МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

Отчет

По лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: «Реализация связного списка»

Вариант: 17

Студент гр. 2309 _____ Шуббе Л. П.

Преподаватель _____ Пестерев Д. О.

Санкт-Петербург

2023

1. Постановка задачи

Реализовать объект в виде двусвязного списка с набором методов/функций.

2. Описание реализуемых классов, методов и функций

Классы

1) Element
Представляет из себя элемент будущего списка. В качестве данных хранит значение (любого типа), объявляемое при инициализации и ссылки на предыдущий и следующий элемент.

2) List2
Представляет из себя двусвязный список, хранящий в себе ссылки на первый (head), последний (tail) и текущий (cur) элемент, а также переменную, указывающую на пустоту списка (empty).

Методы класса Element

1) set(val)
Устанавливает значение данного элемента равное val.

2) get()
Получает значение данного элемента.

3) copy()
Возвращает элемент со значением равным значению данного элемента.

4) __eq__()
Необходимо для корректного сравнения двух элементов. Проверяет other на тип и производит проверку на равенство значений.

Методы класса List2

1) add_bottom(el)
Добавляет новый элемент в конец списка. Принимает на вход элемент (или значение элемента для создания), который необходимо добавить. Ничего не возвращает.

2) add_top(el)
Добавляет новый элемент в начало списка. Принимает на вход элемент (или значение элемента для создания), который необходимо добавить. Ничего не возвращает.

3) delete_bottom()
Удаляет последний элемент списка. Ничего не возвращает.

4) delete_top()
Удаляет первый элемент списка. Ничего не возвращает.

5) add_by_index(index, el)
Вставляет новый элемент в список по индексу (вставка перед элементом, который был ранее доступен по этому индексу). Принимает на вход индекс и элемент (или значение элемента для создания), который необходимо добавить. Ничего не возвращает.

6) get_by_index(index)
Метод для получения элемента по индексу. Принимает на вход этот индекс. Возвращает элемент (типа Element), находящийся по этому индексу.

7) delete_by_index(index)
Метод для удаления элемента по индексу. Принимает на вход этот индекс. Ничего не возвращает.

8) get_length()
Метод, используемый для получения длины списка.
Ничего не принимая, возвращает длину списка.

9) delete_all()
Метод, удаляющий весь список целиком. Ничего не принимает на вход и ничего не возвращает.

10) replace_by_index(index, el)
Метод для замены элемента списка на передаваемый. Принимает на вход индекс и элемент (или значение элемента для создания), на который необходимо заменить. Ничего не возвращает.

11) is_empty()
Метод, используемый для проверки списка на пустоту. Не принимает на вход аргументы. Данный метод возвращает True, если список пуст или False, если нет.

12) invert()
Метод, используемый для размещения элементов списка в обратном порядке.
Ничего не принимает на вход и ничего не возвращает.

13) merge_by_index(index, l2)
Метод, используемый для вставки другого списка начиная с индекса.
Принимает на вход индекс, где будет вставлен список и список для вставки. Ничего не возвращает.

14) merge_top(l2)
Метод, используемый для вставки другого списка в начало.
Принимает на вход список для вставки. Ничего не возвращает.

15) merge_bottom(l2)
Метод, используемый для вставки другого списка в конец.
Принимает на вход список для вставки. Ничего не возвращает.

16) contains(l2)
Метод, используемый для проверки наличия вхождения списка в другой.
Принимает на вход список, нахождение которого необходимо проверить в текущем.
Данный метод возвращает False если вхождения не существует или True если оно существует.

17) find(l2)
Метод, используемый для поиска первого вхождения списка в другой.
Принимает на вход список, нахождение которого необходимо проверить в текущем.
Данный метод возвращает номер индекса, с которого один список входит в другой или -1 если вхождения не существует.

18) rfind(l2)
Метод, используемый для поиска последнего вхождения списка в другой.
Принимает на вход список, нахождение которого необходимо проверить в текущем.
Данный метод возвращает номер индекса, с которого один список входит в другой или -1 если вхождения не существует.

19) swap_by_index(index1, index2)
Метод, используемый для обмена двумя элементами списка по индексам. Принимает на вход эти два индекса и ничего не возвращает.

20) add_first(el)
Метод для добавления первого элемента в пустой список. Принимает на вход элемент или его значение. Добавляет элемент и меняет переменную is_empty.

21) print()
Метод для вывод списка на экран. Ничего не принимает на вход и ничего не возвращает.
Вывод осуществляется стандартным print через пробел.

