3.6 Szybkie porządki

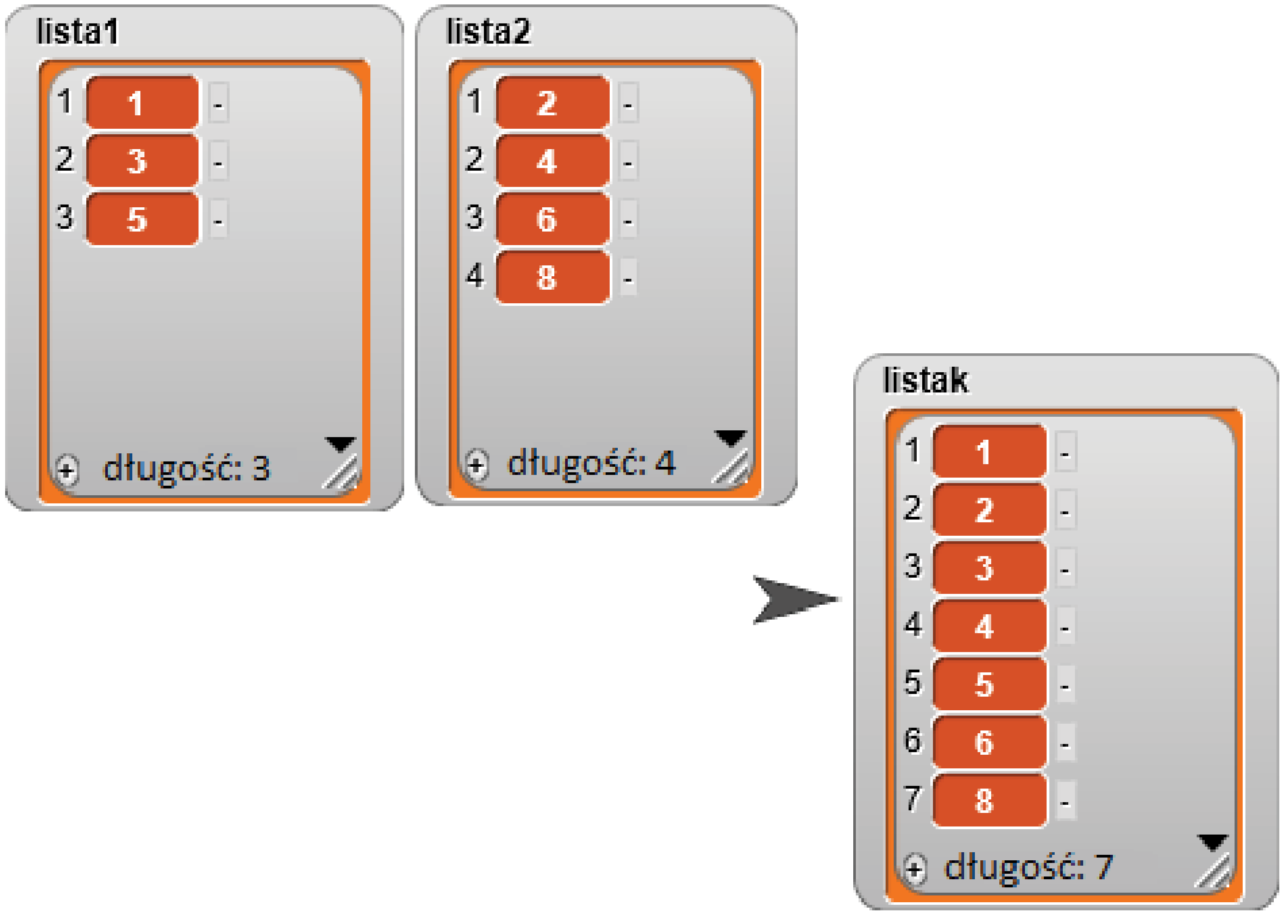
Materiał wyłącznie z podręcznika
i stron sugerowanych przez współautora – Witolda Kranasa

