HW4: Data Structure in Mathematics

Saen-Anan Bunyasiwa (SCIM 6105626)
Monchita Toopsuwan (SCIM 6105731)

February 3, 2020

Assignment IV

Consider two arrays, as shown below, which are created from the class DymanicArray (See the file `chap05_dynamic_array.ipynb` from class.)

$A =$

0	1	2	3	4	5	6	7
7	5	2	11	3			

$B =$

0	1	2	3	4	5
7	5	2	11	3	

Draw the array A or B and find the size in bytes after applying each of the following methods from the file `chap05_dynamic_array.ipynb`. Assume that it takes 72 bytes to create an empty array on a 64-bit computer.

Instructions

1. `append` the numbers 21 and 22 to the array A.
2. `append` the numbers 21 and 22 to the array B.
3. `insert` the number 21 to the array A at the index 1.
4. `insert` the number 21 to the array B at the index 3.
5. `remove` the number 7 from array A.
6. `remove` the number 11 from array B.

Solutions

Initialization to match the figure

```
import sys

if __name__ == '__main__':
        # From the assignment, we must set the initial value of empty array
        # which is 64 bytes but in the question it asssume to be 72.
        # Therefore, we set the value to
        C = DynamicArray() # empty array
        EmpArrB = 72 - sys.getsizeof(C) # assumption - actual value

        A = DynamicArray()
        B = DynamicArray()

        A._resize(8)
        B._resize(6)

        # initial state of both arrays
        a = [7, 5, 2, 11, 3]
        b = [7, 5, 2, 11, 3]

        # Looping to insert the elements in the figure
        for i in range(len(a)): A.insert(i, a[i])
        for i in range(A.__len__()): print(A.__getitem__(i))

        for i in range(len(b)): B.insert(i, b[i])
        for i in range(B.__len__()): print(B.__getitem__(i))

        print('Initial length of A is: ', A.__len__())
        print('Initial length of B is: ', B.__len__())
```

Continuation

Appendix Insertion and Removal

```
if __name__ == '__main__':

        # Appendix
        A.append(21); A.append(22)
        print('Now, the size of the Array A is: ', sys.getsizeof(A) + EmpArrB)
        B.append(21); B.append(22)
        print('Now, the size of the Array B is: ', sys.getsizeof(B) + EmpArrB)

        # Insertion
        A.insert(1, 21); B.insert(3, 21)
        print('Now the size of array A after insertion is: ',
                sys.getsizeof(A) + EmpArrB, ', with length: ', A.__len__())

        print('Now the size of array B after insertion is: ',
                sys.getsizeof(B) + EmpArrB, ', with length: ', B.__len__())

        # Removal
        A.remove(7); B.remove(11)
        print('Now the size of array A after removal is: ', sys.getsizeof(A)
                + EmpArrB, ', with length: ', A.__len__())
        print('Now the size of array B after removal is: ', sys.getsizeof(B)
                + EmpArrB, ', with length: ', B.__len__())
```

Output

Output of Initialization

```
7
5
2
11
3

7
5
2
11
3

Initial length of A is: 8
Initial length of B is: 6
```

Output of the solutions

```
Now, the size of the Array A is: 72
Now, the size of the Array B is: 72
Now the size of array A after insertion is: 72 , with length: 8
Now the size of array B after insertion is: 72 , with length: 8
Now the size of array A after removal is: 72 , with length: 7
Now the size of array B after removal is: 72 , with length: 7
```

R - 5.4

Our `DynamicArray` class, as given in Code Fragment 5.3, does not support use of negative indices with `__getitem__` Update that method to better match the semantics of a python list

Original Code Fragment

```
def __getitem__(self, k):
        """Return element at index k."""
        if not 0 <= k < self._n:
                raise IndexError('invalid index')
        return self._A[k]                # retrieve from array
```

Edited Source Code

```
def __getitem__(self, k):
        #Accepting negative index
        if 0 <= k < self._n:
                return self._A[k]
        else:
                return self._A[self._n+k]

print(A.__getitem__(1))
print(A.__getitem__(-A.__len__()+1))
```

Output

```
5
5
```