Другие функции

1) copy(l2)
Функция для копирования некоторого списка. Принимает на вход список для копирования.
Создает копию каждого элемента и возвращает ссылку на новый список из этих элементов.

2) generate(N, min_el, max_el)
Функция для генерации случайного списка. Принимает на вход желаемое количество элементов в списке, минимальный и максимальный возможный элемент. Генерирует элементы со значениями от минимального до максимального (включая обе границы) и возвращает ссылку на новый список из этих элементов.

3) get_time(func, *args)
Функция для определения времени выполнения некоторой функции. Принимает на вход функцию и аргументы. Вызывает эту функцию с переданными аргументами и возвращает время в секундах, затраченное на выполнение.

3. Временная и практическая сложность

Все методы были воспроизведены и время их выполнения было зафиксировано на списках длиной от 10^5 до 10^6 с шагом 10^5 при элементах списка, сгенерированных случайно в диапазоне от 0 до 10^6. Если некоторый метод требует некоторый индекс или другой список, то они также генерировались случайно (индекс — так чтобы он попадал в список, а другие списки — так же как и оригинальный, с той же длиной). Для методов с неконстантной асимптотикой были построены графики зависимости времени выполнения от размера списка.

1. add_bottom – O(1)

   Изменяется ссылка на tail и соответствующие ссылки у вставляемого элемента и предыдущего tail поэтому зависимости от N не будет.

```
100000 4.0531158447265625e-06
200000 1.430511474609375e-06
300000 1.6689300537109375e-06
400000 1.9073486328125e-06
500000 1.6689300537109375e-06
600000 1.9073486328125e-06
700000 1.6689300537109375e-06
800000 1.6689300537109375e-06
900000 3.5762786865234375e-06
1000000 1.430511474609375e-06
```

2. add_top – O(1)
   Изменяется ссылка на head и соответствующие ссылки у вставляемого элемента и предыдущего head поэтому зависимости от N не будет.

```
100000 3.0994415283203125e-06
200000 2.86102294921875e-06
300000 2.384185791015625e-06
400000 2.86102294921875e-06
500000 2.384185791015625e-06
600000 3.814697265625e-06
700000 3.337860107421875e-06
800000 3.337860107421875e-06
900000 3.814697265625e-06
1000000 4.76837158203125e-06
```

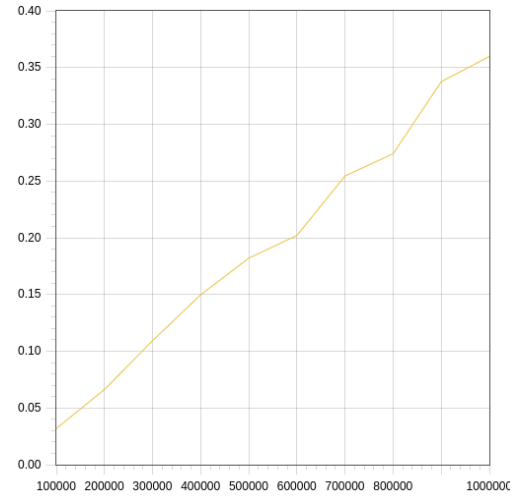
3. delete_bottom - O(1)
   Изменяется ссылка на tail и соответствующие ссылки у удаляемого элемента и нового tail поэтому зависимости от N не будет.

```
100000 9.059906005859375e-06
200000 2.384185791015625e-06
300000 1.9073486328125e-06
400000 2.86102294921875e-06
500000 2.384185791015625e-06
600000 2.384185791015625e-06
700000 2.1457672119140625e-06
800000 2.384185791015625e-06
900000 3.337860107421875e-06
1000000 4.76837158203125e-06
```

4. delete_top - O(1)
   Изменяется ссылка на head и соответствующие ссылки у удаляемого элемента и нового head поэтому зависимости от N не будет.

```
100000 3.814697265625e-06
200000 3.337860107421875e-06
300000 3.0994415283203125e-06
400000 4.5299530029296875e-06
500000 3.0994415283203125e-06
600000 5.245208740234375e-06
700000 8.106231689453125e-06
800000 3.5762786865234375e-06
900000 3.337860107421875e-06
1000000 3.337860107421875e-06
```

