[2pkt.] Zadanie 1. Cyfra jednokrotna to taka, która występuje w danej liczbie dokładnie jeden raz. Cyfra wielokrotna to taka, która w liczbie występuje więcej niż jeden raz. Mówimy, że liczba naturalna A jest ładniejsza od liczby naturalnej B jeżeli w liczbie A występuje więcej cyfr jednokrotnych niż w B, a jeżeli cyfr jednokrotnych jest tyle samo to ładniejsza jest ta liczba, która posiada mniej cyfr wielokrotnych. Na przykład: liczba 123 jest ładniejsza od 455, liczba 1266 jest ładniejsza od 114577, a liczby 2344 i 67333 są jednakowo ładne.
Dana jest tablica T zawierająca liczby naturalne. Proszę zaimplementować funkcję:

`pretty_sort(T)`

która sortuje elementy tablicy T od najładniejszych do najmniej ładnych. Użyty algorytm powinien być możliwie jak najszybszy. Proszę w rozwiązaniu umieścić 1-2 zdaniowy opis algorytmu oraz proszę oszacować jego złożoność czasową.

QS

HS ✓

MS ✓

T = [123, 445, 28, 22, 4456]

3 0 1 1 2 0 0 1 2 1

T = [123, 28, 4456, 445, 22]

A = [(123, 3, 0), (445, 1, 1), ...]

O(m*log(n)) O(m)

[2pkt.] Zadanie 2.

Szablon rozwiązania: `zad2.py`

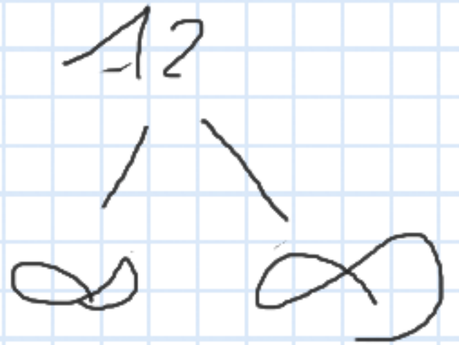
Węzły jednokierunkowej listy odsyłaczowej reprezentowane są w postaci:

```
class Node:
    def __init__(self):
        self.val  = None  # przechowywana liczba rzeczywista
        self.next = None  # odsyłacz do nastepnego elementu
```

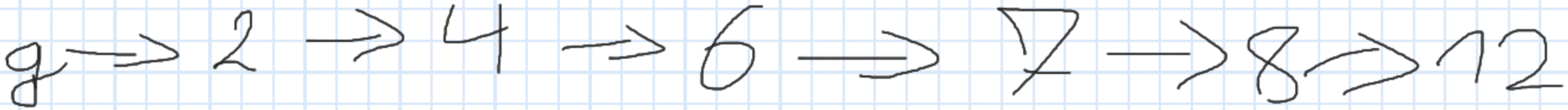
Niech p będzie wskaźnikiem na niepustą listę odsyłaczową zawierającą parami różne liczby rzeczywiste $a_1, a_2, \ldots, a_n$ (lista nie ma wartownika). Mówimy, że lista jest k-chaotyczna jeśli dla każdego elementu zachodzi, że po posortowaniu listy znalazłby się na pozycji różniącej się od bieżącej o najwyżej k. Tak więc 0-chaotyczna lista jest posortowana, przykładem 1-chaotycznej listy jest $1, 0, 3, 2, 4, 6, 5$, a $(n-1)$-chaotyczna lista długości n może zawierać liczby w dowolnej kolejności. Proszę zaimplementować funkcję `SortH(p,k)`, która sortuje k-chaotyczną listę wskazywaną przez p. Funkcja powinna zwrócić wskazanie na posortowaną listę. Algorytm powinien być jak najszybszy oraz używać jak najmniej pamięci (w sensie asymptotycznym, mierzonym względem długości n listy oraz parametru k). Proszę oszacować jego złożoność czasową dla $k = \Theta(1)$, $k = \Theta(\log n)$ oraz $k = \Theta(n)$.

$O(n \log n)$

$O(n \log k)$



$k = 3$



[2pkt.] Zadanie 3.

