczbę i zamienić ją miejscami z pierwszą. Potem należy wrócić do przeglądania ciągu, ale od drugiej pozycji – znaleźć kolejną najmniejszą liczbę i zamienić ją miejscami z drugą liczbą w ciągu. I tak dalej.



To jest rozwiązanie – kolejne najmniejsze liczby stoją już na swoich miejscach!

Rys. 1. Porządkowanie ciągu 13, 47, 17, 2, 53 w kolejności rosnącej

Na czym polega sortowanie przez wybieranie

Sortowanie przez wybieranie polega na przeglądaniu zbioru, wyszukiwaniu i zamianie miejscami kolejnych elementów mających się znaleźć na żądanej pozycji. Aby otrzymać zbiór uporządkowany rosnąco, zamienia się miejscami minimum z wartością, która znajduje się na pierwszej pozycji, a następnie powtarza wyszukiwanie i zamianę dla elementów zbioru. Aby otrzymać zbiór uporządkowany malejąco, wyszukuje się i zamienia miejscami maksimum.

A zatem lista kroków, które trzeba wykonać, żeby posortować dany ciąg liczb przez prosty wybór, jest następująca:

- ustaw wskaźnik na pozycji pierwszej liczby;
- wykonuj, aż wskaźnik dojdzie do końca ciągu:
 - znajdź najmniejszą liczbę podczas przeglądania ciągu od pozycji wskaźnika do końca,
 - przestaw ją na pozycję wskaźnika (zamień miejscami z liczbą, która tam jest),
 - przesun wskaźnik na następną pozycję w ciągu.

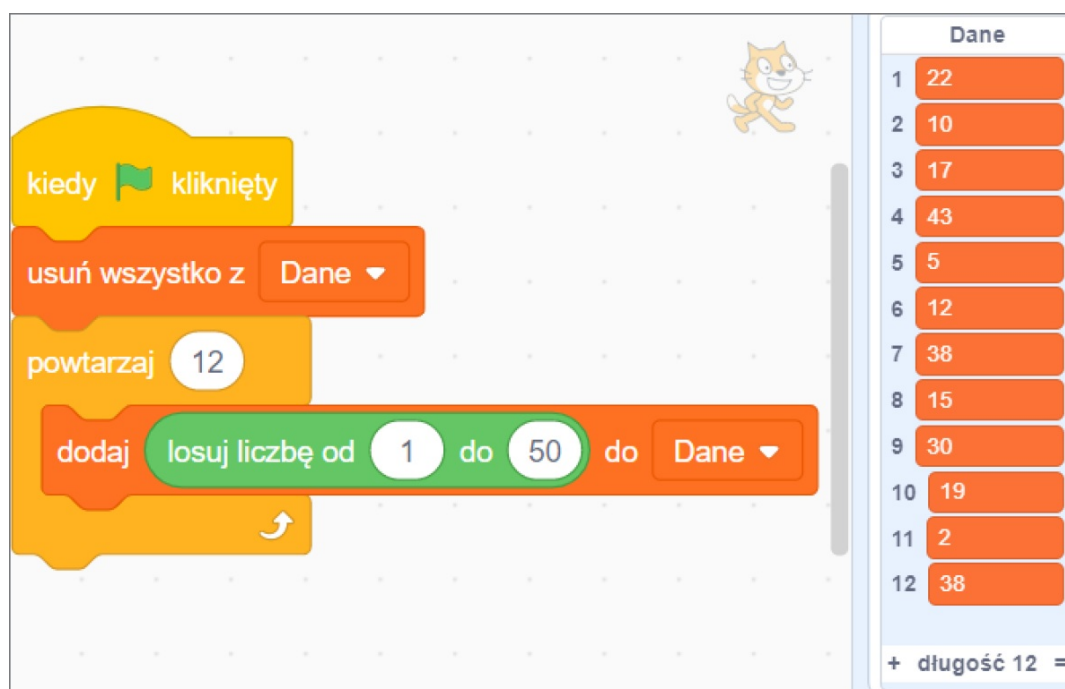
Jeśli trzeba zapisać ten algorytm w postaci skryptu czy programu, to dobrze rozbić go na dodatkowe etapy. Warto najpierw przygotować:

- skrypt znajdowania najmniejszej liczby ciągu i pozycji, na której się ona znajduje (zaczynij od pozycji wskaźnika),
- skrypt przestawiania – zamiany miejscami liczb o wybranych pozycjach.

REALIZACJA ALGORYTMU W SCRATCHU

Pracę w Scratchu zacznij od przygotowania ciągu liczb do sortowania. Ogranicz się do 12 liczb z zakresu od 1 do 50, aby można było oglądać na scenie cały ciąg liczb. Duszka zostaw na scenie – może się przydać na końcu, żeby poinformować o ukończeniu sortowania.