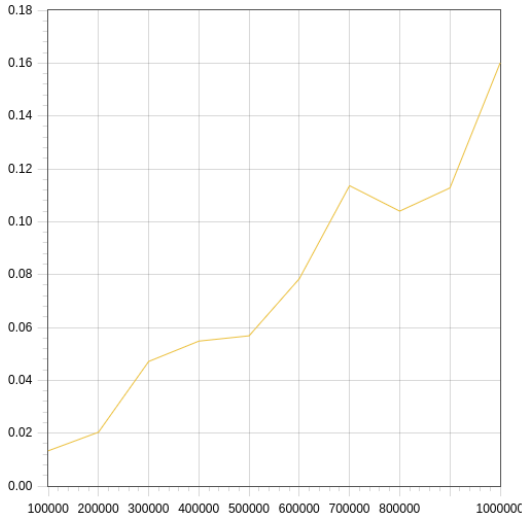
5. add_by_index - O(N)




```
100000 0.03187108039855957
200000 0.0662384033203125
300000 0.10931873321533203
400000 0.14974331855773926
500000 0.18204474449157715
600000 0.2020125389099121
700000 0.2544593811035156
800000 0.27406811714172363
900000 0.3375709056854248
1000000 0.3597557544708252
```

Метод вызывает get_length и get_by_index после чего осуществляет константное количество операций присваивания поэтому асимптотика будет той же — O(N). График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

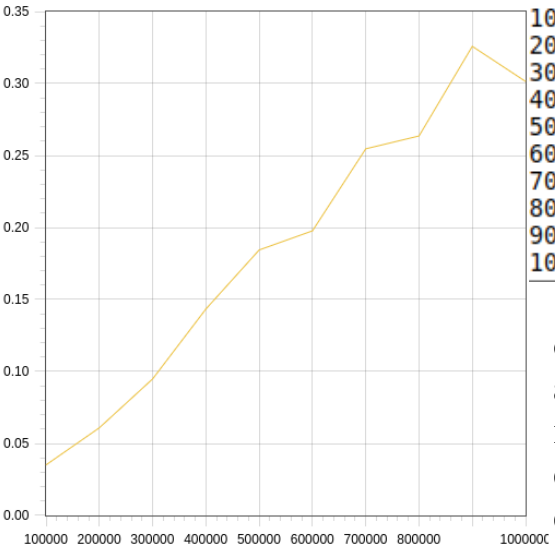
6. get_by_index - O(N)




```
100000 0.01332235336303711
200000 0.020357608795166016
300000 0.047116994857788086
400000 0.05482292175292969
500000 0.05682730674743652
600000 0.07841014862060547
700000 0.11362504959106445
800000 0.10400223731994629
900000 0.1127626895904541
1000000 0.16005539894104004
```

Метод вызывает get_length, а затем в цикле совершает по одной операции присваивания, всего index раз. Поэтому асимптотика будет O(N).График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

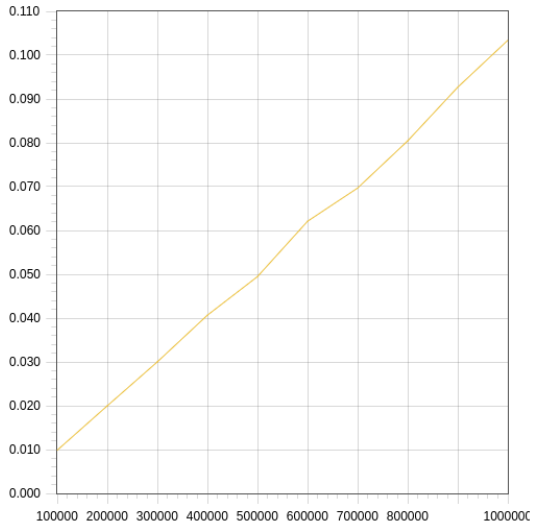
7. delete_by_index - O(N)




```
100000 0.009844303131103516
200000 0.019978761672973633
300000 0.030008792877197266
400000 0.04066300392150879
500000 0.049466848373413086
600000 0.06209373474121094
700000 0.06967520713806152
800000 0.08044290542602539
900000 0.09268927574157715
1000000 0.10339808464050293
```

Метод вызывает get_length и get_by_index после чего осуществляет константное количество операций присваивания, поэтому асимптотика будет той же — O(N). График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

8. get_length - O(N)




```
100000 0.03511929512023926
200000 0.06087923049926758
300000 0.09481406211853027
400000 0.14337491989135742
500000 0.18440794944763184
600000 0.19767498970031738
700000 0.25464677810668945
800000 0.2635796070098877
900000 0.32582640647888184
1000000 0.3014390468597412
```

