Python: wyszukiwanie binarne, algorytmy sortujące, złożoność obliczeniowa.

Wyszukiwanie binarne

Algorytm służy do wyszukiwania elementu w tablicy uporządkowanej (posortowanej). Opiera się na metodzie „dziel i zwyciężaj”.

Algorytm:

1. Wyznaczymy element środkowy zbioru
2. Sprawdzimy, czy jest on poszukiwanym elementem. Jeśli:
   - tak - to element został znaleziony i możemy zakończyć poszukiwania
   - nie - poszukiwany element jest mniejszy albo od elementu środkowego.
3. Powyższe punkty wykonujemy dla tej części tablicy w której znajduje się poszukiwany element.

Zadanie 1:

1. Napisać funkcję realizująca wyszukiwanie binarne.
2. Napisać funkcję wyszukująca elementy w sposób liniowy
3. Porównać czas działania obu algorytmów.

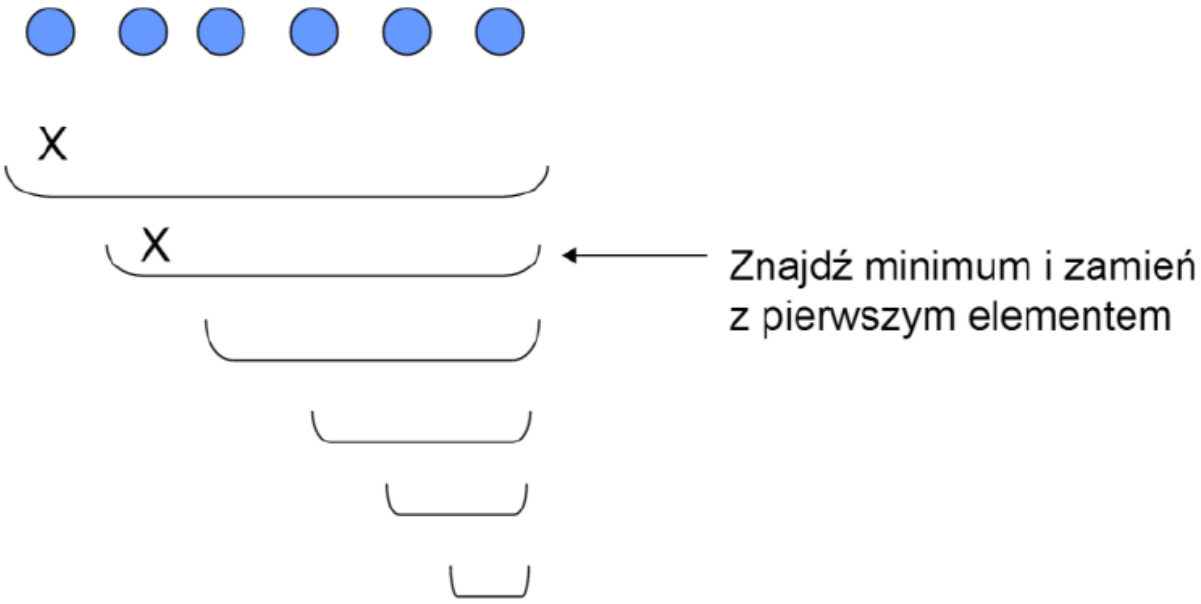
Sortowanie bąbelkowe

Algorytm polega na porównywaniu kolejnych dwóch elementów i zamianie ich kolejności na pożądaną. Porównywanie odbywa się w dwóch zagnieżdżonych pętlach.

Sortowanie przez wybieranie

Algorytm polega na wyszukaniu elementu mającego się znaleźć na żądanej pozycji i zamianie miejscami z tym, który jest tam obecnie:

