Listy

Definicja Lista jest to struktura danych zawierająca pewne elementy ustawione w jakiejś kolejności.

Zatem mając element x możemy mówić
o elemencie natsępnym i poprzednim na liście

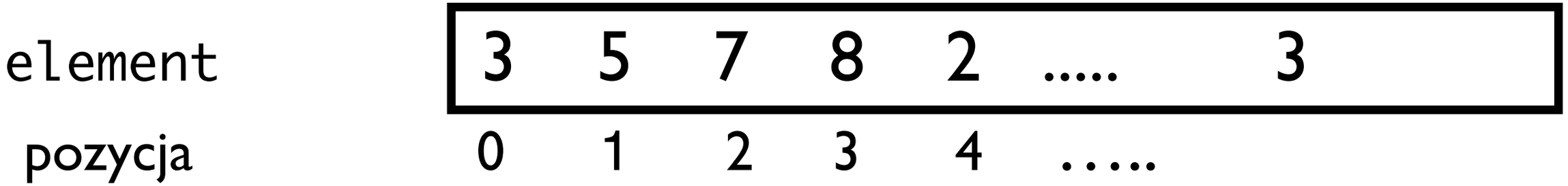
Typowe operacje na liście to: wstaw, usuń, szukaj

Implementacje listy:

- tablicowa
- dowiązaniowa

implementacja tablicowa list

elementy listy są pamiętane na kolejnych pozycjach w pamięci i są indeksowane kolejnymi liczbami 0, 1, 2, …



takie np. są listy w Pythonie

implementacja tablicowa list

Obserwacja czasowa złożoność pesymistyczna operacji wstaw, szukaj, usuń jest $\Theta(n)$, *gdzie n to* ilość elementów na liście (czyli w tablicy)

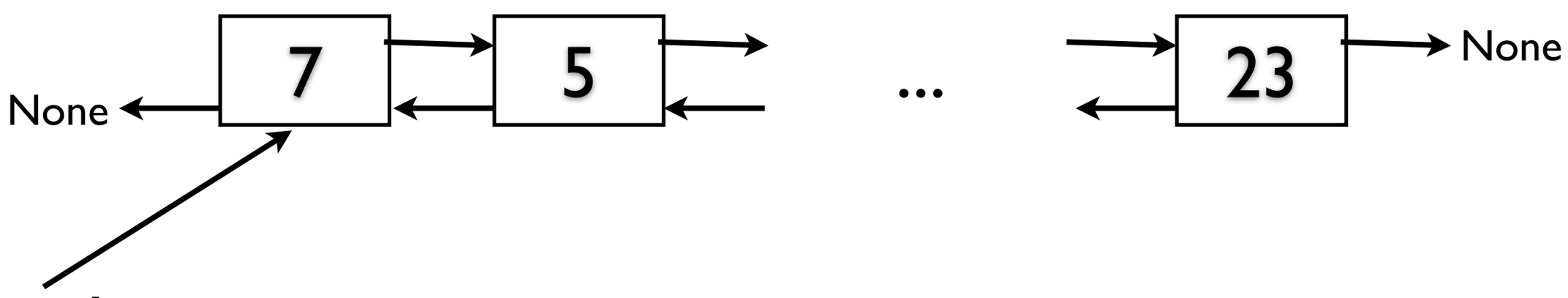
Uzasadnienie: wstawienie na początek, czy usunięcie elementu z początku listy pociąga za sobą przesuwanie elementów listy

Szukanie też może wymagać sprawdzenia wszystkich elementów zanim np. stwierdzimy, że szukanego elementu nie ma na liście

listy dowiązaniowe

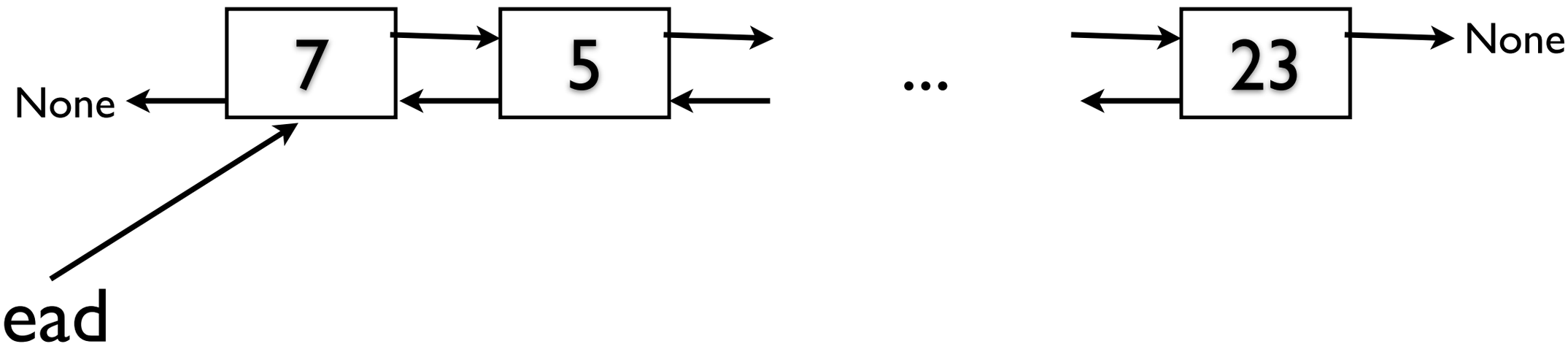
elementy listy to pewne struktury w zapisane w pamięci na niekoniecznie kolejnych miejscach, ale każdy element ma informację, jaki jest następny element i jaki poprzedni, czyli ma dowiązanie (wskaźnik) do następnego i poprzedniego, reprezentowane na poniższych rysunkach strzałkami