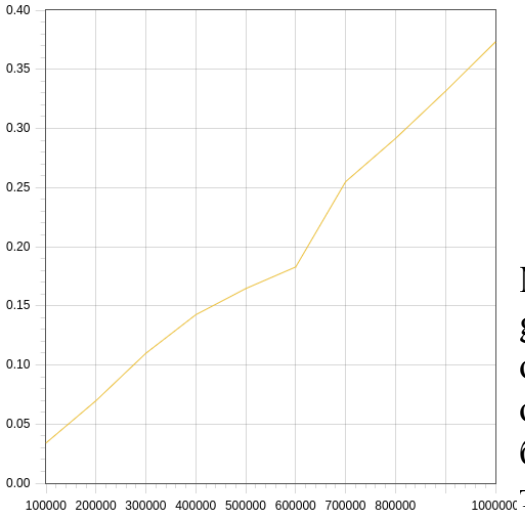
Метод осуществляет полное прохождение по списку с начала до конца (пока ссылки на следующий элемент не окажется), поэтому время линейно зависит от длины.

10. delete_all - O(1)

Удаляет ссылки на первый, последний и текущий элемент независимо от длины списка поэтому время выполнения константное.

```
100000 1.9073486328125e-06
200000 2.1457672119140625e-06
300000 1.0251998901367188e-05
400000 2.384185791015625e-06
500000 2.6226043701171875e-06
600000 2.6226043701171875e-06
700000 2.384185791015625e-06
800000 8.106231689453125e-06
900000 2.86102294921875e-06
1000000 2.1457672119140625e-06
```

11. replace_by_index - O(N)




```
100000 0.03401541709899902
200000 0.06973671913146973
300000 0.10988402366638184
400000 0.1424729824066162
500000 0.16457009315490723
600000 0.18282389640808105
700000 0.2549729347229004
800000 0.2915492057800293
900000 0.3314509391784668
1000000 0.37314295768737793
```

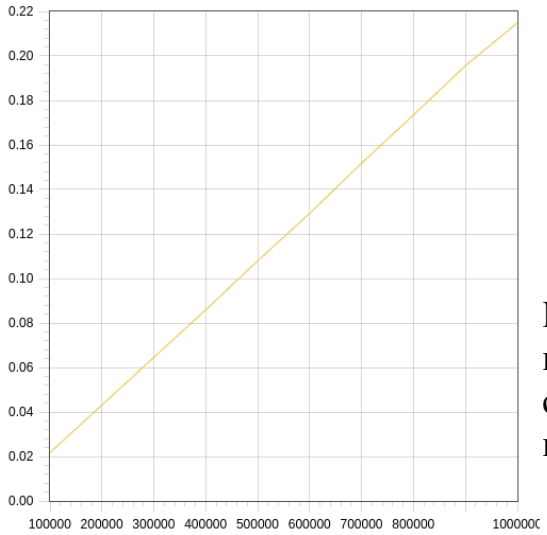
Метод вызывает get_length и get_by_index после чего осуществляет константное количество операций присваивания, поэтому асимптотика будет той же — O(N). График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

12. is_empty - O(1)

Этот метод возвращает поле is_empty, время константное.

```
100000 1.9073486328125e-06
200000 9.5367431640625e-07
300000 9.5367431640625e-07
400000 1.6689300537109375e-06
500000 2.1457672119140625e-06
600000 1.430511474609375e-06
700000 1.1920928955078125e-06
800000 7.152557373046875e-07
900000 1.1920928955078125e-06
1000000 7.152557373046875e-07
```

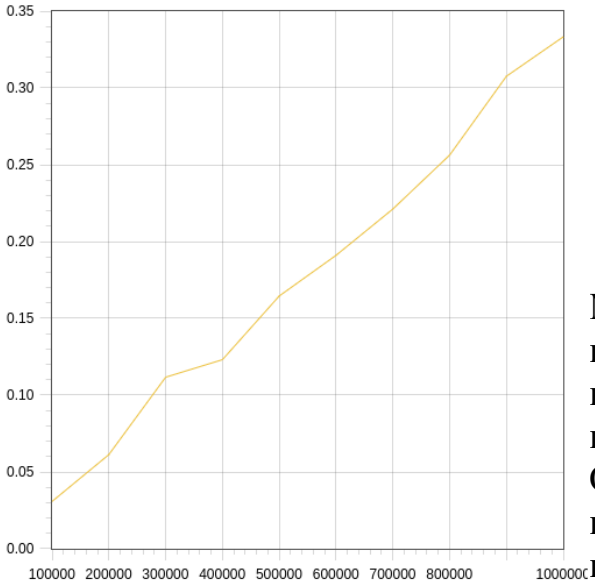
13. invert - O(N)




```
100000 0.021698474884033203
200000 0.04308319091796875
300000 0.06453704833984375
400000 0.08592557907104492
500000 0.10794639587402344
600000 0.12922334671020508
700000 0.15184545516967773
800000 0.17349958419799805
900000 0.1956179141998291
1000000 0.21489167213439941
```

Метод осуществляет полное прохождение по списку с начала до конца (пока ссылки на следующий элемент не окажется), поэтому время линейно зависит от длины.