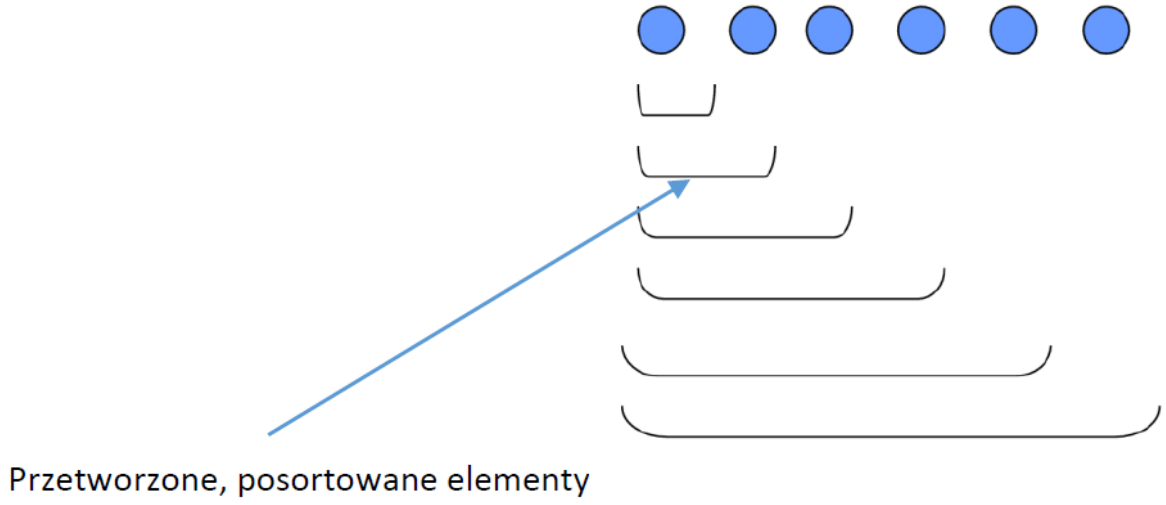
Szukamy najmniejszego elementu na liście. Następnie zamieniamy go z pierwszym elementem listy. Potem ponownie szukamy najmniejszego elementu, tym razem zaczynając od drugiego. Zamieniamy z nim znaleziony rezultat. Powtarzamy ten schemat dopóki nie dotrzemy do ostatniego elementu na liście. Zawsze szukamy najmniejszego elementu z zakresu [i, N-1] i zamieniamy go z i-tym elementem. Gdy i = N - 1 mamy już pewność, że wszystkie elementy za nami zostały posortowane, a przed nami nie ma żadnych elementów – lista jest posortowana.



Sortowanie przez wstawianie

Algorytm polega na wstawianiu kolejnych elementów w odpowiednie miejsca posortowanej już części tablicy:

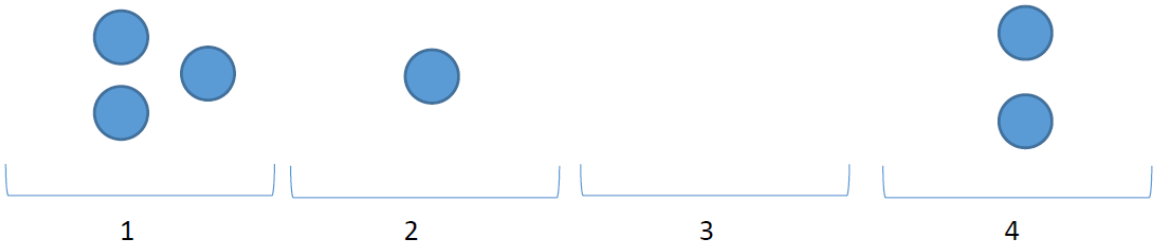
Przy sortowaniu przez wstawianie nasza lista podzielona jest na dwie części. Elementy od 0 do i-1 są już posortowane, a elementy od i do N-1 wymagają posortowania. Na początku i=1 więc za nami jest tylko jeden („posortowany”) element. W każdym kroku usuwamy i-ty element i porównujemy go po kolei z każdym z pierwszych i-1 elementów, dopóki są one mniejsze lub równe od naszego elementu. Wstawiamy nasz element przed pierwszy większy od niego element, który napotkamy. Lista jest posortowana, gdy wstawimy w odpowiednie miejsce ostatni element listy, tj. i = N - 1.



Sortowanie przez zliczanie

Algorytm polega na zliczaniu, ile razy dana liczna występuję w ciągu, następnie tworzy nowy ciąg korzystając z zebranych danych. Algorytm zakłada, że klucze elementów należą do skończonego zbioru. Opis:

Tworzymy listę *counter* o długości K (maksymalna wartość elementu) i wypełniamy ją wartością 0. Wykorzystujemy *counter* do zliczenia wystąpień każdego elementu z listy liczb: gdy i-ty element listy ma wartość x, podnosimy wartość counter[x] o 1. Gdy przejdziemy w ten sposób przez wszystkie N liczb, tworzymy nową listę, na której będziemy zapisywać posortowane liczby. Następnie przechodzimy po liście *counter* od 0 do K-1 i dodajemy do wynikowej listy daną pozycję tyle razy, ile wystąpiła według statystyki. Na przykład, jeśli w i-tym elemencie listy *counter* mamy wartość 2 to dodajemy na koniec wynikowej listy wartość i dwa razy.



Zadanie 2:

1. Zaimplementować powyższe algorytmy sortujące.
2. Porównać czas działania.