Sortowanie przez scalanie

- Klasycznym zagadnieniem, które wymaga szybkich obliczeń, jest sortowanie. Jeden z szybkich algorytmów sortowania nosi właśnie nazwę **szybkie sortowanie** (ang. *Quick-Sort*; czytaj: kłiksort) albo **sortowanie przez scalanie**. Algorytm ten wykorzystuje starą rzymską maksymę *divide et impera* – **dziel i rządź** (łatwiej jest pokonać wielu słabych przeciwników niż jednego silnego). Zbiór jednoelementowy jest uporządkowany. Jeśli więc podzielimy zbiór na pojedyncze elementy, to wystarczy je odpowiednio scalić, aby uzyskać zbiór uporządkowany. To scalenie nie jest trudne i czasochłonne.

Sposób scalania dwóch zbiorów uporządkowanych (załóżmy, że są one uporządkowane rosnąco – czyli od najmniejszego do największego elementu) można opisać następująco:

- Porównaj pierwsze elementy zbiorów.
- Mniejszy element wstaw do nowego zbioru (w którym będą przechowywane elementy uporządkowane) i usuń go ze starego zbioru.
- Powtórz te czynności, aż oba stare zbiory będą puste (nowy zbiór będzie wtedy zapełniony uporządkowanymi elementami).



https://snap.berkeley.edu/snapsource/snap.html#present:Username=witek&ProjectName=scalanie

- Przygotowane już skrypty tworzą dwie listy, z których kolejno wyławiane są najmniejsze elementy po porównaniu tych obu list, a następnie PRZENOSZONE czyli tam znikają, a pojawiają się w nowej liście – po prawej stronie….
- Tak się składa, że jedna sklada się z nieparzystych, a druga – z parzystych liczb…. A powstaje lista złożona z kolejnych liczb…

Najważniejsze standardowe algorytmy sortowania:

- http://wazniak.mimuw.edu.pl/index.php?title=Algorytmy_i_struktury_danych/Sortowanie_-_Bubble Sort%2C_SelectionSort%2C_InsertionSort

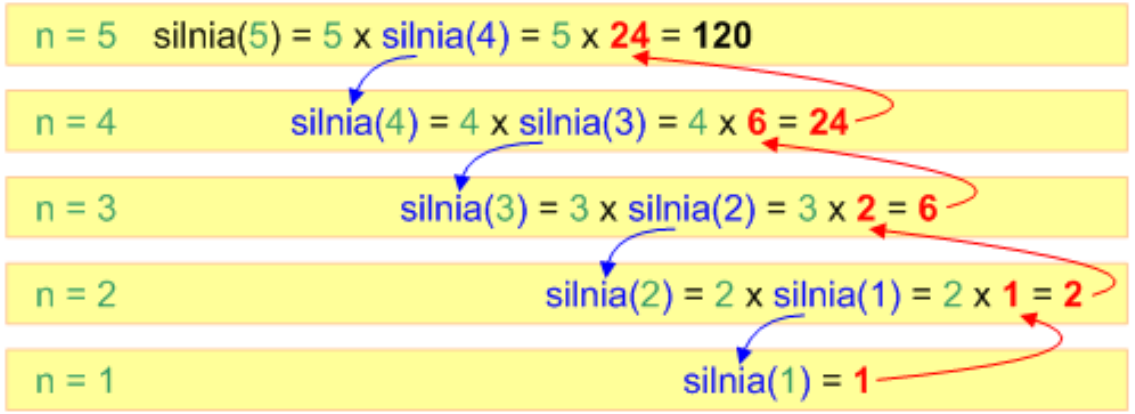
- W podanym w podręczniku adresie „wstawiła” się spacja – przez to link nie działał

- http://wazniak.mimuw.edu.pl/index.php?title=Algorytmy_i_struktury_danych/Sortowanie_-_BubbleSort%2C_SelectionSort%2C_InsertionSort

Treści są zdecydowanie dla tych, którzy programowaniem JUŻ się zajmują, a nie zaczynają….. Idźmy do drugiego linku.