14. merge_by_index - O(N)




```
100000 0.030792713165283203
200000 0.06129193305969238
300000 0.11168169975280762
400000 0.12309503555297852
500000 0.16454625129699707
600000 0.191025972366333
700000 0.22115755081176758
800000 0.25625038146972656
900000 0.3076002597808838
1000000 0.33324360847473145
```

Метод вызывает get_length и get_by_index после чего осуществляет константное количество операций присваивания, поэтому асимптотика будет той же — O(N). График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

15. merge_bottom - O(1)

Метод меняет ссылки конечных элементов собственного и передаваемого списка и собственную ссылку на последний элемент, поэтому время константное.

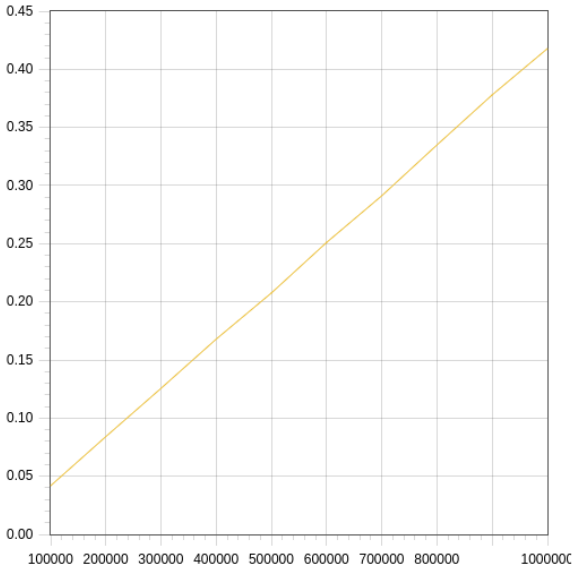
```
100000 2.384185791015625e-06
200000 1.6689300537109375e-06
300000 2.1457672119140625e-06
400000 2.1457672119140625e-06
500000 5.9604644775390625e-06
600000 2.384185791015625e-06
700000 2.1457672119140625e-06
800000 2.1457672119140625e-06
900000 5.4836273193359375e-06
1000000 2.384185791015625e-06
```

16. merge_top - O(1)

Метод меняет ссылки конечных элементов собственного и передаваемого списка и собственную ссылку на первый элемент, поэтому время константное.

```
100000 2.384185791015625e-06
200000 2.1457672119140625e-06
300000 2.384185791015625e-06
400000 2.384185791015625e-06
500000 2.384185791015625e-06
600000 2.384185791015625e-06
700000 2.86102294921875e-06
800000 2.6226043701171875e-06
900000 1.9073486328125e-06
1000000 2.1457672119140625e-06
```

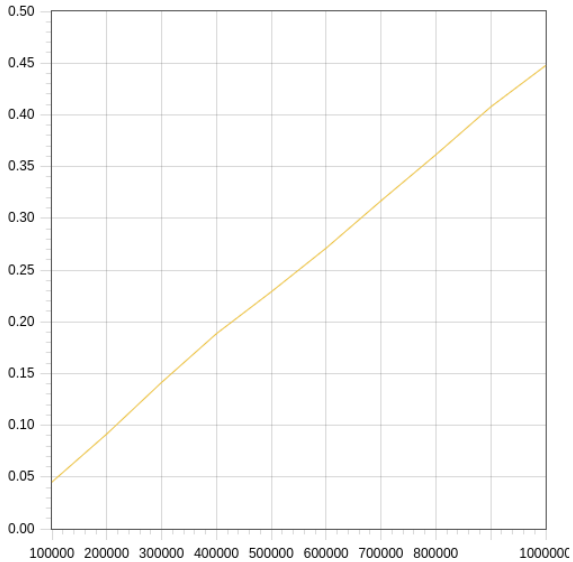
17. contains - $O(N^2)$




```
100000 0.04183149337768555
200000 0.08409404754638672
300000 0.1256103515625
400000 0.1679694652557373
500000 0.20764851570129395
600000 0.251126766204834
700000 0.2912781238555908
800000 0.33499908447265625
900000 0.37804388999938965
1000000 0.4178335666656494
```

Метод осуществляет полное прохождение по списку с начала до конца (пока ссылки на следующий элемент не окажется). При каждом прохождении элемента алгоритм пытается проверить содержание другого списка начиная с текущего элемента. В худшем случае это приведёт в прохождению списка до конца при прохождении каждого элемента, то есть $O(N^2)$. Однако при генерации случайных списков ситуация длительного прохождения маловероятно поэтому в действительности видна линейная зависимость.