Szablon rozwiązania: `zad3.py`

Mamy daną N elementową tablicę T liczb rzeczywistych, w której liczby zostały wygenerowane z pewnego rozkładu losowego. Ten rozkład mamy zadany jako k przedziałów $[a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k]$ takich, że i-ty przedział jest wybierany z prawdopodobieństwem c_i, a liczba z przedziału jest wybierana zgodnie z rozkładem jednostajnym. Przedziały mogą na siebie nachodzić, liczby a_i, b_i są liczbami naturalnymi ze zbioru $\{1, \ldots, N\}$. Proszę zaimplementować funkcję `SortTab(T,P)` sortująca podaną tablicę. Pierwszy argument to tablica do posortowania a drugi to opis przedziałów w postaci:

`P = [`(a_1, b_1, c_1)`, `(a_2, b_2, c_2)`, ..., `(a_k, b_k, c_k)`]`.

Na przykład dla wejścia:

```
P = [(1,5, 0.75) , (4,8, 0.25)]
T = [6.1, 1.2, 1.5, 3.5, 4.5, 2.5, 3.9, 7.8]
```

po wywołaniu `SortTab(T,P)` tablica `T` powinna być postaci:

```
T = [1.2, 1.5, 2.5, 3.5, 3.9, 4.5, 6.1, 7.8]
```

Algorytm powinien być możliwie jak najszybszy. Proszę podać złożoność czasową i pamięciową zaproponowanego algorytmu.

BS ✓

[1, 2], [2, 3], ... [N-1, N]

Mówimy, że tablica `T` ma współczynnik nieuporządkowania równy k (jest k-Chaotyczna), jeśli spełnione są łącznie dwa warunki:

1. tablicę można posortować niemalejąco przenosząc każdy element `A[i]` o co najwyżej k pozycji (po posortowaniu znajduje się on na pozycji różniącej się od i co najwyżej o k),
2. tablicy nie da się posortować niemalejąco przenosząc każdy element o mniej niż k pozycji.

Proszę zaproponować i zaimplementować algorytm, który otrzymuje na wejściu tablicę liczb rzeczywistych `T` i zwraca jej współczynnik nieuporządkowania. Algorytm powinien być jak najszybszy oraz używać jak najmniej pamięci. Proszę uzasadnić jego poprawność i oszacować złożoność obliczeniową. Algorytm należy zaimplementować jako funkcję:

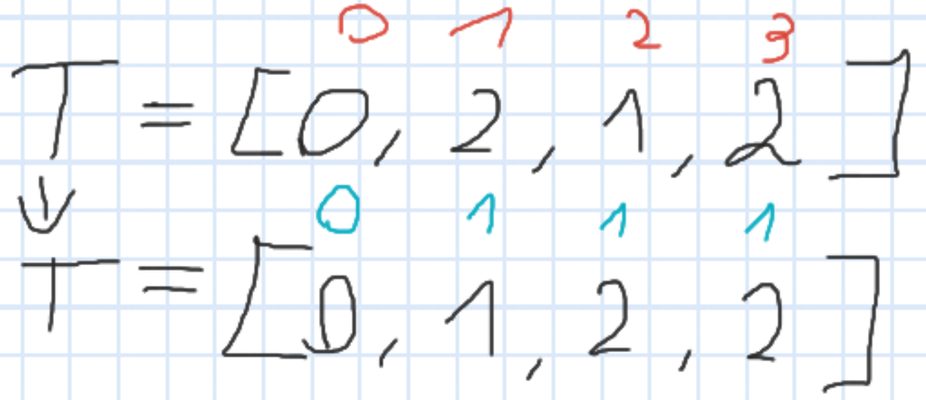
```
def chaos_index( T ):
    ...
```

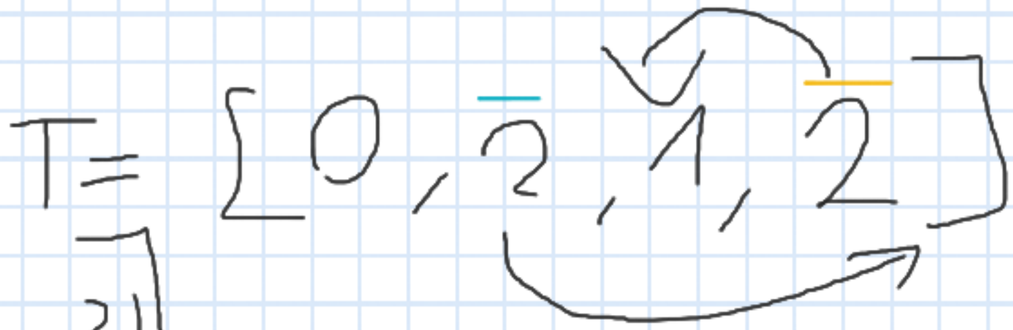
przyjmującą tablicę `T` i zwracającą liczbę całkowitą będącą wyznaczonym współczynnikiem nieuporządkowania.

