

# Data Flow

## array\_merger.py

```
__author__ = "Zelin Cai, Patrick Silvestre"
__version__ = "0.1.0"
__license__ = "MIT"

def array_merger(list1, list2):
    """ Variables to iterate through the lists """
    i = j = 0
    merged_list = []
    while i < len(list1) and j < len(list2):
        if list1[i] <= list2[j]:
            merged_list.append(list1[i])
            i += 1
        else:
            merged_list.append(list2[j])
            j += 1

    """ Checks if there are any index values remaining in list1 and appends them """
    while i < len(list1):
        merged_list.append(list1[i])
        i += 1

    """ Has the same purpose as the previous while loop but for list2 instead """
    while j < len(list2):
        merged_list.append(list2[j])
        j += 1

    return merged_list
```

## test\_array\_merger.py

```
__author__ = "Zelin Cai, Patrick Silvestre"
__version__ = "0.1.0"
__license__ = "MIT"

from array_merger import *
import unittest

class TestTwoEmptyLists(unittest.TestCase):
    def test_01_two_empty_lists(self):
        """
        Nodes:
        1-2-3-8-10-12
        """
        list1 = []
        list2 = []
        expected_output = []
        actual_output = array_merger(list1, list2)
        self.assertEqual(expected_output, actual_output)

class TestOneEmptyList(unittest.TestCase):
```

```

def test_02_first_list_empty(self):
    """
    Nodes:
    1-2-3-8-
      10-11-
      10-12
    """