18. find - $O(N^2)$




```
100000 0.04521441459655762
200000 0.09164547920227051
300000 0.14150238037109375
400000 0.1885986328125
500000 0.22904586791992188
600000 0.2712540626525879
700000 0.31697607040405273
800000 0.361480712890625
900000 0.4076828956604004
1000000 0.4475862979888916
```

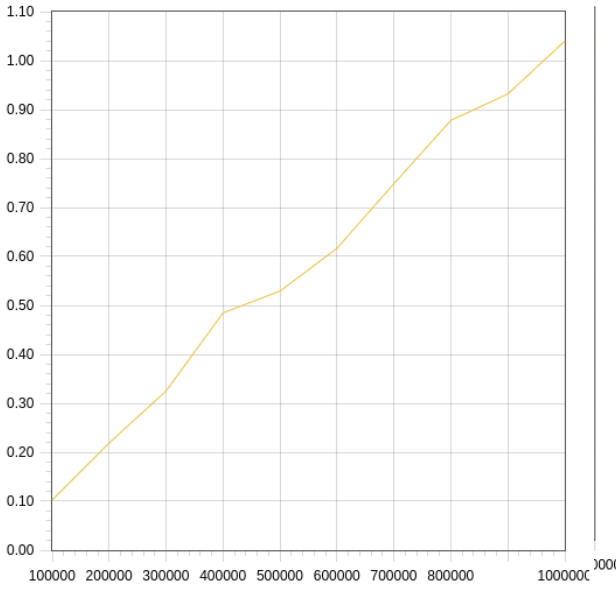
Данный метод работает также как и contains, возвращая другие значения, используя хранение счётчика, который ни коем образом не влияет на асимптотику.

19. rfind - $O(N^2)$

Данный метод работает также как и contains, возвращая другие значения, используя хранение счётчика, который ни коем образом не влияет на асимптотику.

```
100000 0.05777740478515625
200000 0.11522817611694336
300000 0.1722884178161621
400000 0.22645115852355957
500000 0.28179335594177246
600000 0.33992481231689453
700000 0.39727091789245605
800000 0.451643705368042
900000 0.5096137523651123
1000000 0.5651373863220215
```

20. swap_by_index - O(N)




```
100000 0.10262942314147949
200000 0.2190089225769043
300000 0.3252871036529541
400000 0.48471832275390625
500000 0.5292599201202393
600000 0.616642951965332
700000 0.748685359954834
800000 0.8779890537261963
900000 0.9318709373474121
1000000 1.0398902893066406
```

Метод вызывает get_length и get_by_index после чего осуществляет константное количество операций присваивания, поэтому асимптотика будет той же — O(N). График искривлен из-за того что индекс принимает случайное значение от нуля до N, а не строго N.

4. Пример работы

```
Enter method index to get its time to run: 1
N      Time, seconds
100000 4.0531158447265625e-06
200000 1.430511474609375e-06
300000 1.6689300537109375e-06
400000 1.9073486328125e-06
500000 1.6689300537109375e-06
600000 1.9073486328125e-06
700000 1.6689300537109375e-06
800000 1.6689300537109375e-06
900000 3.5762786865234375e-06
1000000 1.430511474609375e-06
```

Пример работы с вызовом первого метода. Примеры с исполнением других методов приведены по ходу разбора временной сложности.