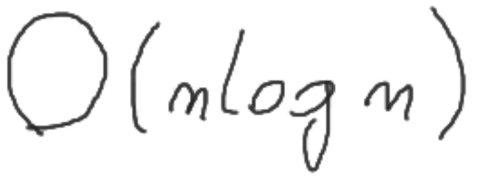
Przykład. Dla tablicy:

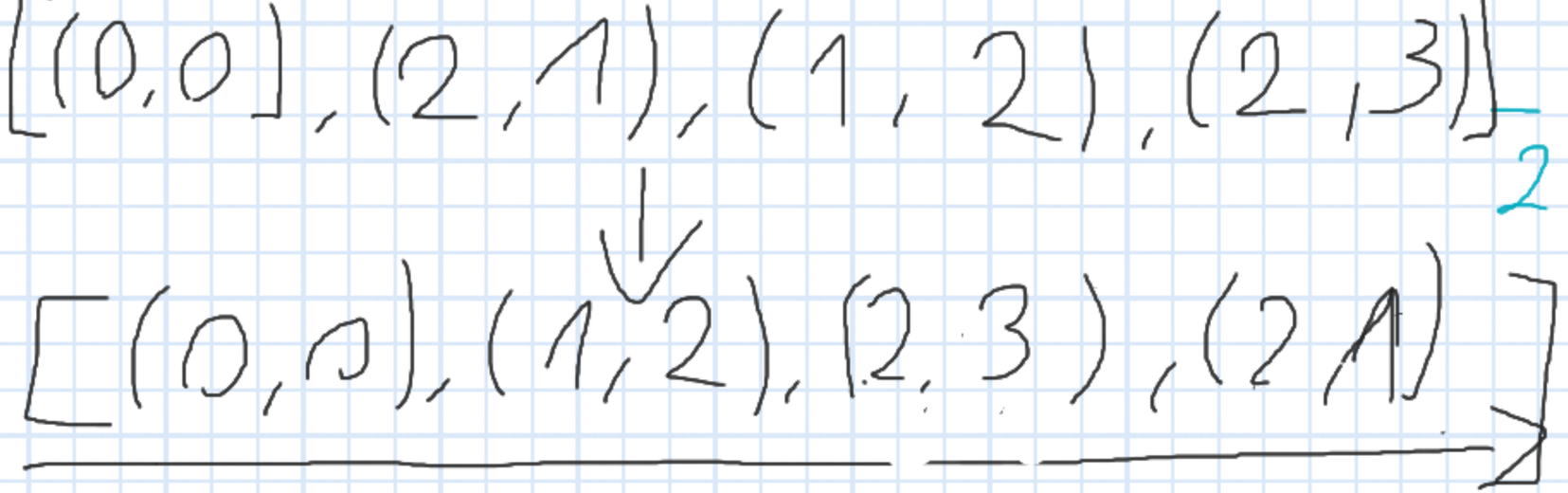
```
T = [0, 2, 1.1, 2]
```

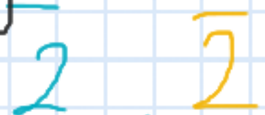
prawidłowym wynikiem jest $k = 1$.