None to specjalna wartość oznaczająca brak dowiązania

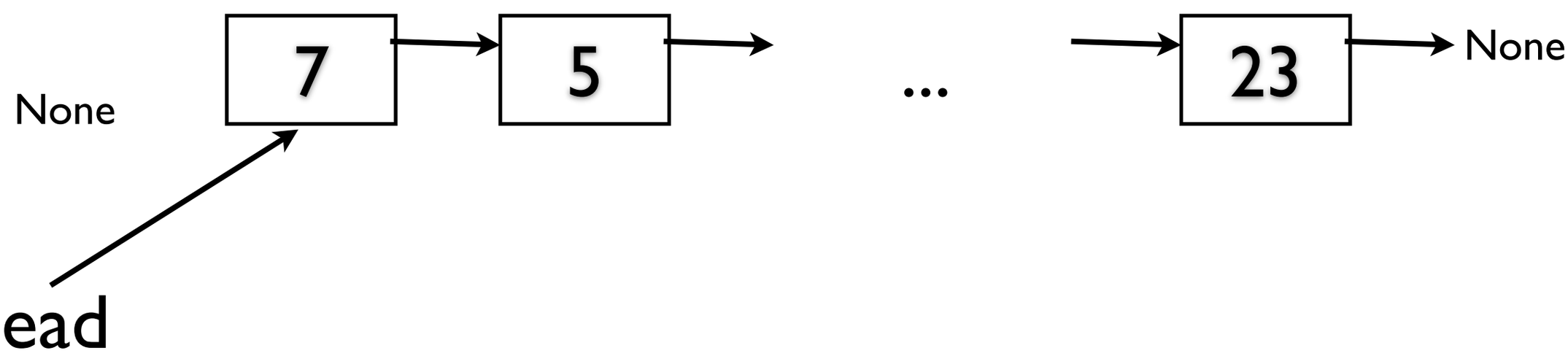


listy dowiązaniowe mogą być:

- dwukierunkowe

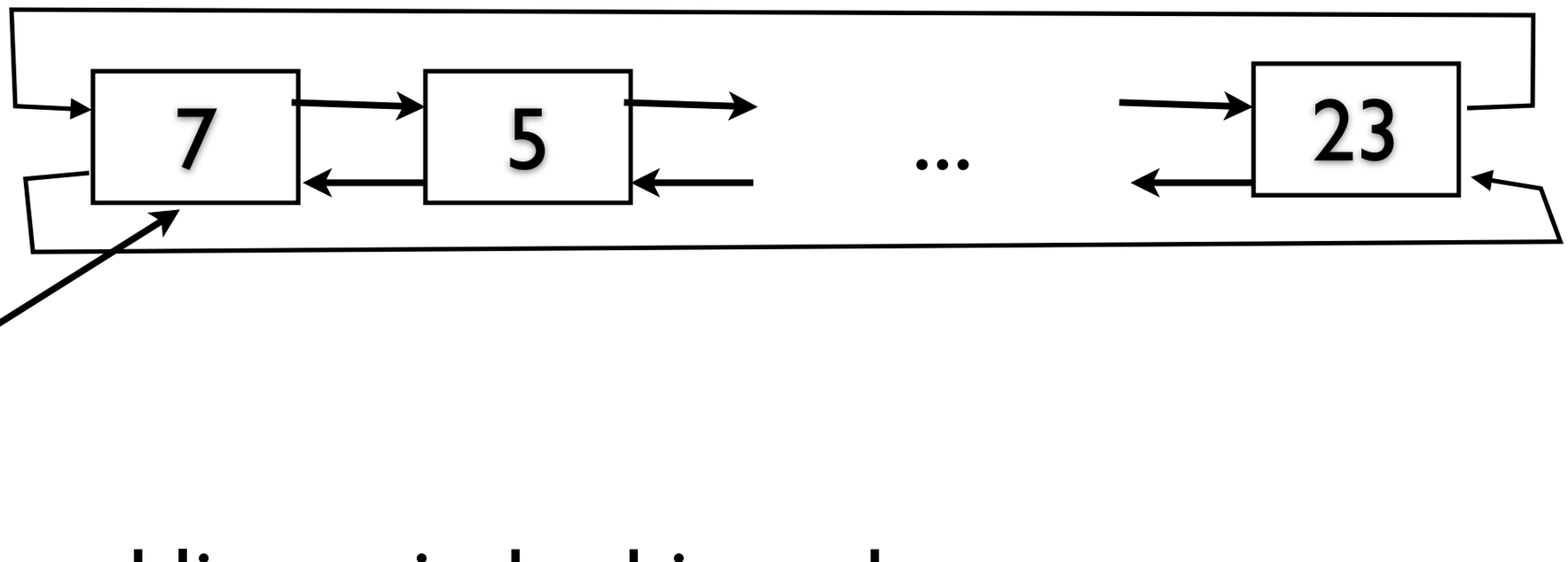


- jednokierunkowe

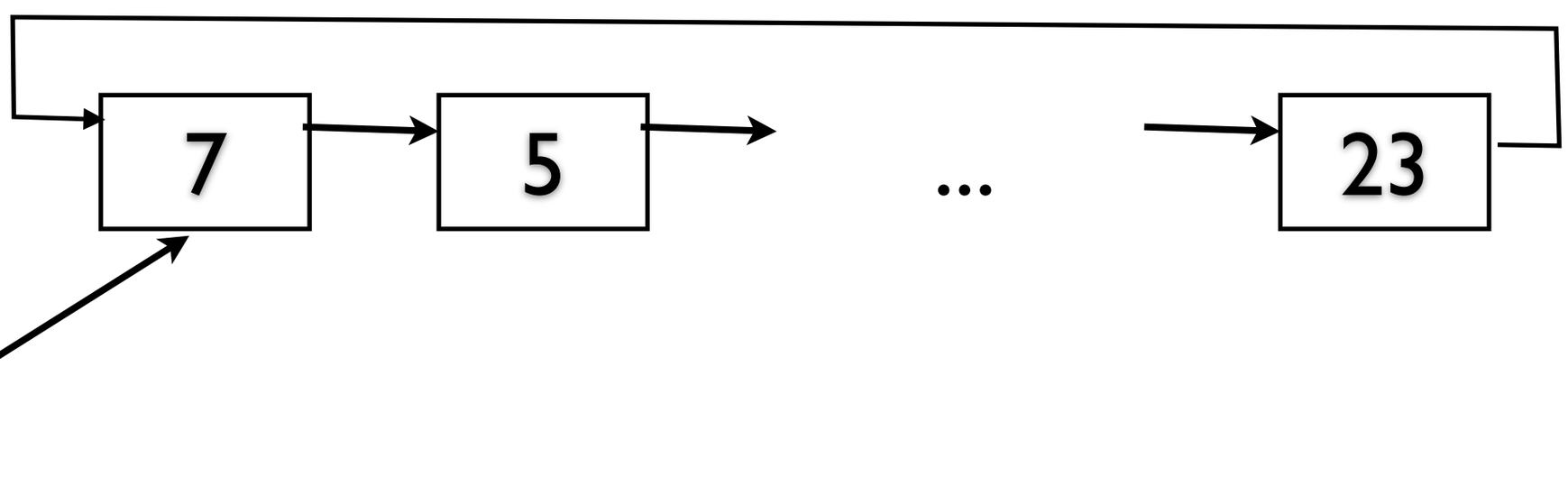


listy dowiązaniowe mogą być:

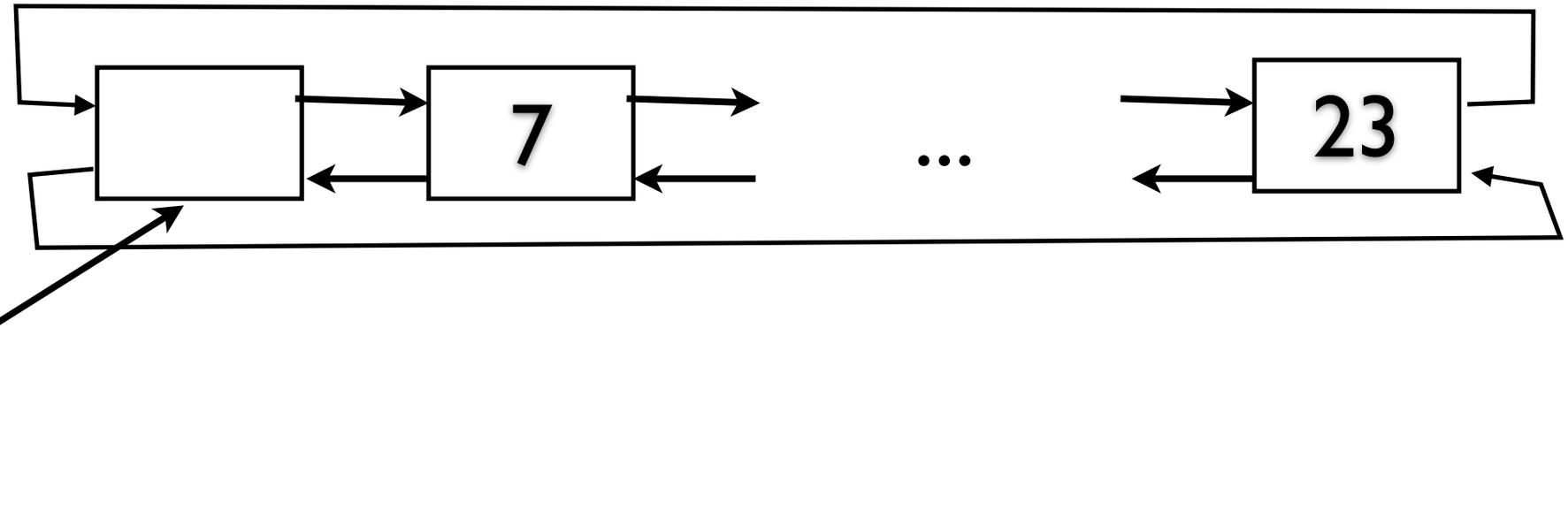
- cykliczne dwukierunkowe



- cykliczne jednokierunkowe



- i te same warianty z wartownikiem



none

none (przez małe n) oznacza tutaj dodatkowe miejsce na liście, w którym nie przechowujemy żadnej wartości ale służące tylko do tego, żeby lista nie była nigdy pusta

Notacja

None - pusty wskaźnik
l.head - początek listy *l*
x.next - następny element po elemncie *x*
x.prev - poprzedni element po elemencie *x*
x.key - wartość (klucz) identyfikujący węzeł *x*
l.none - wartownik listy *l* (jeżeli lista ma wartowników)

elementy listy dowiązaniowej często nazywamy węzłami listy; dokładniej, węzeł to zawartość elementu listy plus dowiązania

węzeł listy w Pythonie

```
class Node:
    def __init__(self,k):
        self.key = k
        self.next = None
        self.prev = None
```

jest to definicja klasy

Utworzenie obiektu tej klasy, czyli węzła:

```
x = Node(7)
```

instrukcja

```
x = Node(7)
```

wywołuje konstruktor klasy `Node`, czyli funkcję

```
__init__(self,k)
```

`self` oznacza tutaj konstruowny obiekt czyli `x`

lista w Pythonie

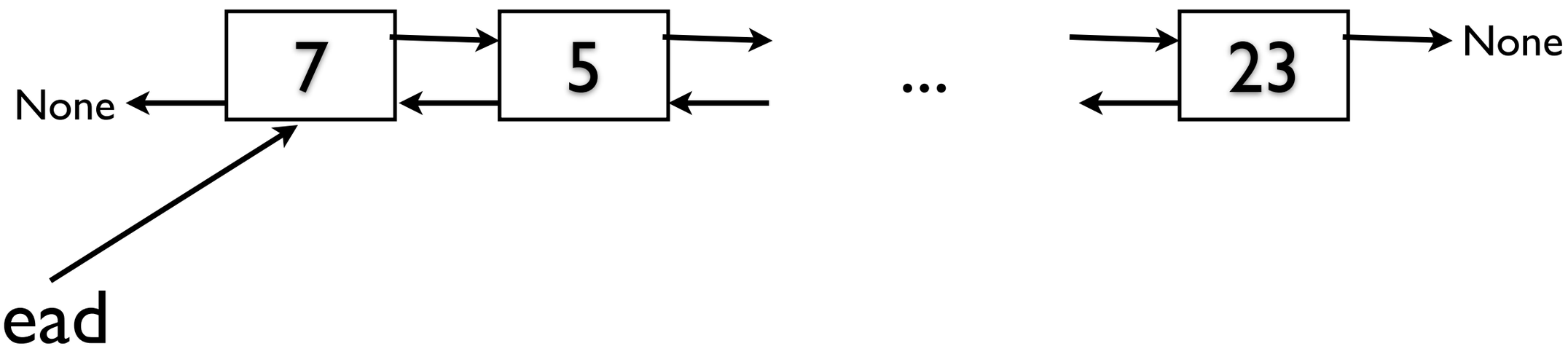
```
class LinkedList:
    def __init__(self):
        self.head = None

    def listInsert(self, x):
    ......................
    # i dalej definicje pozostałych
    # funkcji
```