5. Листинг

```
import random
import time

class Element:
  def __init__(self, val=None, prev=None, next=None):
    self.val = val
    self.prev = prev
    self.next = next

  def __eq__(self, other):  # Описание механизма сравнения двух элементов
    if other is None:
      return False
    if not isinstance(other, int | Element):
      raise TypeError
    return self.val == (other if isinstance(other, int) else other.val)

  def set(self, val):  # Установить значение
    self.val = val
    return self.val

  def get(self):  # Получить значение
    return self.val

  def copy(self):  # Копирование себя
    return Element(self.get())


class List2:
  def __init__(self, *elements):
    self.head = None
    self.tail = None
    self.cur = None
    self.empty = True
    for el in elements:
      self.add_bottom(el)

  def is_empty(self):  # Проверка на пустоту списка
    return self.empty

  def get_length(self):  # Получение размера списка
    counter = 0
    self.cur = self.head
    while self.cur is not None:
      counter += 1
      self.cur = self.cur.next
    return counter
```

```
def add_first(self, el):  # Добавление первого элемента в пустой список
  if not isinstance(el, Element):
    el = Element(el)
  self.head = el
  self.tail = el
  self.empty = False

def print(self):  # Вывод списка на экран
  self.cur = self.head
  while self.cur is not None:
    print(self.cur.get(), end=' ')
    self.cur = self.cur.next
  print('\n', end='')

def add_top(self, el):  # Добавление в начало списка
  if not isinstance(el, Element):
    el = Element(el)
  if self.is_empty():
    self.add_first(el)
  else:
    self.head.prev = el
    el.next = self.head
    self.head = el

def add_bottom(self, el):  # Добавление в конец списка
  if not isinstance(el, Element):
    el = Element(el)
  if self.is_empty():
    self.add_first(el)
  else:
    self.tail.next = el
    el.prev = self.tail
    self.tail = el

def delete_top(self):  # Удаление первого элемента
  if self.is_empty() or self.head.next is None:
    self.head = None
    self.tail = None
    self.empty = True
  else:
    self.head.next.prev = None
    self.head = self.head.next

def delete_bottom(self):  # Удаление последнего элемента
  if self.is_empty() or self.tail.prev is None:
    self.head = None
```

```
            self.tail = None
            self.empty = True
        else:
            self.tail.prev.next = None
            self.tail = self.tail.prev

    def get_by_index(self, index):  # Получение элемента по индексу
        if index < 0 or index >= self.get_length():
            raise Exception
        self.cur = self.head
        for i in range(index):
            self.cur = self.cur.next
        return self.cur

    def delete_by_index(self, index):  # Удаление элемента по индексу
        if index < 0 or index >= self.get_length():
            raise Exception
        elif index == 0:
            self.delete_top()
        elif index == self.get_length() - 1:
            self.delete_bottom()
        else:
            self.cur = self.get_by_index(index)
            self.cur.prev.next = self.cur.next
            self.cur.next.prev = self.cur.prev

    def add_by_index(self, index, el):  # Добавление элемента по индексу
        if not isinstance(el, Element):
            el = Element(el)
        if index < 0 or index >= self.get_length():
            raise Exception
        elif index == 0:
            self.add_top(el)
        elif index == self.get_length() - 1:
            self.add_bottom(el)
        else:
            self.cur = self.get_by_index(index)
            el.prev = self.cur.prev
            el.next = self.cur
            self.cur.prev.next = el
            self.cur.prev = el

    def delete_all(self):  # Удаление всех элементов списка
        self.head = None
        self.tail = None
        self.cur = None
        self.empty = True
```

```
  def replace_by_index(self, index, el):  # Замена элемента по индексу на передаваемый элемент
    if not isinstance(el, Element):
      el = Element(el)
    if index < 0 or index >= self.get_length():
      raise Exception
    elif index == 0:
      self.delete_top()
      self.add_top(el)
    elif index == self.get_length() - 1:
      self.delete_bottom()
      self.add_bottom(el)
    else:
      self.cur = self.get_by_index(index)
      el.prev = self.cur.prev
      el.next = self.cur.next
      self.cur.prev.next = el
      self.cur.next.prev = el

  def swap_by_index(self, index1, index2):  # Обмен двух элементов списка по индексам
    index1, index2 = min(index1, index2), max(index1, index2)
    if index1 < 0 or index2 >= self.get_length():
      raise Exception
    elif index1 == index2:
      return
    else:
      self.cur = self.head
      el1 = self.get_by_index(index1).copy()
      el2 = self.get_by_index(index2).copy()
      self.replace_by_index(index1, el2)
      self.replace_by_index(index2, el1)

  def invert(self):  # Меняет порядок элементов в списке на обратный
