3. Code:

# operations of Proto-vEB-Tree: Successor, Insert and Delete  
# author: Yutong Gao  
# email: gyt@bu.edu  
import math  
  
  
class protovEB:  
 *'Basic functions high(), low(), index() and operations Successor(), Insert() and Delete()'* # u is the universe size of data structure which is 256 in this case  
 # high(x) = x / (sqrt(u)): a given value x resides in cluster number high(x)  
 def high(self, x):  
 return int(math.floor(x / math.sqrt(self.u)))  
  
 # low(x) = x % (sqrt(u)): low(x) is x’s position within its cluster  
 def low(self, x):  
 return int((x % math.sqrt(self.u)))  
  
 # index(x, y) = x \* sqrt(u) + y : x is high(x) and y is low(x)  
 def index(self, x, y):  
 return int((x \* math.sqrt(self.u)) + y)  
  
 # the Proto-vEB-Tree data structure store u, summary, cluster, in this case we will initial a Proto\_vEB\_Tree with u = 256  
 def \_\_init\_\_(self, u):  
 self.u = u  
 # base case  
 if self.u == 2:  
 self.A = [0, 0]  
 else:  
 self.summary = protovEB(int(math.sqrt(self.u)))  
 self.cluster = [protovEB(int(math.sqrt(self.u)))  
 for i in range(int(math.sqrt(self.u)))]  
  
 # Member is an operation to check whether x is in array A or not  
 def member(self, x):  
 # base case  
 if self.u == 2:  
 if self.A[x] == 1:  
 return True  
 else:  
 return False  
 # search the binary number in x's position  
 else:  
 return self.cluster[self.high(x)].member(self.low(x))  
  
 # Find the minimum element of proto-vEB-Tree  
 def vebmin(self):  
 #base case  
 if self.u == 2:  
 if self.A[0] == 1:  
 return 0  
 elif self.A[1] == 1:  
 return 1  
 else:  
 return None  
 # go to the first cluster that has element is not 0 and return the first none-zero element  
 else:  
 min\_cluster = self.summary.vebmin()  
 if min\_cluster is None:  
 return None  
 else:  
 offset = self.cluster[min\_cluster].vebmin()  
 return self.index(min\_cluster, offset)  
  
 # Find successor of x  
 def successor(self, x):  
 # base case only one condition exists successor: we ask the successor of A[0] and A[1] exists.  
 if self.u == 2:  
 if x == 0 and self.A[1] == 1:  
 return 1  
 else:  
 return None  
 # if there is a successor inside the same cluster of x, return it  
 # else find the next cluster contains elements and return the first element.  
 else:  
 offset = self.cluster[self.high(x)].successor(self.low(x))  
 if offset is not None:  
 return self.index(self.high(x), offset)  
 else:  
 succ\_cluster = self.summary.successor(self.high(x))  
 if succ\_cluster is None:  
 return None  
 else:  
 offset = self.cluster[succ\_cluster].vebmin()  
 return self.index(succ\_cluster, offset)  
  
 # Insert the element x into the array(assume x inside universe)  
 def insert(self, x):  
 # base case just set the boolean value to 1 of A[x]  
 if self.u == 2:  
 self.A[x] = 1  
 # set the corresponding position inside the cluster to 1 and set the related summary to 1  
 else:  
 self.cluster[self.high(x)].insert(self.low(x))  
 self.summary.insert(self.high(x))  
  
 # Delete the element x of the array  
 def delete(self, x):  
 # base case just set A[x] to 0  
 if self.u == 2:  
 self.A[x] = 0  
 # we need to set the A[x] to 0 and check if the cluster is empty, if empty then set its summary to 0.  
 else:  
 self.cluster[self.high(x)].delete(self.low(x))  
 cluster\_empty = False  
 for n in range(0, int(math.sqrt(self.u))):  
 if self.cluster[self.high(x)].member(n):  
 cluster\_empty = True  
 break  
 if cluster\_empty is False:  
 self.summary.delete(self.high(x))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # open the hw2test.txt and get the operations in it  
 f = open('hw2test.txt')  
 lines = f.readlines()  
 operation = []  
 for item in lines:  
 content = item.strip()  
 temp = content.split(" ")  
 operation.append(temp)  
 # build a proto\_vEB\_tree  
 tree = protovEB(256)  
 for i in operation:  
 if i[0] is 'S':  
 print(tree.successor(int(i[1]))) # print successor of x  
 elif i[0] is 'I':  
 tree.insert(int(i[1])) # insert x  
 elif i[0] is 'D':  
 tree.delete(int(i[1])) # delete x

The result is