jest to definicja kolejnej klasy

`head` jest pierwszym elementem listy

Utworzenie obiektu tej klasy: **`l = LinkedList()`**

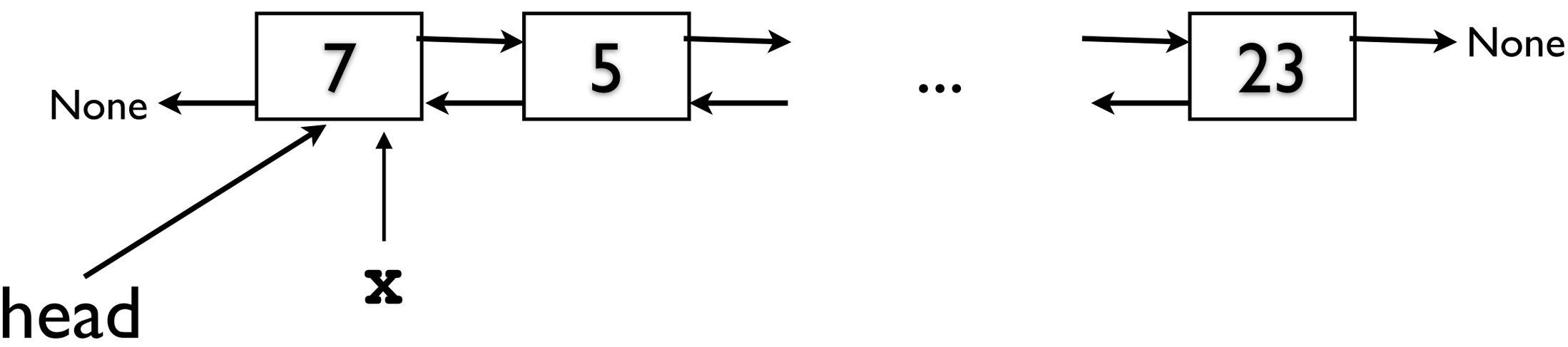
szukanie (poprzez przeglądanie listy od początku)



złożoność pesym. $\Theta(n)$, n - ilość elementów

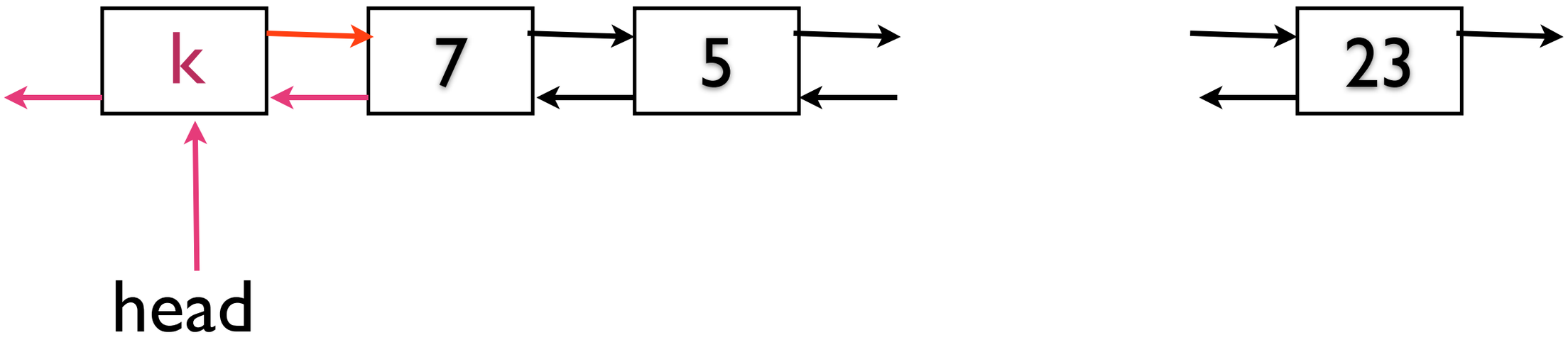
szukanie (poprzez przeglądanie listy od początku)

```
def listSearch(self,k):
    # szuka wezla zawierajacego klucz k
    # lista dwukierunkowa niecykliczna bez wartownika
        x = self.head
        while x!=None and x.key!=k:
            x = x.next
        return x    # None oznacza, ze szukanego klucza
                    # nie ma na liscie
```