- Utwórz listę **Dane** i ułóż skrypt zielonej flagi, za pomocą którego dodasz do niej 12 razy kolejne losowane liczby z zakresu od 1 do 50. Pamiętaj, aby wstawić na początku blok czyszczący listę.



Rys. 2. Losowanie liczb do sortowania

Liczby na liście mają numery – na powyższym zrzucie najmniejsza jest liczba 2, na pozycji 11. Pierwszym krokiem sortowania przez wybieranie powinno więc być przestawienie liczb na pozycjach 1 i 11.

- Utwórz blok **przestaw**, który dokona zamiany. Powinien on mieć dwa parametry liczbowe – numery przestawianych elementów, np. **nr1liczby** i **nr2liczby**.

- Zdefiniuj blok **przestaw**. Aby dokonać przestawienia elementów, trzeba zapamiętać podmieniany element – utwórz zmienną **zast** i przypisz jej wartość tego elementu. Następnie wstaw dwa bloki **zamień (...)** z (...) na (...) – pierwszy zamieni element **nr1liczby** na element **nr2liczby**, a drugi zamieni element **nr2liczby** na wartość zmiennej **zast**.
- Wypróbuj działanie bloku **przestaw** – dokonaj zamiany odpowiednich elementów (w tym przypadku elementów o numerach 1 i 11).

The image shows a Scratch-like programming environment. On the left, a script area contains a pink 'definiuj przestaw nr1liczby nr2liczby' block. Below it are three orange blocks: 'ustaw zast na element nr1liczby z Dane', 'zamień nr1liczby z Dane na element nr2liczby z Dane', and 'zamień nr2liczby z Dane na zast'. To the right, a 'przestaw 1 11' block is shown. On the far right, a 'Dane' list contains 12 elements: 2, 10, 17, 43, 5, 12, 38, 15, 30, 19, 22, 38. The length of the list is shown as 12.

Rys. 3. Blok przestawiający dwa elementy listy i efekt zamiany elementów 1 i 11

- Przejrzyj ciąg z rys. 3. Jakie parametry powinien mieć blok **przestaw** w drugim kroku? Najmniejsza wartość (5) znajduje się na pozycji 5, a więc 2 i 5. Wykonaj to przestawienie. A w trzecim kroku? 3 i...
- Pora zbudować blok **numerMin** wyszukujący najmniejszą liczbę. Powinien on mieć parametr mówiący, od której pozycji trzeba przeszukiwać ciąg – możesz go nazwać np. **nrStart**.
- Zdefiniuj blok **numerMin**. Zauważ, że potrzebne są trzy nowe zmienne:
 - **nrMin** – przechowująca numer najmniejszego elementu,
 - **min** – przechowująca wartość tego elementu,
 - **n** – zapamiętująca numer aktualnie przeglądane elementu.

W momencie utworzenia zmiennych na scenie pojawią się plakietki z ich wartościami. Nie odhaczaj pól wyboru – dzięki temu łatwo podejrzysz wartości zmiennych w trakcie realizacji projektu.

- Ustaw wartości zmiennych:

- **nrMin = nrStart**,
- **min = element** o numerze **nrStart**,
- **n = nrStart** (można by było również ustawić **n** na element **nrStart + 1**).

- Wstaw pętlę **powtarzaj aż** – działania w pętli powinny trwać, aż skończy się lista (oznacza to, że wartość **n** powinna być większa niż długość listy **Dane**) – a w niej:

- sprawdzenie, czy n -ty element jest mniejszy od **min**,
– jeśli tak, to zamiana **nrMin** na **n** i **min** na wartość tego elementu,
- zmiana **n** o 1.

Rys. 4. Znajdowanie numeru najmniejszego elementu

Po wykonaniu bloku **numerMin** zmienne **nrMin** i **min** powinny zawierać numer i wartość kolejnej najmniejszej liczby w ciągu.

- Pozostało utworzenie i zdefiniowanie bloku **sortujWybierając**, który będzie odpowiadał za sortowanie listy w miejscu.
- Utwórz zmienną **i** przechowującą informację, od którego elementu dane nie zostały jeszcze posortowane.
- Ustaw wartość początkową zmiennej **i** na 1, a potem powtarzaj, aż do osiągnięcia końca listy:
 - znajdowanie numeru najmniejszej liczby w ciągu, począwszy od pozycji zmiennej **i**;
 - przestawianie elementów: tego, który jest najmniejszą liczbą oraz tego, na którym jest wskaźnik (czyli **i**-tego);
 - czekanie 2 s, żeby można było zauważyć zamianę elementów;
 - przesunięcie wskaźnika na następny element, czyli zwiększenie zmiennej **i** o 1.
- Przetestuj skrypt – sprawdź, czy uruchomienie bloku **sortujWybierając** spowoduje posortowanie ciągu.
- Dodaj skrypt uruchamiający sortowanie po naciśnięciu klawisza spacji. Możesz w nim umieścić również komunikat o posortowaniu listy **Dane**.
- Uzupełnij skrypt o komunikat o ukończeniu sortowania – wykorzystaj zamianę tekstu na mowę.