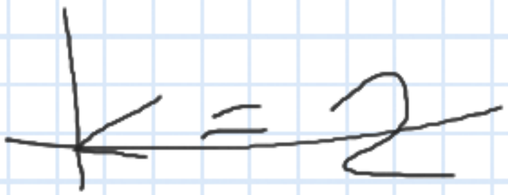








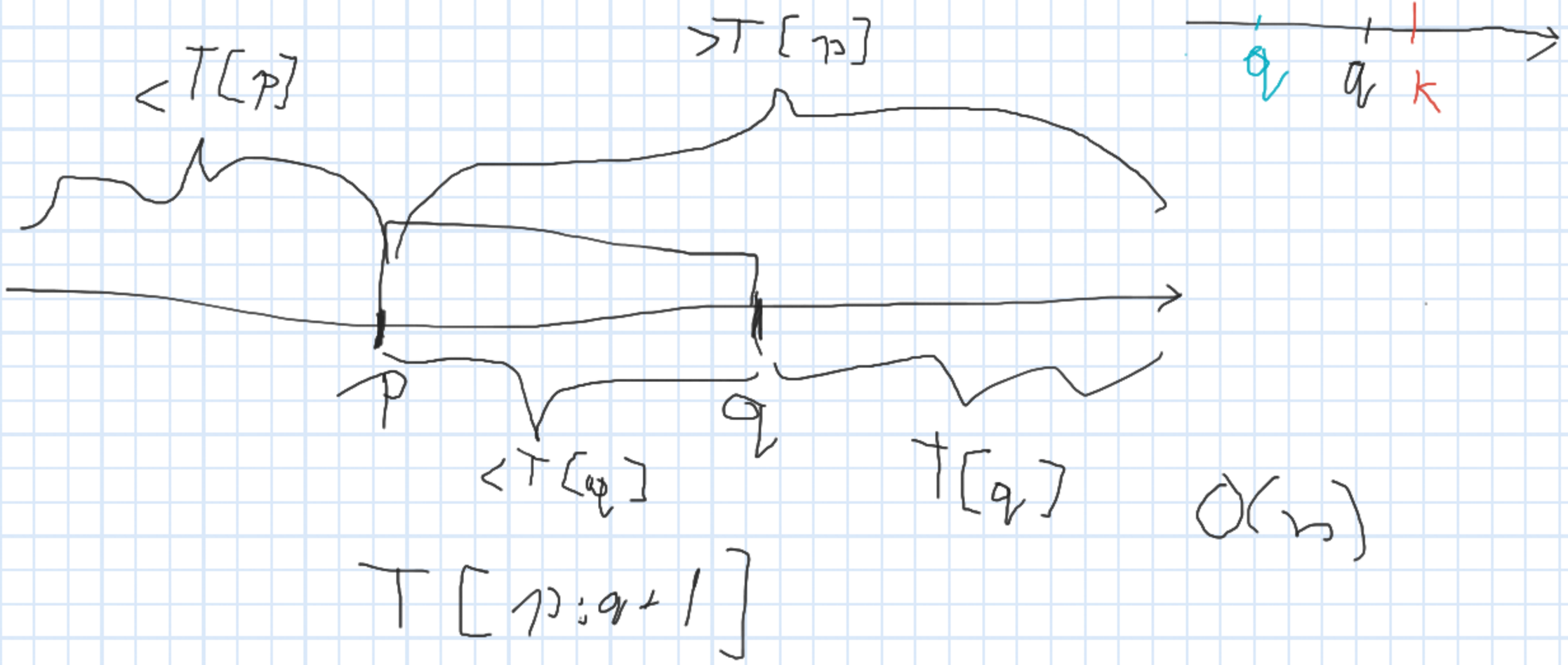




[2pkt.] Zadanie 2. Mamy n żołnierzy różnego wzrostu i nieuporządkowaną tablicę, w której podano wzrosty żołnierzy. Żołnierze zostaną ustawieni na placu w szeregu malejąco względem wzrostu. Proszę zaimplementować funkcję:

`section(T,p,q)`

która zwróci tablicę ze wzrostami żołnierzy na pozycjach od p do q włącznie. Użyty algorytm powinien być możliwie jak najszybszy. Proszę w rozwiązaniu umieścić 1-2 zdaniowy opis algorytmu oraz proszę oszacować jego złożoność czasową.



[2pkt.] Zadanie 3.

Szablon rozwiązania: `zad2.py`

Pewien eksperyment fizyczny daje w wyniku liczby rzeczywiste postaci a^x, gdzie a to pewna stała większa od 1 ($a > 1$) zaś x to liczby rzeczywiste rozłożone równomiernie na przedziale $[0, 1]$. Napisz funkcję `fast_sort`, która przyjmuje tablicę liczb z wynikami eksperymentu oraz stałą a i zwraca tablicę z wynikami eksperymentu posortowanymi rosnąco. Funkcja powinna działać możliwie jak najszybciej. Uzasadnij poprawność zaproponowanego rozwiązania i oszacuj jego złożoność obliczeniową. Nagłówek funkcji `fast_sort` powinien mieć postać:

```
def fast_sort(tab, a):
    ...
```

[10.5, 11.7, ...]

[(10.5, 3.2), ...]