https://eduinf.waw.pl/inf/alg/003_sort/0013.php

- Obejrzyjmy:
- Jak za pomocą rekurencji obliczana jest silnia
- Jak na grafice pokazane jest dzielenie na małe zbiory (dziel), porządkowanie i łączenie (i rządź)



Sortowany zbiór								Opis wykonywanych operacji
d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]	d[8]	
6	5	4	1	3	7	9	2	Zbiór wyjściowy.
6	5	4	1	3	7	9	2	Pierwszy podział.
6	5	4	1	3	7	9	2	Drugi podział
6	5	4	1	3	7	9	2	Trzeci podział.
5	6	1	4	3	7	2	9	Pierwsze scalanie.
1	4	5	6	2	3	7	9	Drugie scalanie.
1	2	3	4	5	6	7	9	Trzecie scalanie. Koniec.

Analiza sortowania przez scalanie

Scalane zbiory	Zbiór tymczasowy	Opis wykonywanych działań
[1] 3 6 7 9 2 3 4 6 8		Porównujemy ze sobą najmniejsze elementy scalanych zbiorów. Ponieważ zbiory te są już uporządkowane, to najmniejszymi elementami będą zawsze ich pierwsze elementy.
3 6 7 9 2 3 4 6 8	[1]	W zbiorze tymczasowym umieszczamy mniejszy element, w tym przypadku będzie to liczba 1. Jednocześnie element ten zostaje usunięty z pierwszego zbioru
3 6 7 9 [2] 3 4 6 8	1	Porównujemy kolejne dwa elementy i mniejszy umieszczamy w zbiorze tymczasowym.
[3] 6 7 9 3 4 6 8	1[2]	Następne porównanie i w zbiorze tymczasowym umieszczamy liczbę 3. Ponieważ są to elementy równe, to nie ma znaczenia, z którego zbioru weźmiemy element 3.
6 7 9 [3] 4 6 8	1 2[3]	Teraz do zbioru tymczasowego trafi drugie 3.
6 7 9 [4] 6 8	1 2 3[3]	W zbiorze tymczasowym umieszczamy mniejszy z porównywanych elementów, czyli liczbę 4.
[6] 7 9 6 8	1 2 3 3[4]	Porównywane elementy są równe, zatem w zbiorze tymczasowym umieszczamy dowolny z nich.
7 9 [6] 8	1 2 3 3 4[6]	Teraz drugą liczbę 6.
[7] 9 8	1 2 3 3 4 6[6]	W zbiorze tymczasowym umieszczamy liczbę 7
9 [8]	1 2 3 3 4 6 6[7]	Teraz 8
[9]	1 2 3 3 4 6 6 7[8]	Drugi zbiór jest pusty. Od tego momentu już nie porównujemy, lecz wprowadzamy do zbioru tymczasowego wszystkie pozostałe elementy pierwszego zbioru, w tym przypadku będzie to liczba 9.
	1 2 3 3 4 6 6 7 8[9]	Koniec scalania. Zbiór tymczasowy zawiera wszystkie elementy scalanych zbiorów i jest uporządkowany. Możemy w dalszej kolejności przepisać jego zawartość do zbioru docelowego.

Przejrzyj podręcznik samodzielnie – do końca

Zadania

1. Sprawdź, czy sortowanie zastosowane do listy liczb w projekcie w programie SNAP! sprawi, że te liczby ustawią się rosnąco czy malejąco. Jak można odwrócić kierunek sortowania? Zrób to w projekcie. – nie musisz
2. Znajdź w internecie informacje o różnych sposobach (algorytmach) sortowania i zapisz jeden z nich. **Zrobiliśmy to na poprzedniej lekcji – bez zapisywania, bo nie prowadzimy zeszytu.**
3. Znajdź w internecie informacje na temat sortowania bąbelkowego, wyszukaj wizualizacje tego algorytmu i opisz, jak wygląda to sortowanie. **Odwiedziliśmy różne strony z symulacjami i oglądaliśmy tańczących studentów.**

- **TERAZ zrób quiz.**
  Link na stronie sp373.srv.pl/materialy