linkedlist.md

기말03, 컴퓨터의 개념 및 실습 수업용 파일입니다. 언어학과, 2021-12659 박유나가 작성하였습니다.

코드 설명

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

    def __repr__(self):
        return repr(self.value)

# 노드를 형성하는 코드입니다.

class LinkedList:
    def __init__(self):
        self.head = None
        self.tail = None
        self.__length = 0

    def __len__(self):
        return self.__length

    def __iter__(self):
        self.current = self.head
        return self

    def __next__(self):
        if self.current is None:
            raise StopIteration

        value = self.current.value
        self.current = self.current.next

        return value

    def __repr__(self):
        return repr(list(self))

#매직 매소드를 이용하여 초기화, 길이반환, 반복자,  표현 등을 하여주었습ㄴ디.


    def append(self, value):
        node = Node(value)
        if self.tail is None:
            self.head = self.tail = node
        else:
```

```
            node.prev = self.tail
            self.tail.next = node
            self.tail = node

        self.__length += 1

    def appendleft(self, value):
        node = Node(value)
        if self.tail is None:
            self.head = self.tail = node
        else:
            node.next = self.head
            self.head.prev = node
            self.head = node

        self.__length += 1

# 리스트에 값을 추가할 수 있도록 하였습니다.

    def pop(self):
        if self.tail is None:
           raise Exception('LinkedList is empty')

        target = self.tail.value
        self.tail = self.tail.prev
        self.tail.next = None
        self.__length -= 1
        return target


    def popleft(self):
        if self.head is None:
            raise Exception('LinkedList is empty')
        target = self.head.value
        self.head = self.head.next
        self.head.prev = None
        self.__length -= 1
        return target

# 마지막 원소를 리스트에서 제거한 후 반환하도록 구현하였습니다.

def remove(self, value):
        if self.head is None:
            raise Exception('LinkedList is empty')

        if self.head.value == value:
            target = self.head
            self.head = self.head.next
            del target
        else:
            node = self.head
            while node.next:
                if node.next.value == value:
                    target = node.next
```

```
                    node.next = node.next.next
                    del target
                else:
                    node = node.next

    def reverse(self):
        right = None
        curr = self.head
        while curr:
            right = curr.prev
            curr.prev = curr.next
            curr.next = right
            right = curr
            curr = curr.prev
        if right:
            self.head = right
        return self

    def count(self, value):
        count = 0
        for item in self:
            if item == value:
                count += 1
        return count

# remove(), reverse(), count()함수를 구현하여 특정값 제거, 역순환 출력, 값의 개수세기
를 가능하도록 하였습니다.
```

성능비교

```
start_time = time.time()
ll1 = LinkedList()
for i in range(1, 10000):
    ll1.append(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

start_time = time.time()
l1 = []
for i in range(1, 10000):
    l1.append(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

# 0.027225494384765625 vs 0.034734249114990234

start_time = time.time()
ll2 = LinkedList()
```

```
for i in range(1, 10000):
    ll1.appendleft(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

start_time = time.time()
l2 = []
for i in range(1, 10000):
    l2.insert(0, i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

#0.029201507568359375 vs 0.031928300857543945

start_time = time.time()
for i in range(1, 10000):
    ll1.pop(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

start_time = time.time()
for i in range(1, 10000):
    l1.pop(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

# 0.516134977340698242 vs 0.6081357002258301

start_time = time.time()
for i in range(1, 10000):
    ll2.popleft(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time

start_time = time.time()
for i in range(1, 10000):
    l2.pop(i)
end_time = time.time()
elapsed_time = end_time - start_time
elapsed_time


# 0.514904260635375977 vs0.6564867496490479
```

근소하게나마 이중연결리스트로 만든 함수가 빠른 것을 확인할 수있었습니다.