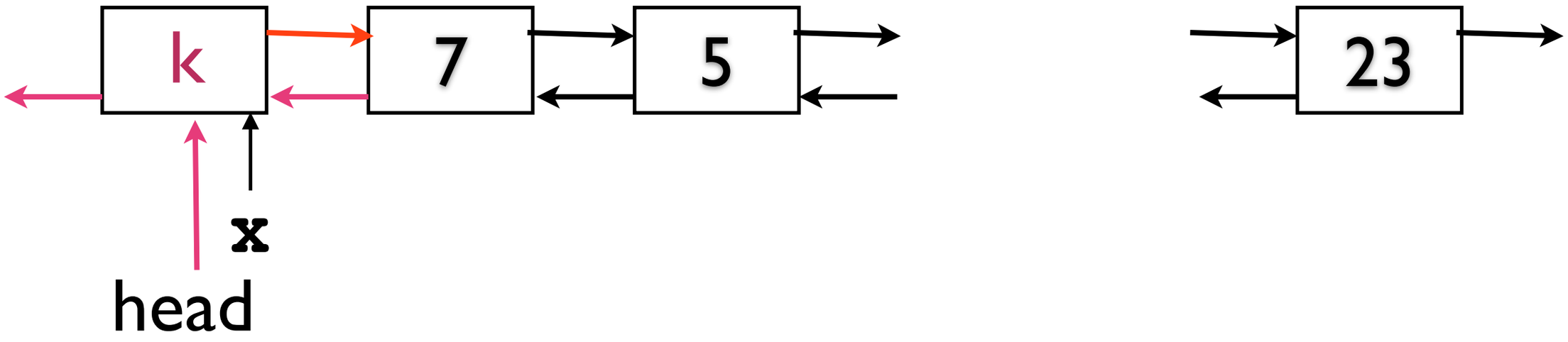
złożoność pesym. $\Theta(n)$, n - ilość elementów

wstawianie (na początek)



złożoność pesymistyczna *O(1)*,

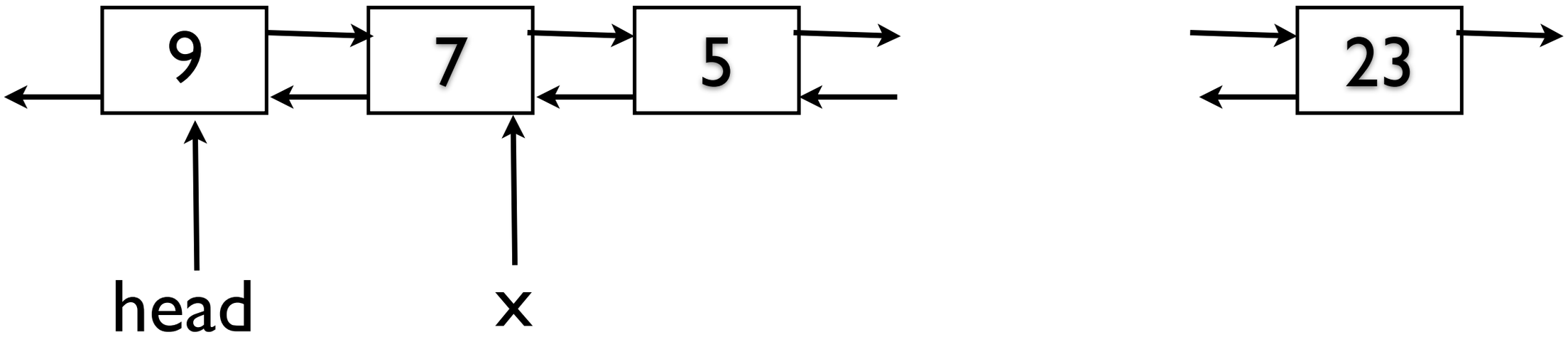
wstawianie (na początek)



