

1 Sequential Search

```
1 def sequential_search_itr(xs, y):
2     '''
3     Check whether y is contained in the list xs
4
5     >>> sequential_search_itr([1, 3, 5, 4, 2, 0], 2)
6     True
7     >>> sequential_search_itr([1, 3, 5, 4, 2, 0], 6)
8     False
9     '''
10    for x in xs:
11        if x == y:
12            return True
13    return False

1 def sequential_search_itr2(xs, y):
2     '''
3     Check whether y is contained in the list xs
4
5     >>> sequential_search_itr2([1, 3, 5, 4, 2, 0], 2)
6     True
7     >>> sequential_search_itr2([1, 3, 5, 4, 2, 0], 6)
8     False
9     '''
10    for i in range(len(xs)):
11        if xs[i] == y:
12            return True
13    return False

1 def sequential_search_rec(xs, y):
2     '''
3     Check whether y is contained in the list xs
4
5     >>> sequential_search_rec([1, 3, 5, 4, 2, 0], 2)
6     True
7     >>> sequential_search_rec([1, 3, 5, 4, 2, 0], 6)
8     False
9     '''
10    if len(xs) == 0:
11        return False
12    if xs[0] == y:
13        return True
14    return sequential_search_rec(xs[1:], y)
```

```

1 def sequential_search_rec2(xs, y):
2     '''
3     Check whether y is contained in the list xs
4
5     >>> sequential_search_rec2([1, 3, 5, 4, 2, 0], 2)
6     True
7     >>> sequential_search_rec2([1, 3, 5, 4, 2, 0], 6)
8     False
9     '''
10    def go(xs):
11        if len(xs) == 0:
12            return False
13        if xs[0] == y:
14            return True
15        return go(xs[1:])
16    return go(xs)

1 def sequential_search_rec3(xs, y):
2     '''
3     Check whether y is contained in the list xs
4
5     >>> sequential_search_rec3([1, 3, 5, 4, 2, 0], 2)
6     True
7     >>> sequential_search_rec3([1, 3, 5, 4, 2, 0], 6)
8     False
9     '''
10    def go(i):
11        if i == len(xs):
12            return False
13        if xs[i] == y:
14            return True
15        return go(i+1)
16    return go(0)

```

2 Binary Search

```
1 def binary_search_itr(xs, y):
2     '''
3     Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
4     Return True if y is in the list xs.
5
6     >>> binary_search_itr([1, 3, 5, 7, 9, 11], 9)
7     True
8     >>> binary_search_itr([1, 3, 5, 7, 9, 11], 8)
9     False
10    >>> binary_search_itr(list(range(-1001, 1001, 2)), 9)
11    True
12    >>> binary_search_itr(list(range(-1000, 1000, 2)), 9)
13    False
14    '''
15    if len(xs) == 0:
16        return False
17    left = 0
18    right = len(xs) - 1
19
20    while left != right:
21        mid = (left + right) // 2
22        if xs[mid] > y:
23            right = mid
24        if xs[mid] < y:
25            left = mid + 1
26        if xs[mid] == y:
27            return True
28            left = mid + 1
29
30    if xs[left] == y:
31        return True
32    else:
33        return False
34
35    return go(0, len(xs) - 1)
```

```

1 def binary_search_rec(xs, y):
2     '''
3     Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
4     Return True if y is in the list xs.
5
6     >>> binary_search_rec([1, 3, 5, 7, 9, 11], 9)
7     True
8     >>> binary_search_rec([1, 3, 5, 7, 9, 11], 8)
9     False
10    >>> binary_search_rec(list(range(-1001, 1001, 2)), 9)
11    True
12    >>> binary_search_rec(list(range(-1000, 1000, 2)), 9)
13    False
14    '''
15    if len(xs) == 0:
16        return False
17
18    def go(left, right):
19        if left == right:
20            if xs[left] == y:
21                return True
22            else:
23                return False
24        mid = (left + right) // 2
25        if xs[mid] > y:
26            right = mid
27        if xs[mid] < y:
28            left = mid + 1
29        if xs[mid] == y:
30            return True
31        left = mid + 1
32        return go(left, right)
33
34    return go(0, len(xs) - 1)

```

```

1 def binary_search_rec2(xs, y):
2     '''
3     Assume that xs is a list of numbers sorted from LOWEST to HIGHEST.
4     Return True if y is in the list xs.
5
6     >>> binary_search_rec2([1, 3, 5, 7, 9, 11], 9)
7     True
8     >>> binary_search_rec2([1, 3, 5, 7, 9, 11], 8)
9     False
10    >>> binary_search_rec2(list(range(-1001, 1001, 2)), 9)
11    True
12    >>> binary_search_rec2(list(range(-1000, 1000, 2)), 9)
13    False
14    '''
15    if len(xs) == 0:
16        return False
17
18    mid = len(xs) // 2
19    if xs[mid] > y:
20        return binary_search_rec2(xs[:mid], y)
21    if xs[mid] < y:
22        return binary_search_rec2(xs[mid+1:], y)
23    if xs[mid] == y:
24        return True

```