    list1 = []
    list2 = [0]
    expected_output = [0]
    actual_output = array_merger(list1, list2)
    self.assertEqual(expected_output, actual_output)

def test_03_second_list_empty(self):
    """
    Nodes:
    1-2-3-
      8-9-
      8-10-12
    """
    list1 = [0]
    list2 = []
    expected_output = [0]
    actual_output = array_merger(list1, list2)
    self.assertEqual(expected_output, actual_output)

class TestFullLists(unittest.TestCase):
    def test_04_list1_with_leftover_values(self):
        """
        Nodes:
        1-2-
          3-4-6-7-
          3-4-6-7-
          3-4-6-7-
          3-
            8-9-
            8-9-
            8-9-
            8-9-
            8-10-12
        """
        list1 = [3, 4, 5, 6]
        list2 = [0, 1, 2]
        expected_output = [0, 1, 2, 3, 4, 5, 6]
        actual_output = array_merger(list1, list2)
        self.assertEqual(expected_output, actual_output)

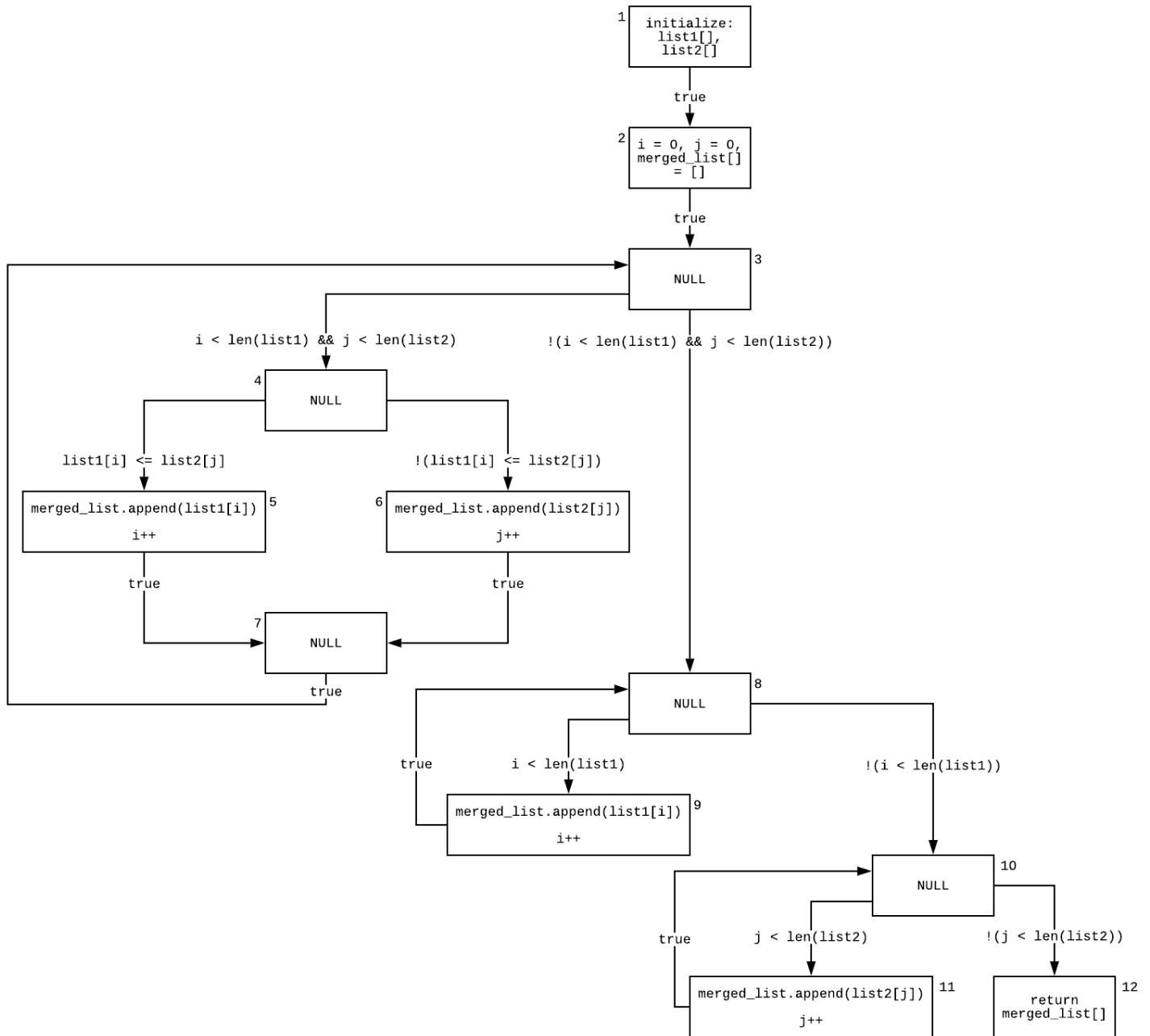
    def test_05_list2_with_leftover_values(self):
        """
        Nodes:
        1-2-
          3-4-5-7-
          3-4-5-7-
          3-4-5-7-
          3-8-
            10-11-
            10-11-
            10-11-
            10-11-
            10-12
        """
        list1 = [0, 1, 2]
        list2 = [3, 4, 5, 6]
        expected_output = array_merger(list1, list2)
        actual_output = array_merger(list1, list2)
        self.assertEqual(expected_output, actual_output)