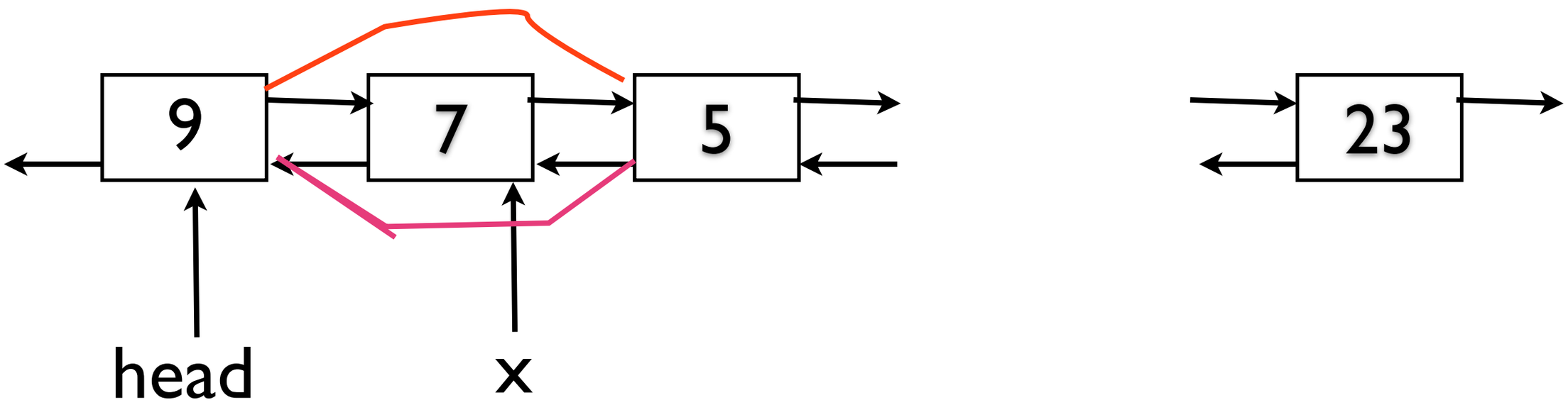

```
def listInsert(self, x):
    # wstawia wezel x do listy L
    # lista dwukierunkowa niecykliczna bez wartownika
        x.next = self.head
        if self.head != None:
            self.head.prev = x
        self.head = x
        x.prev = None
```

lista dowiązaniowa

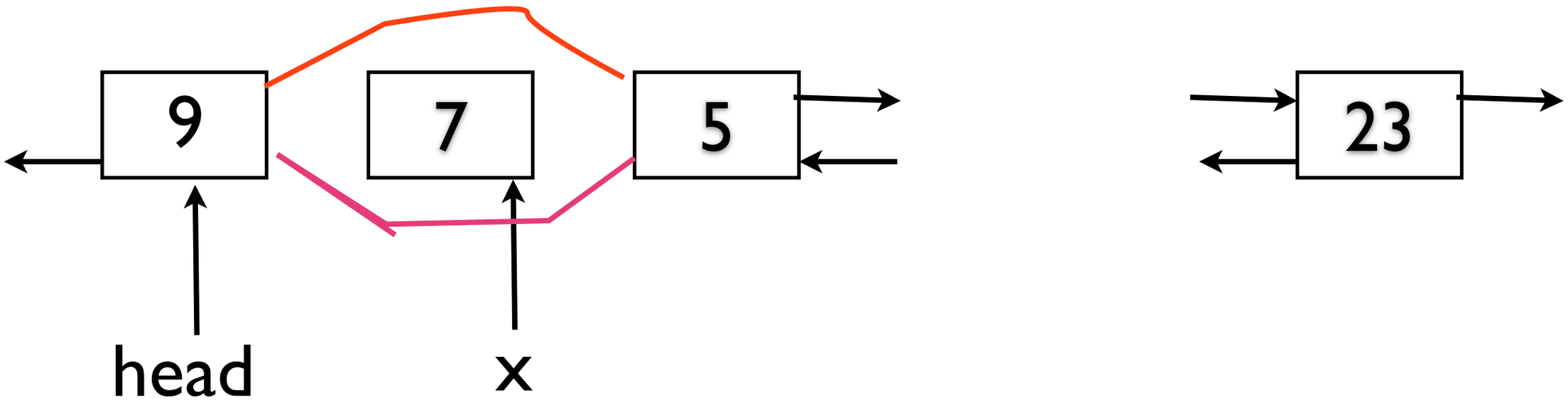
usuwanie (już wyszukanego elementu x)



usuwanie (już wyszukanego elementu x)