    self.cur = self.head
    while self.cur is not None:
      self.cur.prev, self.cur.next = self.cur.next, self.cur.prev
      self.cur = self.cur.prev
    self.head, self.tail = self.tail, self.head

  def merge_top(self, l2):  # Вставка другого списка в начало
    l2.tail.next = self.head
    self.head.prev = l2.tail
    self.head = l2.head

  def merge_bottom(self, l2):  # Вставка другого списка в конец
    l2.head.prev = self.tail
```

```
    self.tail.next = l2.head
    self.tail = l2.tail

  def merge_by_index(self, index, l2):  # Вставка другого списка начиная с индекса
    if index < 0 or index >= self.get_length():
      raise Exception
    elif index == 0:
      self.merge_top(l2)
    elif index == self.get_length() - 1:
      self.merge_bottom(l2)
    else:
      self.cur = self.get_by_index(index)
      l2.head.prev = self.cur.prev
      l2.tail.next = self.cur
      self.cur.prev.next = l2.head
      self.cur.prev = l2.tail

  def contains(self, l2):  # Проверка на содержание другого списка
    remembered = self.head
    while remembered is not None:
      self.cur = remembered
      l2.cur = l2.head
      while self.cur == l2.cur:
        self.cur = self.cur.next
        l2.cur = l2.cur.next
        if l2.cur is None:
          return True
      remembered = remembered.next
    return False

  def find(self, l2):  # Поиск первого вхождения другого списка
    remembered = self.head
    index = 0
    while remembered is not None:
      self.cur = remembered
      l2.cur = l2.head
      while self.cur == l2.cur:
        self.cur = self.cur.next
        l2.cur = l2.cur.next
        if l2.cur is None:
          return index
      remembered = remembered.next
      index += 1
    return -1

  def rfind(self, l2):  # Поиск последнего вхождения другого списка
    remembered = self.tail
```

```
    index = self.get_length()
    while remembered is not None:
      self.cur = remembered
      l2.cur = l2.tail
      while self.cur == l2.cur:
        self.cur = self.cur.prev
        l2.cur = l2.cur.prev
        if l2.cur is None:
          return index - l2.get_length()
      remembered = remembered.prev
      index -= 1
    return -1


def copy(l2):  # Копирование списка
  new_list = List2()
  l2.cur = l2.head
  while l2.cur is not None:
    new_list.add_bottom(l2.cur.copy())
    l2.cur = l2.cur.next
  return new_list


def generate(N, min_el, max_el):  # Генерация случайного списка
  new_list = List2()
  for i in range(N):
    new_list.add_bottom(random.randint(min_el, max_el))
  return new_list


def get_time(func, *args):  # Получение времени выполнения функции
  ts = time.time()
  func(*args)
  return time.time() - ts


run_dict = {  # Примеры вызова методов со случайными параметрами
  "1": lambda N: get_time(generate(N, min_el, max_el).add_bottom, random.randint(min_el,
max_el)),
  "2": lambda N: get_time(generate(N, min_el, max_el).add_top, random.randint(min_el,
max_el)),
  "3": lambda N: get_time(generate(N, min_el, max_el).delete_bottom),
  "4": lambda N: get_time(generate(N, min_el, max_el).delete_top),
  "5": lambda N: get_time(generate(N, min_el, max_el).add_by_index, random.randint(1, N - 1),
random.randint(min_el, max_el)),
  "6": lambda N: get_time(generate(N, min_el, max_el).get_by_index, random.randint(1, N - 1)),
  "7": lambda N: get_time(generate(N, min_el, max_el).delete_by_index, random.randint(1, N -
1)),
  "8": lambda N: get_time(generate(N, min_el, max_el).get_length),
```

```
  "9": lambda N: get_time(generate(N, min_el, max_el).delete_all),
  "10": lambda N: get_time(generate(N, min_el, max_el).replace_by_index, random.randint(1, N - 1), random.randint(min_el, max_el)),
  "11": lambda N: get_time(generate(N, min_el, max_el).is_empty),
  "12": lambda N: get_time(generate(N, min_el, max_el).invert),
  "13": lambda N: get_time(generate(N, min_el, max_el).merge_by_index, random.randint(1, N - 1), generate(N, min_el, max_el)),
  "14": lambda N: get_time(generate(N, min_el, max_el).merge_bottom, generate(N, min_el, max_el)),
  "15": lambda N: get_time(generate(N, min_el, max_el).merge_top,  generate(N, min_el, max_el)),
  "16": lambda N: get_time(generate(N, min_el, max_el).contains, generate(N, min_el, max_el)),
  "17": lambda N: get_time(generate(N, min_el, max_el).find, generate(N, min_el, max_el)),
  "18": lambda N: get_time(generate(N, min_el, max_el).rfind, generate(N, min_el, max_el)),
  "19": lambda N: get_time(generate(N, min_el, max_el).swap_by_index, random.randint(1, N - 1), random.randint(1, N - 1)),
  "": lambda N: exit()
}
min_el = 0  # Минимальное допустимое значение элемента
max_el = 10**6  # Максимальное допустимое значение элемента
points = range(10**5, 10**6 + 1, 10**5)  # Точки (количества элементов в списке для разных вызовов функции)

if __name__ == "__main__":
  while True:
    case = input("Enter method index to get its time to run: ")
    print("N      Time, seconds")
    for N in points:
      print(N, run_dict[case](N))
```