```

```
def test_06_same_sized_lists(self):
    """
    Nodes:
    1-2-3-4-5-7-
      3-4-6-7-
      3-4-5-7-
      3-4-6-7-
      3-4-5-7-
      3-4-6-7-8-10-12
    """
    list1 = [0, 2, 4]
    list2 = [1, 3, 5]
    expected_output = [0, 1, 2, 3, 4, 5]
    actual_output = array_merger(list1, list2)
    self.assertEqual(expected_output, actual_output)
```

## Data Flow Graph

---



## Independent Paths

- Path 1: 1 - 2 - 3 - 8 - 10 - 12
- Path 2: 10 - 11 - 10
- Path 3: 8 - 9 - 8
- Path 4: 2 - 3 - 4 - 5 - 7
- Path 5: 3 - 4 - 6 - 7 - 3

## Simple Paths

Paths 1, 2, 3, 4 and 5 are all simple paths since all of the nodes are distinct aside from the first and last nodes in paths 2, 3 and 5.

## Loop-Free Paths

Paths 1 and 4 are loop-free paths since every node is distinct.

## def(), c-use(), p-use()

Nodes <b>i</b>	<b>def(i)</b>	<b>c-use(i)</b>
1	{ list1[], list2[] }	{ }
2	{ i, j, merged_list[] }	{ }
3	{ }	{ }
4	{ }	{ }
5	{ merged_list[], i }	{ i, list1[i] }
6	{ merged_list[], j }	{ j, list2[j] }
7	{ }	{ }
8	{ }	{ }
9	{ merged_list[], i }	{ i, list1[i] }
10	{ }	{ }
11	{ merged_list[], j }	{ j, list2[j] }
12	{ }	{ merged_list[] }

Edges (i, j)	<b>predicate(i, j)</b>	<b>p-use(i, j)</b>
(1, 2)	true	{ }
(2, 3)	true	{ }
(3, 4)	i < len(list1) && j < len(list2)	{ i, list1, j, list2 }
(4, 5)	list1[i] <= list2[j]	{ list1[i], list2[j] }
(4, 6)	!(list1[i] <= list2[j])	{ list1[i], list2[j] }
(5, 7)	true	{ }
(6, 7)	true	{ }
(7, 3)	true	{ }
(3, 8)	!(i < len(list1) && j < len(list2))	{ i, list1, j, list2 }
(8, 9)	i < len(list1)	{ i, list1 }
(9, 8)	true	{ }
(8, 10)	!(i < len(list1))	{ i, list1 }
(10, 11)	j < len(list2)	{ j, list2 }
(11, 10)	true	{ }
(10, 12)	!(j < len(list2))	{ j, list2 }

k XNkh kX2kḱ” lā nā ək ɛl 2kḱ” lā nxl ƶXkrib kXk

hŷ Ūxɛ ƶX nɛnjnɔkm̃l ƶXkŋT X m̃ñ Hñl m̃ NɛT ñX! Ɔ L ́ÍdGĐě Ĩvđĭ

ƶ ƶā 6Ė i

l ó ħ n̄	ƶXkrib ƶXkɪ	Hɪ n̄
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ƶɪ	<sup>8</sup> ŋ <sup>+</sup> ŋ	<sup>6</sup> ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ôɪ	<sup>5</sup> ŋ <sup>6</sup> ŋ <sup>7</sup>	<sup>6</sup> ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ƶɪ	<sup>+</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ôɪ	<sup>8</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ	<sup>+</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ƶɪ	<sup>7</sup> ŋ <sup>8</sup>	<sup>6</sup> ǎ ǎǎǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ôɪ	<sup>5</sup> ŋ <sup>6</sup> ŋ <sup>7</sup> ŋ <sup>8</sup> ŋ <sup>+</sup> ŋ	<sup>6</sup> ǎ ǎǎǎ ǎ <sup>0</sup>
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ	<sup>7</sup> ŋ <sup>8</sup>	<sup>6</sup> ǎ ǎǎǎ ǎ

ƶ ƶā 6Ė j

l ó ħ n̄	ƶXkrib ƶXkɪ	Hɪ n̄
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ƶɪ	<sup>8</sup> ŋ <sup>+</sup> ŋ	<sup>6</sup> ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ôɪ	<sup>5</sup> ŋ <sup>6</sup> ŋ <sup>7</sup>	<sup>6</sup> ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ƶɪ	<sup>+</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ŋ <sup>6</sup> ôɪ	<sup>8</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ	<sup>8</sup> ŋ	<sup>6</sup> ǎ ǎ ǎ
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ <sup>0</sup> ŋ <sup>6</sup> ƶɪ	<sup>6</sup> ŋ <sup>+</sup>	<sup>6</sup> ǎ ǎǎǎ <sup>5 0</sup> ǎ <sup>5</sup>
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ <sup>0</sup> ŋ <sup>6</sup> ôɪ	<sup>5</sup> ŋ <sup>6</sup> ŋ <sup>7</sup> ŋ <sup>8</sup> ŋ <sup>+</sup> ŋ	<sup>6</sup> ǎ ǎǎǎ <sup>5 0</sup> ǎ <sup>6</sup>
h̄ <sup>6</sup> ŋ <sup>6</sup> ɪ <sup>5</sup> ɪ	<sup>6</sup> ŋ <sup>+</sup>	<sup>6</sup> ǎ ǎǎǎ <sup>5 5</sup>

!