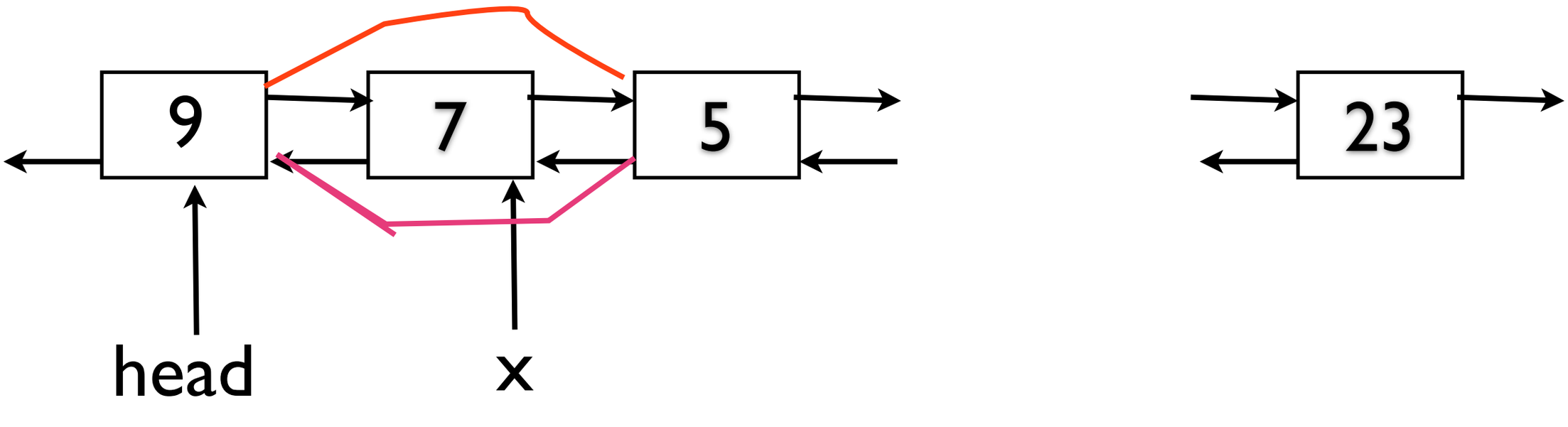


usuwanie (już wyszukanego elementu x)



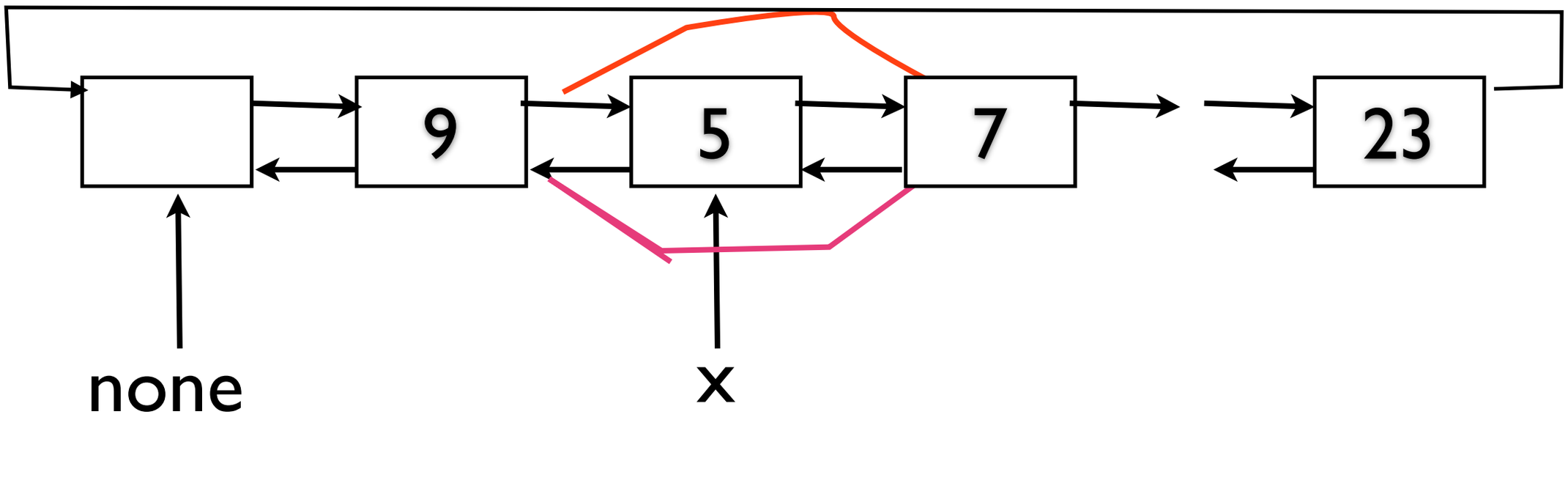
złożoność pesym. *O(1)*,

usuwanie (już wyszukanego elementu x)




```
def listDelete(self,x):
    # usuwa wezel x z listy
    # lista dwukierunkowa niecykliczna bez wartownika
        if x.prev != None:
            x.prev.next = x.next
        else:
            self.head = x.next
        if x.next != None:
            x.next.prev = x.prev
```

usuwanie - wersja z wartownikiem




```
listDelete(self,x):
# usuwa wezel x z listy
# lista dwukierunkowa cykliczna z wartownikiem
        x.prev.next = x.next
        x.next.prev = x.prev
```

szukanie - wersja z wartownikiem

```
def listSearch(self,k):
    # szuka wezla zawierajacego klucz k
    # lista dwukierunkowa cykliczna z wartownikiem
        x = self.none.next
        while x!=None and x.key!=k:
            x = x.next
        return x    # wynik “none” oznacza, ze szukanego klucza
                    # nie ma na liscie
```

złożoność pesym. $\Theta(n)$, n - ilość elementów