PS. 4

Ex. 1

We construct two self-balancing BSTs, BST_S and BST_D, storing respectively the actual elements in S, and wsfs $[d, count]$ where d is the distance between two elements in S and $count$ is the frequency of said distance in S.

```
def ADD(S,x):
    if x not in BST_S:
        BST_S.insert(x)
        prev, next = find_prev_next(BST_S, x)
        if prev != None:
            if d(x, prev) not in BST_D:
                BST_D.insert([d(x, prev), 1])
            else:
                increase count of d(x, prev) in BST_D
        if next != None:
            if d(x, next) not in BST_D:
                BST_D.insert([d(x, next), 1])
            else:
                increase count of d(x, next) in BST_D
        if prev != None and next != None:
            decrease count of d(prev, next) in BST_D
```

```
def find_prev_next(BST, z):
    prev = find greatest value in BST ≤ z
    next = find smallest value in BST ≥ z
    return prev, next
```

running time: $O(\log |BST|)$

```
def REMOVE(S,x):
    if x in BST_S:
        BST_S.remove(x)
        prev, next = find_prev_next(BST_S, x)
        if prev != None:
            if count of d(x, prev) == 1:
                BST_D.remove([d(x, prev), 1])
            else:
                decrease count of d(x, prev) in BST_D
        if next != None:
            if count of d(x, next) == 1:
                BST_D.remove([d(x, next), 1])
            else:
                decrease count of d(x, next) in BST_D
        if prev != None and next != None:
            increase count of d(prev, next) in BST_D
```

```
def MINDIST(S):
    if |S| ≤ 1:
        return +∞
    return min(BST_D[0])
```

→ $O(\log |S|)$