ECE374
Lab 1 Report

Name: Zhang Yichi
Student ID: 3180111309

September 22, 2020

1 Topic

This is a warm-up lab about the set and operations over sets and between sets. We are required to perform manipulating elements from the sets and operations like union, intersection, difference, length, subtraction, etc.

2 Code

[1]:
```
class Set:
    # a) create a set
    def __init__(self,contents=[],loadMax=0.75,loadMin=0.25):
        self.items = [None] * 20
        self.numItems = 0
        self.loadMax = loadMax
        self.loadMin = loadMin
        for e in contents:
            self.add(e)

    def __contains__(self,item):
        index = hash(item) % len(self.items)
        while self.items[index] != None:
            if self.items[index] == item:
                return True
            index = (index + 1) % len(self.items)
        return False

    # b) iterate over sets
    def __iter__(self):
        for e in self.items:
            if e != None and type(e) != Set.__Placeholder:
                yield e

    ## punlic functions
```

```
    # c) add
    def add(self,item):
        if Set.__add(item,self.items):
            self.numItems += 1
            if self.numItems / len(self.items) >= self.loadMax:
                self.items = Set.__rehash(self.items,[None]*(2*len(self.items)))

    def addlist(self,items):
        for e in items:
            self.add(e)

    # d) remove
    def remove(self,item):
        if Set.__remove(item,self.items):
            self.numItems -= 1
            if self.numItems / len(self.items) <= self.loadMin:
                self.items = Set.__rehash(self.items,[None]*(len(self.items)//
↪2))
        else:
            raise KeyError("Item not in the set")

    def removelist(self,items):
        for e in items:
            self.remove(e)

    # e) delete -  remove an item from a set if it is present in the set
    def delete(self,item):
        if Set.__remove(item,self.items):
            self.numItems -= 1
            if self.numItems / len(self.items) <= self.loadMin:
                self.items = Set.__rehash(self.items,[None]*(len(self.items)//
↪2))

    # f) intersection
    def intersect(self,other):
        res = []
        for i in self:
            if i in other:
                res.append(i)
        return Set(res)

    # g) union
    def union(self,other):
        res = []
        for i in self:
            res.append(i)
        for i in other:
```

```
            if i not in self:
                res.append(i)
        return Set(res)

    # h) difference
    def differ(self,other):
        res = []
        for i in self:
            if i not in other:
                res.append(i)
        return Set(res)

    # i) check subset
    def subsetof(self,other):
        for i in self:
            if i not in other:
                return False
        return True

    # j) length
    def length(self):
        return self.numItems

    # k) check if two given sets have no elements in common
    def noCommonWith(self,other):
        for i in self:
            if i in other:
                return False
        return True

    # l) remove intersection
    def removeIntersection(self,other):
        for i in self:
            if i in other:
                self.remove(i)

    # help to show the output
    def print(self):
        for e in self:
            print(e,end=' ')
        print()

    ## Hidden Helper Class
    class __Placeholder:
        def __init__(self):
            pass
        def __eq__(self,other):
```

```
            return False

    ## private functions
    def __add(item,items):
        index = hash(item) % len(items)
        location = -1
        while items[index] != None:
            if items[index] == item:
                return False
            if location < 0 and type(items[index]) == Set.__Placeholder:
                location = index
            index = (index + 1) % len(items)
        if location < 0:
            location = index
        items[location] = item
        return True

    def __remove(item,items):
        index = hash(item) % len(items)
        while items[index] != None:
            nextIndex = (index + 1) % len(items)
            if items[index] == item:
                if items[nextIndex] == None: # check the next one
                    items[index] = None
                else:
                    items[index] = Set.__Placeholder()
                return True
            index = nextIndex
        return False

    def __rehash(olditems,newitems):
        for e in olditems:
            if e != None and type(e) != Set.__Placeholder:
                Set.__add(e,newitems)
        return newitems
```

3 Test Input and Output

[2]:
```
# a) create a set
s = Set([i*2 for i in range(10)])
s.print()
```

```
0 2 4 6 8 10 12 14 16 18
```

[3]:
```
# b) iterate over sets
for i in s:
```

```
    print(i,end=' ')
print()
```

```
0 2 4 6 8 10 12 14 16 18
```

[4]:
```
# c) add member(s)
print("c)")
s.add(23)
print("add 23")
s.print()
s.addlist([34,45,56,67])
print("add a list")
s.print()
```

```
c)
add 23
0 2 23 4 6 8 10 12 14 16 18
add a list
0 2 4 45 6 8 10 12 14 16 56 18 23 67 34
```

[5]:
```
# d) remove item(s)
print("d)")
s.remove(45)
print("remove 45")
s.print()
s.removelist([23,67])
print("remove a list")
s.print()
```

```
d)
remove 45
0 2 4 6 8 10 12 14 16 56 18 23 67 34
remove a list
0 2 4 6 8 10 12 14 16 56 18 34
```

[6]:
```
# remove element not exist
s.remove(77)
```

```
␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-6-0afee307e854> in <module>
  1 # remove element not exist
----> 2 s.remove(77)
```

```
        <ipython-input-1-f2cd1c7e5e6f> in remove(self, item)
         42                 self.items = Set.__rehash(self.
↪items,[None]*(len(self.items)//2))
         43         else:
    ---> 44             raise KeyError("Item not in the set")
         45
         46     def removelist(self,items):


        KeyError: 'Item not in the set'
```

[7]:
```
# e) remove an item from a set if it is present in the set
print("e)")
s.delete(10)
print("delete 10")
s.print()
s.delete(77)
print("delete non-exist element 77")
s.print()
```

```
e)
delete 10
0 2 4 6 8 12 14 16 56 18 34
delete non-exist element 77
0 2 4 6 8 12 14 16 56 18 34
```

[8]:
```
# f) intersection
print("f)")
s1 = Set([i for i in range(1,10)])
s2 = Set([i for i in range(6,17)])
print("two sets")
s1.print()
s2.print()
print("intersection")
s3 = s1.intersect(s2)
s3.print()
```

```
f)
two sets
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
intersection
6 7 8 9
```

[9]:
```
# g) union
print("g)")
print("two sets")
s1.print()
s2.print()
print("union")
s3 = s1.union(s2)
s3.print()
```

```
g)
two sets
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
union
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

[10]:
```
# h) difference
print("h)")
print("two sets")
s1.print()
s2.print()
print("difference")
s3 = s1.differ(s2)
s3.print()
```

```
h)
two sets
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
difference
1 2 3 4 5
```

[11]:
```
# i) subset of
print("i)")
print("s1,s2,s1-s2")
s1.print()
s2.print()
s3 = s1.differ(s2)
s3.print()
print("(s1-s2) is subset of s1? s2?")
print(s3.subsetof(s1),s3.subsetof(s2))
```

```
i)
s1,s2,s1-s2
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5
(s1-s2) is subset of s1? s2?
```

```
True False
```

[12]:
```
# j) length
print("j)")
print("set")
s1.print()
print(f"length is {s1.length()}")
```

```
j)
set
1 2 3 4 5 6 7 8 9
length is 9
```

[13]:
```
# k) no elements in common
print("k)")
s1.print()
s2.print()
print(s1.noCommonWith(s2))
```

```
k)
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
False
```

[14]:
```
# l) remove intersection
print("l)")
s1.print()
s2.print()
s1.removeIntersection(s2)
print("remove intersection")
s1.print()
```

```
l)
1 2 3 4 5 6 7 8 9
6 7 8 9 10 11 12 13 14 15 16
remove intersection
1 2 3 4 5
```