Solve

Set A

```
def find(self, num_vertices, adj_list):
        max_outgoing = -1
        vertex_with_max = -1

        for i in range(self.num_vertices):
            count = 0
            current = self.adj_list[i]
            while current:
                count += 1
                current = current.next

            if count > max_outgoing):
                max_outgoing = count
                vertex_with_max = i

        if max_outgoing < 2:
            return -1

        outgoing_weights =
np.zeros(max_outgoing, dtype=int)
        current = self.adj_list[vertex_with_max]
        index = 0

        while current:
            if current.weight > 5:
                outgoing_weights[index] =
current.weight
                index += 1
            current = current.next

        return outgoing_weights
```

```
public int[] find(int numVertices, Node[]
adjLis) {
    int maxOutgoing = -1;
    int vertexWithMax = -1;

    for (int i = 0; i < numVertices; i++) {
        int count = 0;
        Node current = adjList[i];
        while (current != null) {
            count++;
            current = current.next;
        }

        if (count > maxOutgoing) {
            maxOutgoing = count;
            vertexWithMax = i;
        }
    }

    if (maxOutgoing < 2) {
        return new int[]{};
    }

    int[] outgoingWeights = new
int[maxOutgoing];
    Node current = adjList[vertexWithMax];
    int index = 0;

    while (current != null) {
        if (current.weight > 5) {
            outgoingWeights[index] =
current.weight;
        } else {
            outgoingWeights[index] = 0;
        }
        index++;
        current = current.next;
    }

    return outgoingWeights;
}
```

Set B

```
def find(self, num_vertices, adj_list):
        min_outgoing = float('inf')
        vertex_with_min = -1

        for i in range(self.num_vertices):
            count = 0
            current = self.adj_list[i]
            while current:
                count += 1
                current = current.next

            if 0 < count < min_outgoing:
                min_outgoing = count
                vertex_with_min = i

        if vertex_with_min == -1:
            return -1

        outgoing_weights =
np.zeros(min_outgoing, dtype=int)
        current = self.adj_list[vertex_with_min]
        index = 0

        while current:
            if current.weight < 5:
                outgoing_weights[index] =
current.weight
                index += 1
            current = current.next

        return outgoing_weights
```

```
public int[] find(int numVertices, Node[]
adjList) {
    int minOutgoing = Integer.MAX_VALUE;
    int vertexWithMin = -1;

    for (int i = 0; i < numVertices; i++) {
        int count = 0;
        Node current = adjList[i];
        while (current != null) {
            count++;
            current = current.next;
        }

        if (count > 0 && count < minOutgoing)
{
            minOutgoing = count;
            vertexWithMin = i;
        }
    }

    if (vertexWithMin == -1) {
        return new int[]{};
    }

    int[] outgoingWeights = new
int[minOutgoing];
    Node current = adjList[vertexWithMin];
    int index = 0;

    while (current != null) {
        if (current.weight < 5) {
            outgoingWeights[index] =
current.weight;
            index++;
        }
        current = current.next;
    }

    return outgoingWeights;
}
```

Rubric

Giving correct parameters	1
Finding max/min outgoing edged vertex	5
Returning -1 / int [] {}	2
Initializing result array with proper size	2
Selecting the correct vertex for building the resulting array	1
Building Resulting array	3
Returning proper array	1
Total	15