The image shows a Scratch script for selection sort. The script starts with a 'definiuj sortujWybierając' block. It then sets a loop counter 'i' to 1. A 'powtarzaj aż' loop runs while 'i' is less than or equal to the length of the 'Dane' list. Inside the loop, it finds the minimum element (numerMin) starting from index 'i', swaps it with the element at index 'i', waits 2 seconds, and increments 'i' by 1. After the loop, it says 'Dane posortowane'. A key press event (space) triggers the 'sortujWybierając' block. To the right, a table shows the 'Dane' list being sorted from [2, 5, 10, 12, 15, 17, 19, 22, 30, 38, 38, 43] to [2, 5, 10, 12, 15, 17, 19, 22, 30, 38, 38, 43]. The 'nrMin' is 12 and 'min' is 43.

Dane	
1	2
2	5
3	10
4	12
5	15
6	17
7	19
8	22
9	30
10	38
11	38
12	43

+ długość 12 =

Rys. 5. Blok wykonujący sortowanie przez wybieranie

INNE METODY SORTOWANIA

W algorytmie sortowania przez wybieranie są dwie pętle: jedna przeszukuje ciąg, żeby znaleźć najmniejszy element (blok **numerMin**), druga przestawia kolejne elementy (blok **sortujWybierając**). W ten sposób algorytm wykonuje liczbę operacji rzędu n^2 , gdzie n jest liczbą elementów porządkowanego ciągu.

Czy istnieje oszczędniejszy, a więc szybszy sposób porządkowania? Tak, jest on nazywany QuickSort (czytaj: kliksort; szybkie sortowanie). Jego odmianą jest **sortowanie przez scalanie**, nazwane tak ze względu na zastosowany sposób działania. Algorytm ten wykorzystuje **rekurencję** oraz starą rzymską maksymę *divide et impera* – **dziel i rządź** (łatwiej jest pokonać wielu słabych przeciwników niż jednego silnego). Zbiór jednoelementowy jest uporządkowany. Jeśli więc będziesz dzielić zbiór, aż dojdiesz do pojedynczych elementów, to wystarczy je odpowiednio scalić, aby uzyskać zbiór uporządkowany.

ZADANIA

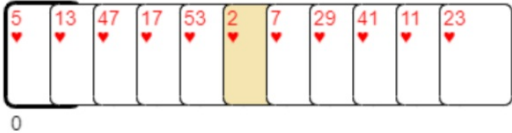
1. Odwróć kierunek sortowania w projekcie – uporządkuj liczby od największej do najmniejszej.
2. Sprawdź, czy sortowanie przez wybieranie działa dla liczb ułamkowych. Musisz odpowiednio zmienić sposób tworzenia ciągu liczb w skrypcie zielonej flagi.
3. Wejdź na stronę Akademii Khana. W przedmiocie **Informatyka** w dziale **Informatyka** znajduje się kurs **Algorytmy**,

a w nim rozdział poświęcony sortowaniu przez wybieranie.

Informatyka > Informatyka > Algoritmy > Sortowanie przez wybieranie

- Sortowanie
- Wyzwanie: zamiana wartości zmiennych
- Sortowanie przez wybieranie - pseudokod
- Wyzwanie: znajdź najmniejszy element w podtablicy
- Wyzwanie: realizacja sortowania przez wybieranie
- Analiza sortowania przez wybieranie
- Projekt: wizualizacja sortowania przez wybieranie

Za chwilę będziemy realizować różne algorytmy sortowania. Ale, na rozgrzewkę, poniżej znajduje się problem do rozwiązania związany z sortowaniem. Możesz zamieniać każdą parę kart klikając na jedną z nich, następnie na kolejną. Zamieniaj karty do momentu kiedy karty będą posortowane w taki sposób, że po lewej stronie będzie znajdowała się najmniejsza karta.



5	13	47	17	53	2	7	29	41	11	23
---	----	----	----	----	---	---	----	----	----	----

0

Start over

Jakiej strategii użyłes aby posortować te karty? Czy zmieniła się ona w trakcie sortowania?

Znajdź symulację sortowania widoczną na zrzucie i uporządkuj karty za pomocą algorytmu sortowania przez wybieranie.