Сортирање секвенце

Сортирање значи међусобно упоређивање свих елемената из листе и међусобна замена њихове позиције у листи према вредностима, у зависности од тога да ли се елементи сортирају од најмање према највишој вредности или обрнуто.

Из тог разлога, алгоритми за сортирање имају више пролазака кроз листу него алгоритми за претрагу.

Постоји више алгоритама за сортирање секвенци: метода избора, метода замене суседа, метода уметања, метода поделе.

Сортирање методом избора

0262 Метода избора у сортирању

```
def main():
    lista = unos_liste()
    prikaz_liste(lista)
    lista = sortiranje(lista)
    prikaz_liste(lista)

def unos_liste():
    print("Uneti u listu samo cele brojeve.")
    A = []
    jos = "da"
    while jos == "da":
        x = int(input("Uneti zeljeni broj u listu:"))
        A.append(x)
        print("Da li treba uneti jos jedan element u listu? ")
        jos = input("Uneti da ili bilo sta drugo za ne: ")
    return A

def prikaz_liste(A):
    print(A)

def sortiranje(A):
    n = len(A)
    for i in range (0, n - 1):
        for j in range (i + 1, n):
            if (A[i] > A[j]):
                b = A[i]
                A[i] = A[j]
                A[j] = b

    return A

main()
```

Метода избора се заснива на упоређивању вредности елемената из две петље.
Спољна петља (од 1. до претпоследњег елемента) упоређује сваки елемент, осим последњег, са свим осталим елементима унутрашње петље (од 2. до последњег елемента).
На тај начин се неће упоређивати елементи на истим позицијама.
Ако је утврђено да је елемент спољне петље већи од елемента унутрашње петље, они замењују места.
Тако се обезбеђује да индекс мањег елемента буде мањи од индекса већег елемента а то је сортирање од мањег елемента ка већим.

Сортирање методом замене суседа

0263 Метода замене суседа у сортирању

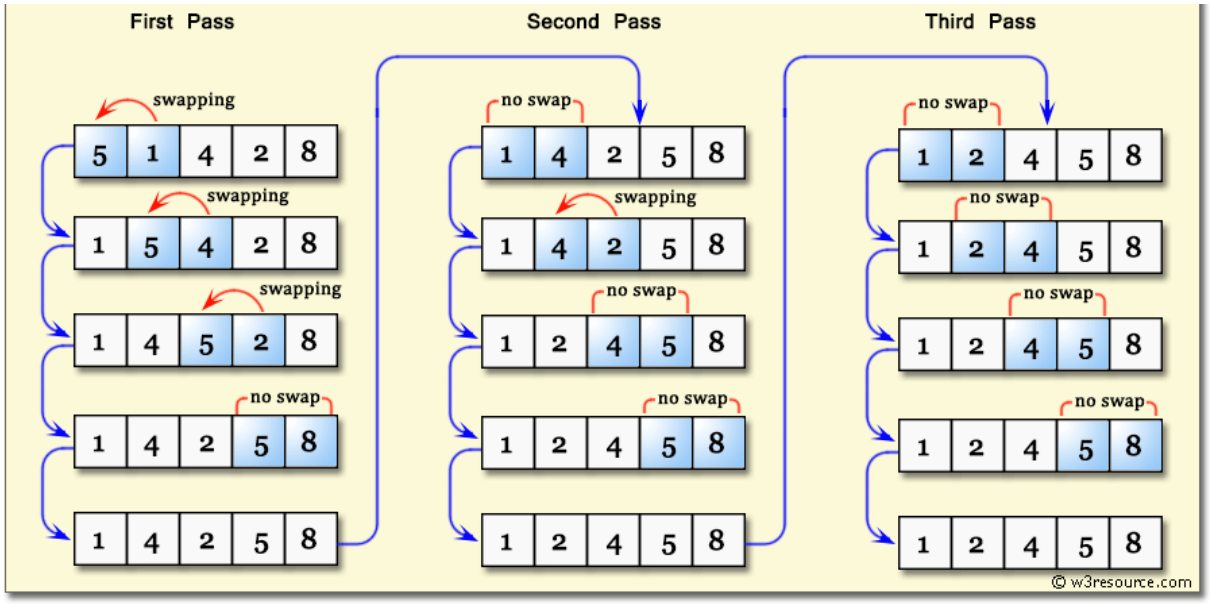
```
def sortiranje(A):
    n = len(A)
    for i in range(0, n):
        for j in range(1, n - i):
            if A[j - 1] > A[j]:
                b = A[j]
                A[j] = A[j - 1]
                A[j - 1] = b

    return A
```

У овој методи се спољна петља користи за обезбеђивање n итерација над секвенцом.
Унутрашња петља се користи за избор елемената који се упоређују.
Увек се упоређују два суседна елемента по позицији у секвенци.
Ако је елемент са мањом позицијом већи од елемента са већом позицијом, врши се замена позиција.
Упоређивање се врши са смањеним бројем елемената како се приближава крају секвенце.

Bubble-sort

Ова метода се заснива на понављању корака сортирања елемената у листи помоћу упоређивања сваког пара суседних вредности и њиховој замени места ако им је редослед погрешан.
Пролазак кроз листу се понавља све док се више замена неће извршити, што значи да је листа сортирана.
Сам алгоритам је доста спор и непрактичан за решавање реалних проблема чак и ако се упоређује са методом избора.
Користи се код сортираних листа којима се могу придодавати појединачни елементи које је потребно сортирати у постојећу листу.



0264 Метода сортирања bubble-sort

```
def bubble_sort(A):
    for x in range(len(A) - 1, 0, -1):
        for i in range(x):
            if A[i] > A[i + 1]:
                y = A[i]
                A[i] = A[i + 1]
                A[i + 1] = y

A = [14,46,43,27,57,41,45,21,70]
bubble_sort(A)
print(A)
```

Види се у коду да постоје две петље: спољна са x бројачем и унутрашња са i бројачем.
Десна граница опсега унутрашње петље зависи од тренутне вредности бројача x и увек је мања од индекса последњег елемента у листи A.
У првом циклусу (x = 0, i = 0, 7) се уређују позиције за парове елемената који се по први пут упоређују и ефекат је да се елементи са већим вредностима почињу померати удесно у листи.
То значи да ће највећи елемент у листи сигурно бити највећи у било којем пару упоређиваних елемената па ће сигурно бити и постављен на последње место у листи.
Из тог разлога нема потребе више, у следећим циклусима, упоређивати елемент на позицији која је у тренутном циклусу правилно попуњена, па се у сваком следећем циклусу опсег унутрашње петље смањује за један.